Deep Learning & Applications
-
time series/recommendations/graph generation

Michalis Vazirgiannis

Data Science & Mining group
LIX, Ecole Polytechnique,

DASCIM web page: http://www.lix.polytechnique.fr/dascim
Google Scholar: https://bit.ly/2rwmvQU
Twitter: @mvazirg

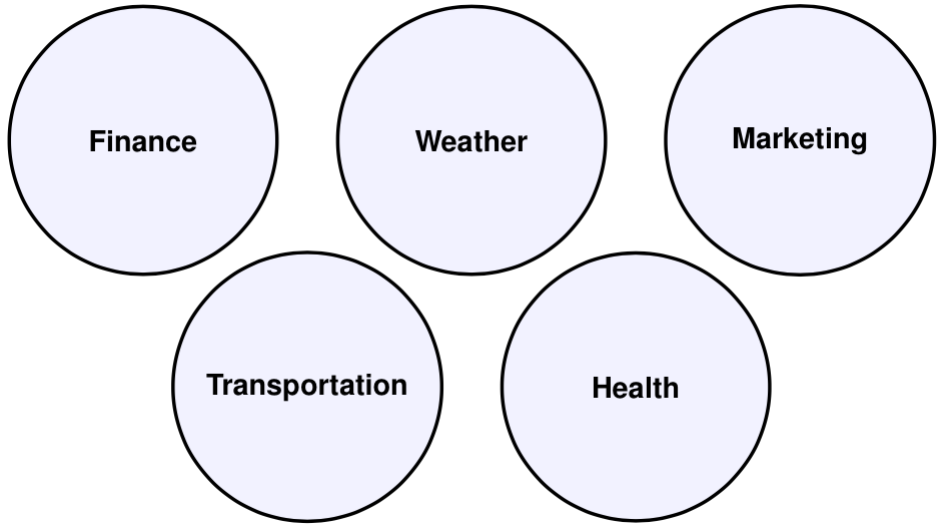
December 2024

- **Time Series**
- Recommendations
- Graph generation

TIME SERIES - Motivation

Why focus on Times Series models?

1. **Constant need to predict the future.**
2. **Focus on minimizing risk, developing long-term strategies, making decisions instantly.**
3. **Availability of a great amount of data.**
4. **Impact on a huge range of applications.**



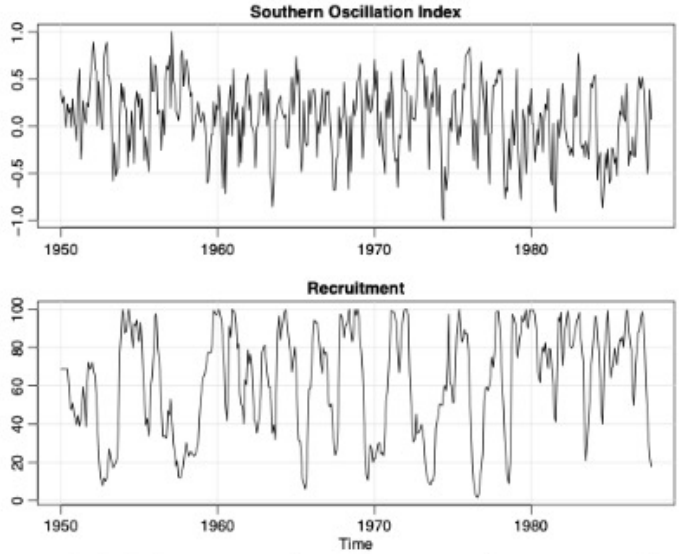
A Time Series Example - Stock prices

APPLE, TESLA stocks, A 5-year weekly evolution[1]

[1] *source: finance.yahoo.com*

A Time Series Example - weather cycles and fish

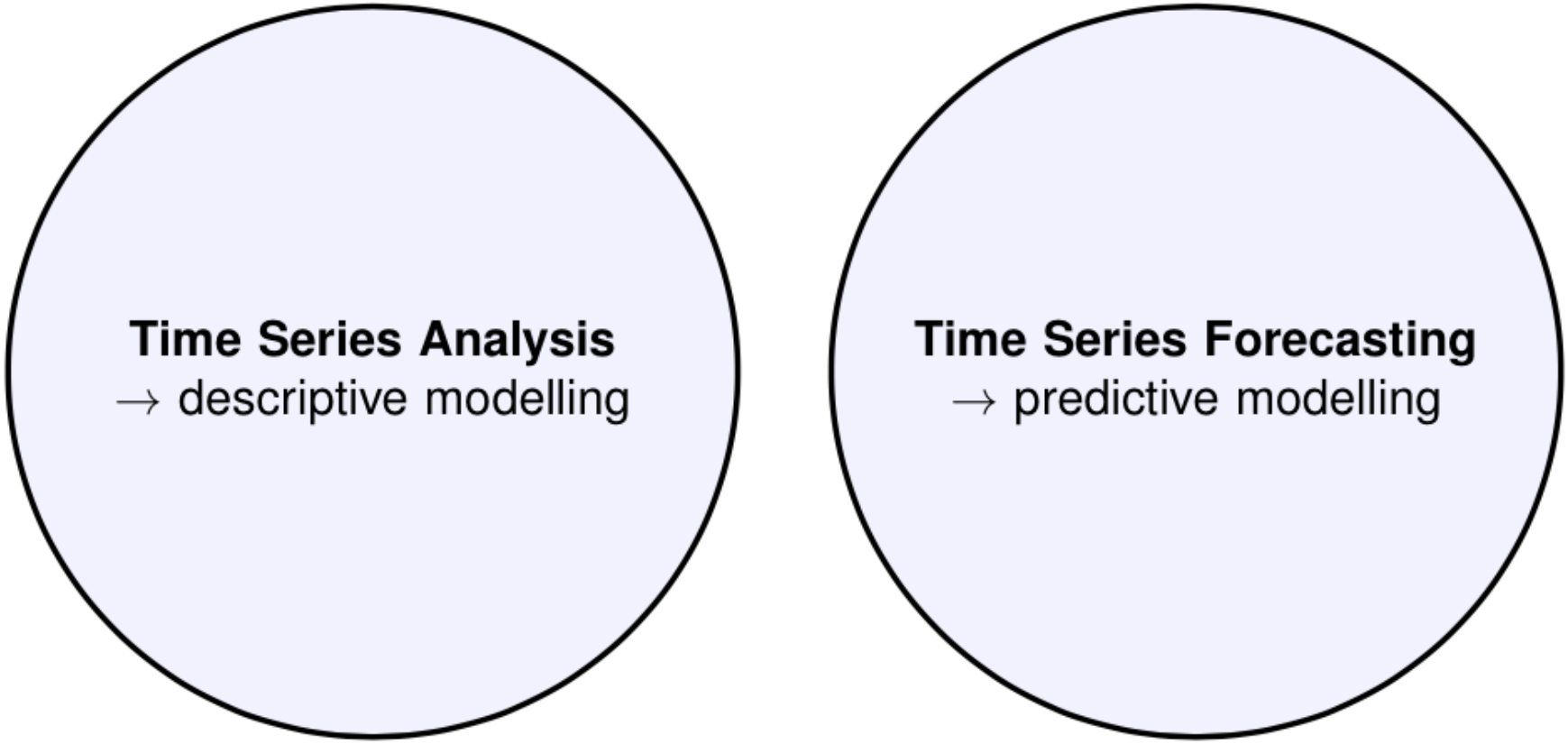
Monthly SOI and Recruitment (estimated new fish), 1950-1987 [4]



- SOI: changes in air pressure, related to sea surface temperatures - central Pacific Ocean.
- Central Pacific warms every 3-7 years (El Niño effect blamed for global extreme weather).
- Both series exhibit repetitive behavior, (cycles) - underlying processes: rate or frequency of oscillation

[4] R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications, With R Examples. Springer Texts in Statistics

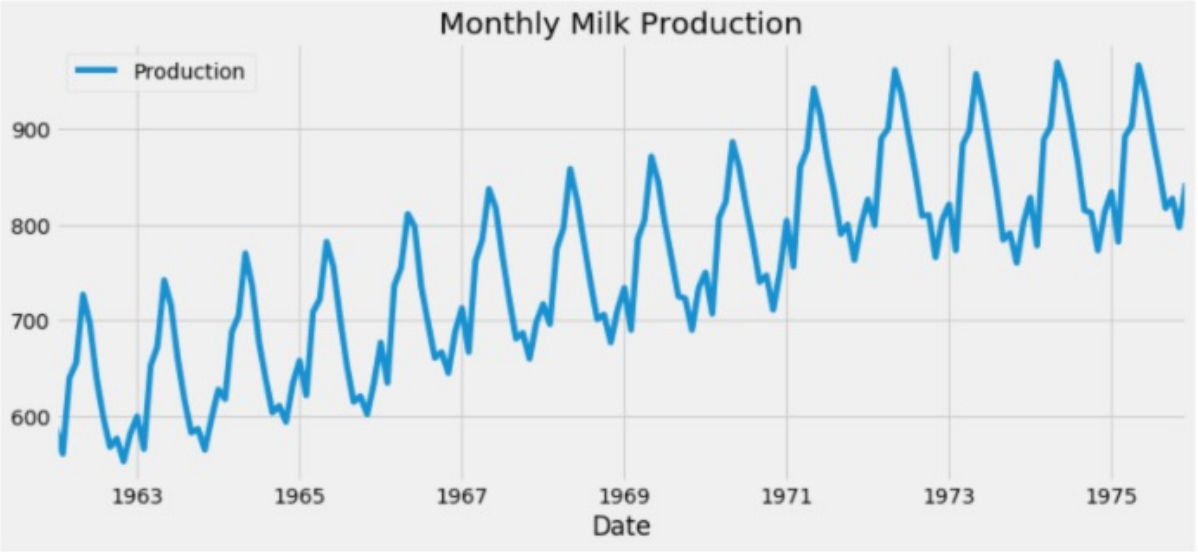
Time Series - Domains of study



Components of a Time Series

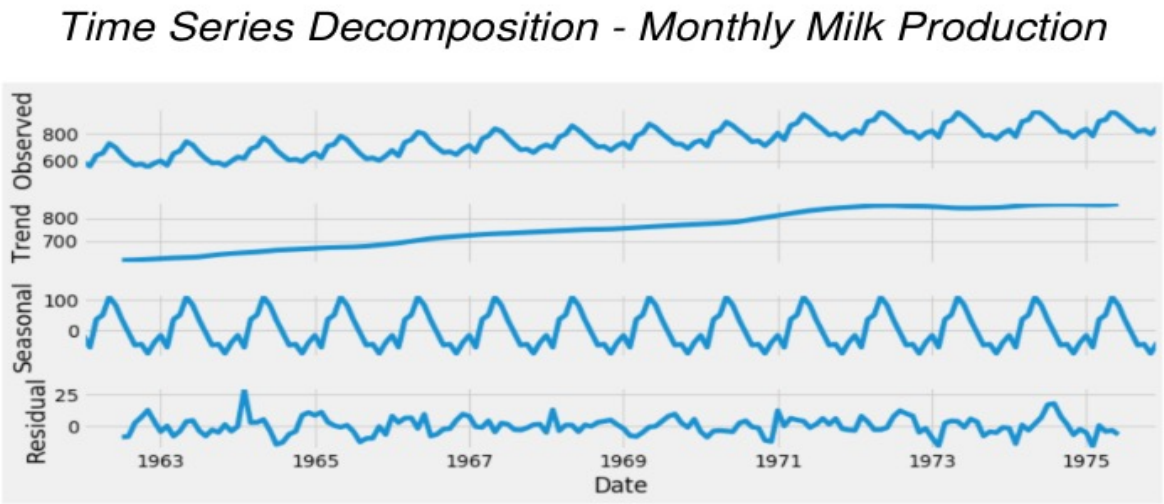
The first step in analyzing a time series for a predictive model is to identify the underlying pattern of the data over time.

Milk Production in pounds per cow



In general, a time series is affected by four components, i.e. **trend, seasonal, cyclical and irregular components**.

Components of a Time Series



- **Trend :**
  The long-term gradual change in the series. This is the simplest trend pattern, as it demonstrates long-term growth or decline.
- **Seasonality:**
  Predictable, short-term patterns that occur within a single unit of time and repeat indefinitely.
- **Noise or irregular variation (residual)**

Random variation not regular and also do not repeat in a particular pattern, caused by incidences such as war, strike, earthquake etc.

Time Series Decomposition

Based on these four components, a time series can be decomposed using the following approaches. Decomposition is primarily used for time series analysis, and as an analysis tool it can be used to inform forecasting models on your problem.

1. **Additive Model:**
   - $Y(t) = T(t) + S(t) + I(t)$
   - These three components are independent of each other.
   - Assumption: seasonal variation is constant over time.
2. **Multiplicative Model:**
   - $Y(t) = T(t) \times S(t) \times I(t)$
   - These three components are not necessarily independent and they can affect one another.
   - Assumption: seasonal variation *increases* over time.

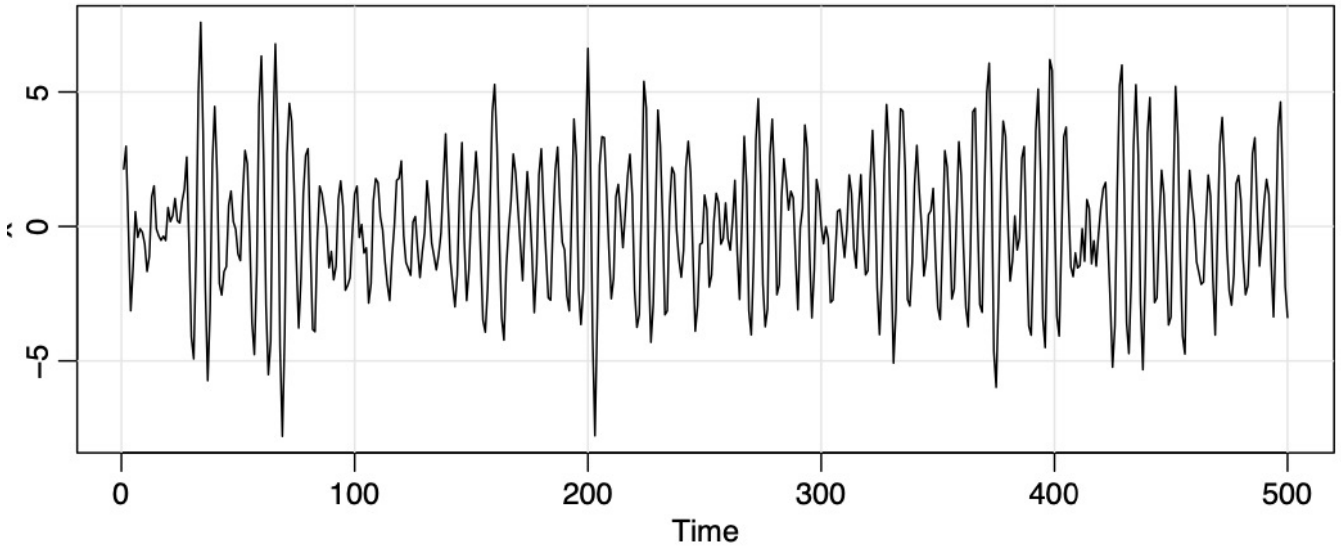
where $T(t)$: trend (with cycles), $S(t)$: seasonal, $I(t)$: irregular (random) components

Time series as a stochastic process

- We can define a time series as a collection of random variables indexed based on the order they are obtained in time, $X_1, X_2, X_3, ...X_t$ will typically be discrete and vary over the integers $t = 0, \pm 1, \pm 2, ...$
- The collection of the random variables X_t is referred to as a stochastic process, while the observed values are referred to as a realization of the stochastic process.
- A time series observed as **a collection of** n **random variables** at arbitrary time points $t_1, t_2, ..., t_n$, for any positive integer n, is provided by the joint distribution function, using the n constants, $c_1, c_2, ..., c_n$ i.e.:
  - $F_{t_1,t_2,...t_n}(c_1, c_2, ..., c_n) = Pr(X_{t_1} \leqslant c_1, X_{t_2} \leqslant c_2, ..., X_{t_n} \leqslant c_n)$

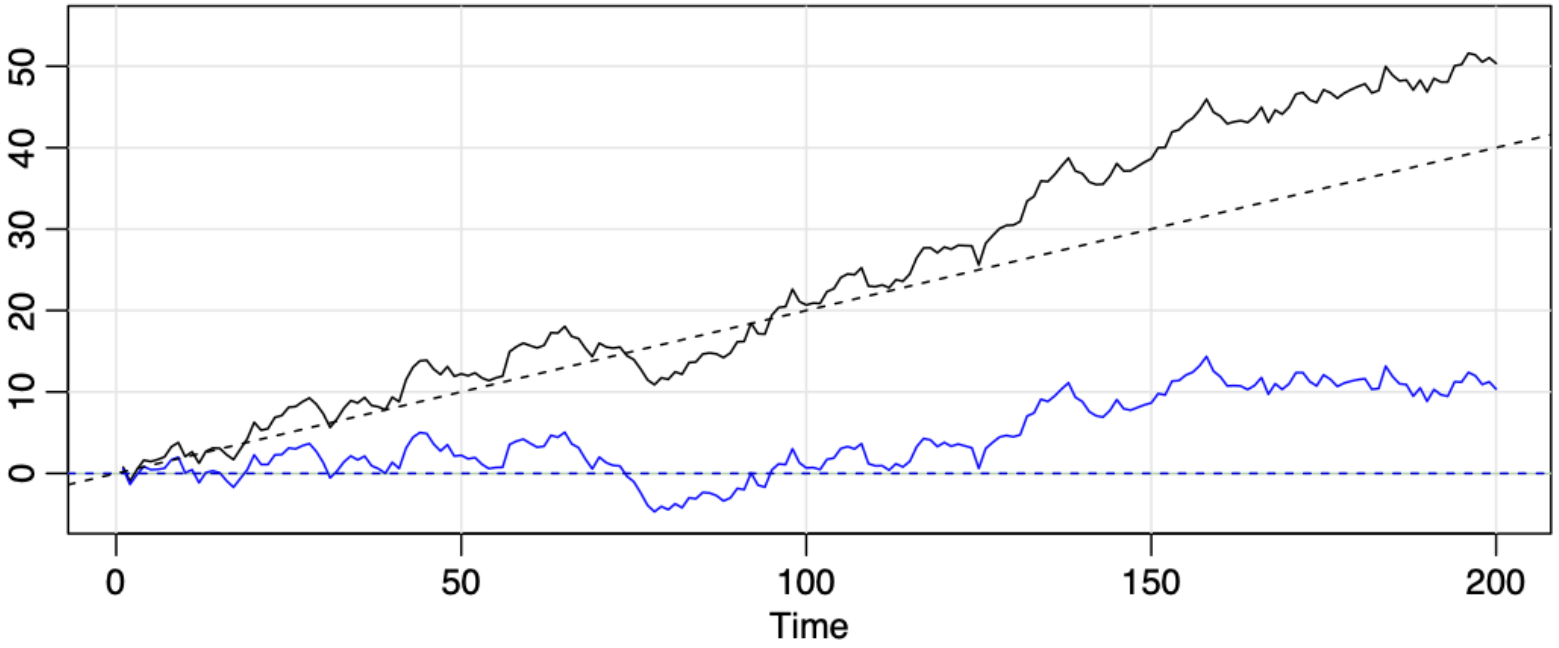
Autoregressive time series

- Assume white noise series w_t : $x_t = x_{t-1} - .9x_{t-2} + w_t$



Random Walk with Drift time series

- *Time series model:* $x_t = \delta + x_{t-1} + w_t$ t = 1, 2, . . ., x_0 = 0, and w_t is white noise. δ: constant is called the *drift*
- when δ = 0, is called simply a *random walk*.

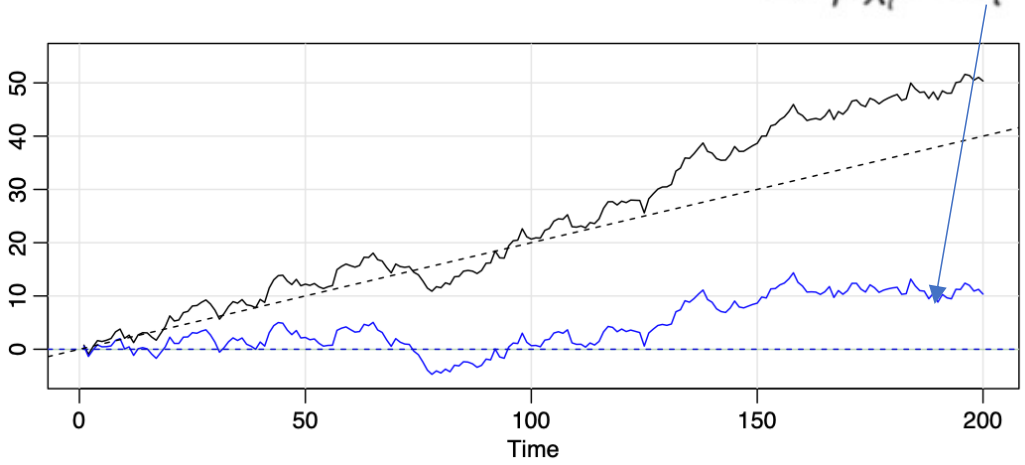


Mean Function - Time Series

- The mean function is defined as $\mu_t = \mu_{X_t} = E[X_t] = -\int_{-\infty}^{\infty} x f_t(x) dx$, provided it exists, where E the usual expected value operator.
- For White Noise: $\mu_{w_t} = E(w_t) = 0$
- For Random Walk:

$$\mu_{X_t} = E[X_t] = E[X_{t-1} + \delta + w_t] = E[\delta_t + w_t] = \delta_t + \sum_{i=1}^{t} E[w_i]$$

$$\Rightarrow \mu_{X_t} = \delta_t$$



Autocovariance - Time Series

- **Autocovariance Function**
  - is defined as the second moment product
    $\gamma(s,t) = \gamma_X(s,t) = cov(X_s, X_t) = E[(X_s - \mu_s)(X_t - \mu_t)], \forall s,t$
  - $\gamma(s,t) = \gamma(t,s), \forall s,t$
- The autocovariance **measures the linear dependence between two points** on the same series observed at different times.

- **cross-covariance function** *between two series,* x_t *and* y_t*, is*

$$\gamma_{xy}(s,t) = \mathrm{cov}(x_s, y_t) = \mathrm{E}[(x_s - \mu_{xs})(y_t - \mu_{yt})].$$

Auto correlation - Time Series

- **Autocorrelation Function (ACF)**
  - $\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}$
  - From Cauchy-Schwarz inequality
    $|\gamma(s,t)|^2 \leqslant \gamma(s,s)\gamma(t,t) \Rightarrow -1 \leqslant \rho(s,t) \leqslant 1$
- ACF measures the **linear predictability of X_t using only X_s**
  - If we can predict X_t perfectly from X_s through a linear relationship, then ACF will be either +1 or -1.
- *The* **cross-correlation function (CCF)** *between two series,* x_t *and* y_t

$$\rho_{xy}(s,t) = \frac{\gamma_{xy}(s,t)}{\sqrt{\gamma_x(s,s)\gamma_y(t,t)}}.$$

Stationary Time Series

- **strictly stationary** *time series: the probabilistic behavior of every collection of values* $\{x_{t1},x_{t2},\ldots,x_{tk}\}$ *is identical to that of the time shifted set* $\{x_{t1+h},x_{t2+h},\ldots,x_{tk+h}\}$ for all *h* or:

$$\Pr\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = \Pr\{x_{t_1+h} \leq c_1, \ldots, x_{t_k+h} \leq c_k\}$$

- **weakly stationary** *time series*
  - The time series $X_t, t \in \mathbb{Z}$ is said to be weak stationary if:
    1. $E[x_t^2] < \infty, \forall t \in \mathbb{Z}$
    2. $E[x_t] = \mu, \forall t \in \mathbb{Z}$
    3. $\gamma_X(s,t) = \gamma_X(s+h, t+h), \forall s,t,h \in \mathbb{Z}$

- *mean value function,* μt *, is constant and does not depend on time t*
- *autocovariance function,* γ(*s*, *t*) *depends on s and t only through their difference* $|s-t|$.
- Stationary datasets are those that have a *stable mean and variance*,

AR Models

Autoregressive Models (AR Models) express the current value of the series, X_t as a linear combination of p past values, $X_{t-1}, X_{t-2}, \ldots, X_{t-p}$, including a random error in the approximation.

Definition: An autoregressive model of order p, **AR(p)**, is defined as

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + w_t \Rightarrow$$

$$\boxed{X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + w_t}$$

where X_t is stationary, $\phi_1, \phi_2, \ldots, \phi_p$ are model parameters (where p the length of historical points) and $w_t \sim wn(0, \sigma_w^2)$.

AR Models: AR(0), AR(1), AR(p)

1. $AR(0)$ is the simplest AR process and has no dependence between the terms (white noise).
2. $AR(1)$ is defined as $X_t = \phi_1 X_{t-1} + w_t$
   - if $|\phi_1| \approx 0$ the process behaves like white noise.
   - if $\phi_1 = 1$ the process is equivalent to random walk with infinite variance (dependent on t, non-stationary).
   - if $\phi_1 < 0$ the process swings between positive and negative values.
3. For the general $AR(p)$ process:
   - if $h > p$ the PACF is $\phi_{hh} = corr(X_{t+h} - \hat{X}_{t+h}, X_t - \hat{X}_t) = corr(w_{t+h}, X_t - \hat{X}_t) = 0$.
   - if $h \leqslant p$ then ϕ_{pp} is not 0.
   - thus the identification of an AR model can be efficiently done with the PACF.

AR Models: Parameters Estimation

1. p is a hyperparameter for the $AR(p)$ process, thus when fitting an AR(p) model p is known and we focus on estimating the coefficients $(\phi_1, \phi_2, \ldots, \phi_p)$.
2. Coefficients estimation can be done by different approaches:
   - Maximum Likelihood Estimation estimator (MLE).
   - Ordinary Least Squares estimator (OLS).

Moving Average models

One problem of AR model is the ignorance of correlated noise structures in the time series. In other words, w_t and its historical terms $w_{t-1}, w_{t-2}, \ldots, w_{t-q}$, include information that can be utilized for predictive models.

Definition: A Moving Average Model (MA) of order q, **MA(q)**, is defined as

$$X_t = \theta_1 w_{t-1} + \theta_2 w_{t-2} + \ldots + \theta_q w_{t-q} + w_t \Rightarrow$$

$$\boxed{X_t = \sum_{j=1}^{q} \theta_j w_{t-j} + w_t}$$

where X_t is stationary, $\theta_1, \theta_2, \ldots, \theta_p$ are model parameters (where q the length of historical points) and $w_t \sim wn(0, \sigma_w^2)$.

- looks like a regression model, the difference is that the w_t is not observable.
- In contrast with AR models, finite MA models are always stationary (observation is a weighted moving average over past forecast errors).

ARMA Models

Autoregressive and Moving Average models can be combined and form **ARMA models**.

Definition: A stationary time series is **ARMA(p, q)** if

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

where $\phi_p, \theta_q \neq 0$ and $w_t \sim wn(0, \sigma_w^2)$.

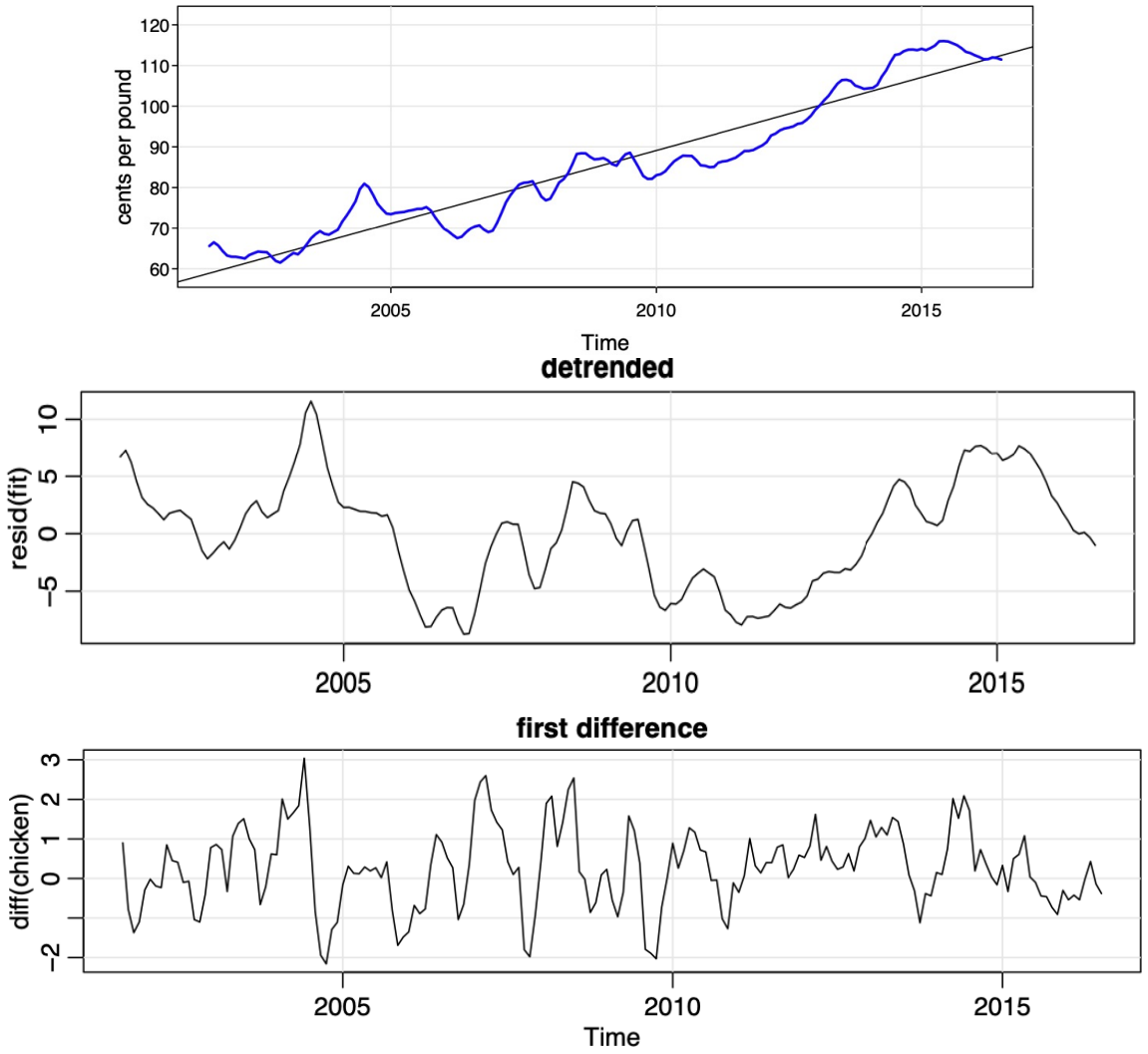
Transforming Time Series to Stationary

One prerequisite of ARMA models is the stationarity of time series

- Assume non-stationary time series: $x_t = \mu_t + y_t$
- x_t are the observations, μ_t trend, y_t is a stationary process.
- Remove the trend: estimate of the trend component: $\hat{\mu}_t$, and then work with the residuals

$$\hat{y}_t = x_t - \hat{\mu}_t$$

- Differencing: $x_t = x_t - x_{t-1}$



History of Time Series

- early 1970s: ARIMA
- late 1970s: Bayesian Approach
- 1980s: GARCH
- 1995: Random Forest
- 1997: Support Vector Regression
- 1999: Boosted Decision Trees
- 2014: LSTMs, GRUs, Seq2Seq
- 2016: Dilated CNNs
- 2017: Transformer

Limitations of Statistical Methods

Focus on

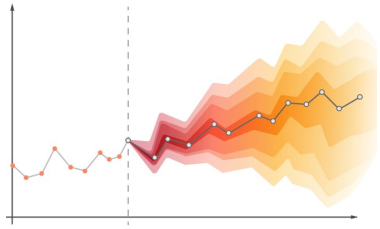
- complete data
- linear relationships
- fixed temporal dependencies
- univariate data
- 1-step forecasts

Why use Deep Learning for Time Series?

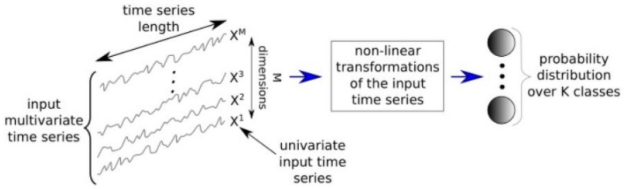
- Learns patterns in data with complex non-linear dependencies
- Supports multiple inputs and outputs
- Has shown good performance in many scenarios
- Models are flexible and expressive
- Can be trained with large datasets
- Easy to introduce exogenous features into the model

Different tasks in Time series

Forecasting
(point or probabilistic)



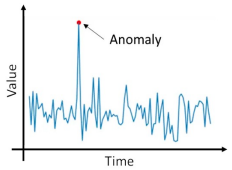
Classification
(to identify a time series as coming from one of possibly many sources or predefined groups)



Other tasks:

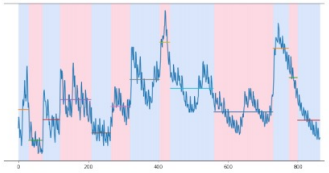
1. Anomaly Detection

Find abnormal events in a time series



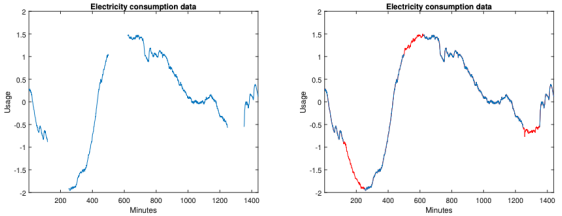
2. Segmentation/ change-point detection

Find significant abrupt changes in the time series



3. Completion/interpolation

Recover missing/lost samples in a time series



4. Query by content/indexation

Given an input time series, retrieve the closest time series in a large database up to a given measure of fit

DL Architectures for Time Series Forecasting

1-step-ahead forecasting:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \boldsymbol{x}_{i,t-k:t}, \boldsymbol{s}_i)$$

Model forecast

observations of target over a look-back window k

exogenous inputs over a look-back window k

static metadata associated with the entity (e.g. sensor location)

multi-horizon forecasting:

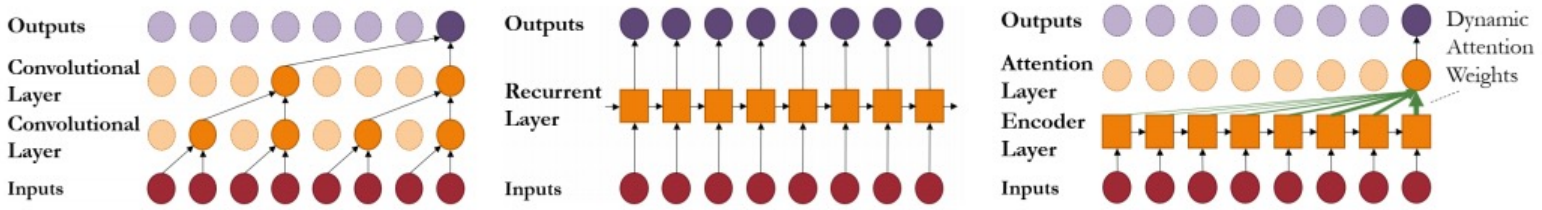
$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{u}_{t-k:t+\tau}, \boldsymbol{s}, \tau)$$

τ is a discrete forecast horizon

known future inputs (e.g. date information, such as the day-of-week or month) across the entire horizon

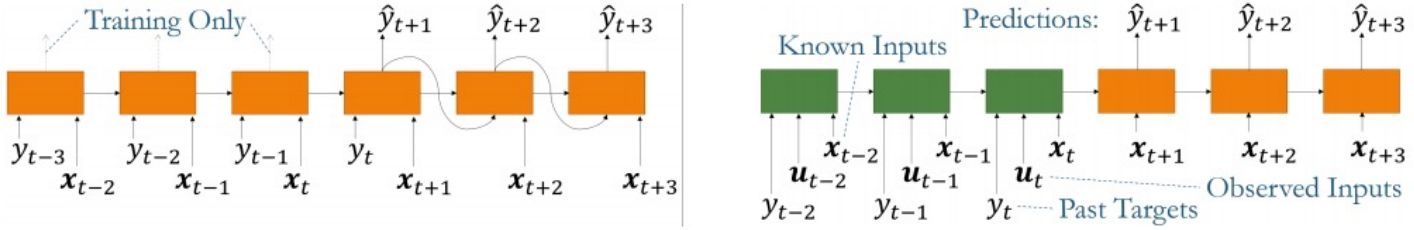
Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

Basic Building Blocks



(a) CNN Model. (b) RNN Model. (c) Attention-based Model.

Multi-horizon Forecasting Models



(a) Iterative Methods (b) Direct Methods

autoregressive deep learning architectures, produce multi-horizon forecasts by recursively feeding samples of the target into future time steps

→ large error accumulations

produce forecasts directly using all available inputs, make use of sequence-to-sequence architectures, (an encoder summarizes past information and a decoder combines them with known future inputs)

→ Fixed maximum forecast horizon

Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

Outputs and Loss functions

1. **Point Estimates:** determine the expected value of a future target. This reformulates the problem to a classification task for discrete outputs (e.g. forecasting future events), and regression task for continuous outputs.

$$\mathcal{L}_{classification} = -\frac{1}{T}\sum_{t=1}^{T} y_t \log(\hat{y}_t) + (1 - y_t)\log(1 - \hat{y}_t)$$

$$\mathcal{L}_{regression} = \frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

2. **Probabilistic Outputs:** model uncertainties and use deep neural networks to generate parameters of known distributions. e.g. Gaussian distributions are used to forecast continuous targets, with the networks outputting means and variance parameters for the predictive distributions at each step:

$$y_{t+\tau} \sim N(\mu(t,\tau), \zeta(t,\tau)^2),$$

$$\mu(t,\tau) = \boldsymbol{W}_\mu \boldsymbol{h}_t^L + \boldsymbol{b}_\mu,$$

Final layer of the network

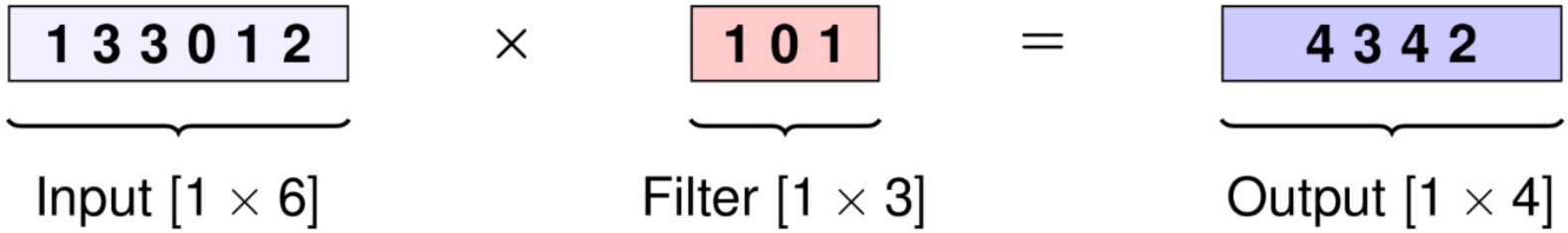
$$\zeta(t,\tau) = \text{softplus}(\boldsymbol{W}_\Sigma \boldsymbol{h}_t^L + \boldsymbol{b}_\Sigma),$$

Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

Convolutional Neural Networks

- Convolution layer
  - units in the hidden layer operate on a field of the output
  - weights are shared across input
- Motivation for CNNs in sequence learning
  - Success of CNN in CV and recently in NLP
  - Achieves state-of-the-art accuracy in audio synthesis and machine translation
- Main advantages:
  - Lower level of model complexity than RNNs
  - Parallelization of computations
  - Dilated CNN can outperform RNN in sequence modelling
- 2D Convolutions encode spatial invariance
- 1D Convolutions encode temporal invariance, “stationarity”

1D Convolutions



1D Convolution with 1 filter (*stride* $= 1$), $t = 6$, $\#features = 1$

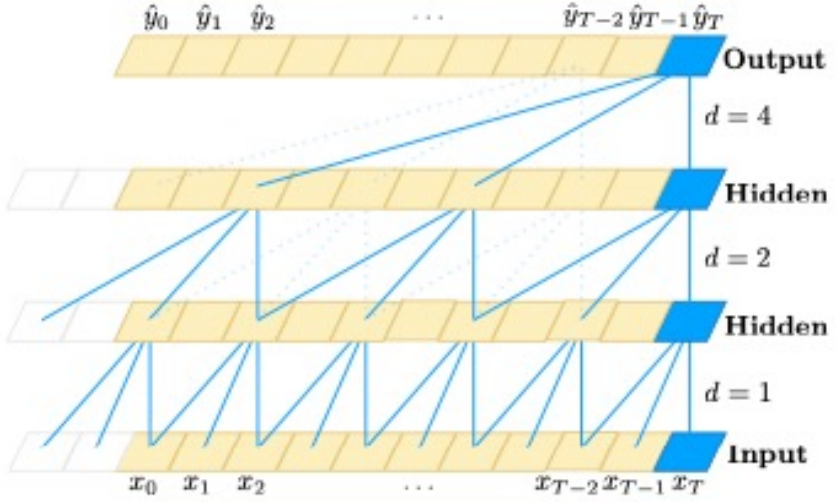
- A convolution kernel that is convolved with the input over a single temporal dimension
- Filters are trained to detect features in the sequence, regardless of where they appear
- Multiple filters can detect multiple features
- Can be implemented to multivariate time series (Input [$t \times \#features$])

Causality

- *causal constraint*: y_t depends only on $x_0,...,x_t$ and not on any "future" inputs $x_{t+1},...,x_T$.
- Learning: network f that minimizes some expected loss between the actual outputs and the predictions, $L(y_0,...,y_T,f(x_0,...,x_T))$
- This formalism encompasses settings
  - auto- regressive prediction: predict some signal given its past:
    - setting the target output to be simply the input shifted by one time step.
  - does not capture domains (machine translation, or sequence- to-sequence prediction)
    - the entire input sequence (including *"future" states*) can be used to predict each output

Causal Convolution

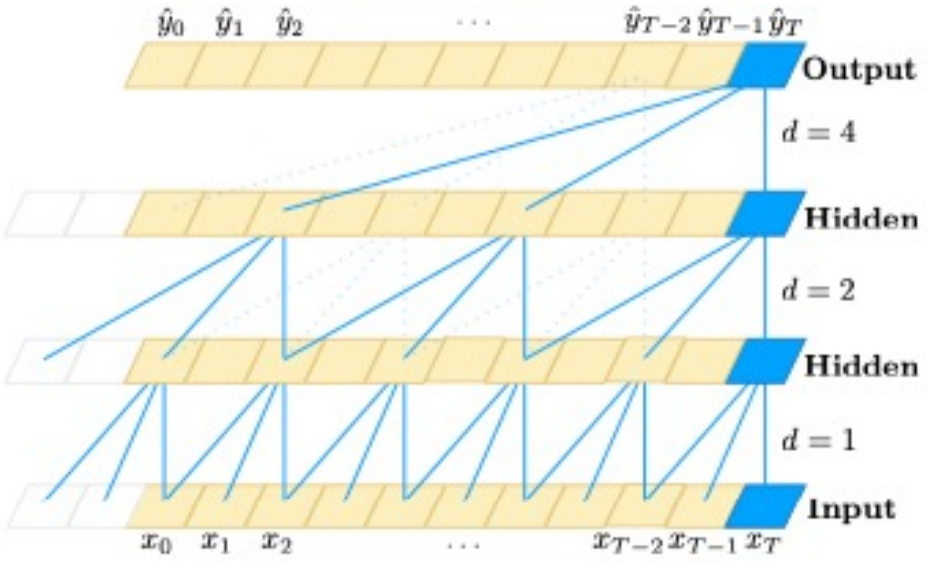
- Temporal Convolution Networks (TCN) constraints:
  - network produces an output of the same length as the input
    - To accomplish this: uses a 1D fully-convolutional network (FCN) architecture
    - each hidden layer is the same length as the input layer, and zero padding of length (kernel size − 1) is added to keep subsequent layers the same length as previous ones.
  - there can be no leakage from the future into the past.
    - TCN uses *causal convolutions:* output at time t is convolved **only** with elements <t in the previous layer.
  - *TCN = 1D FCN + causal convolutions*



An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

Causal Convolution

- Disadvantage: in order to achieve a long effective history size need large kernel size - extremely deep network or very large filters
- *Dilated convolution*: to allow for both very deep networks and very long effective history



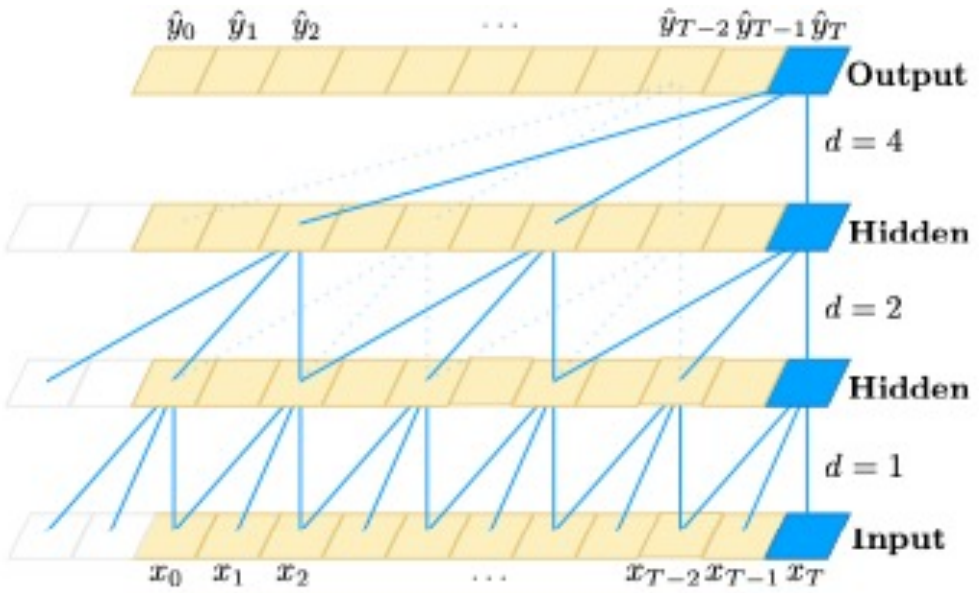
An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

Dilated Convolutions

- 1-D sequence input $x \in R^n$ and a filter $f:\{0,\ldots,k-1\} \rightarrow R$,
- Dilated convolution operation F on element s of the sequence:

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

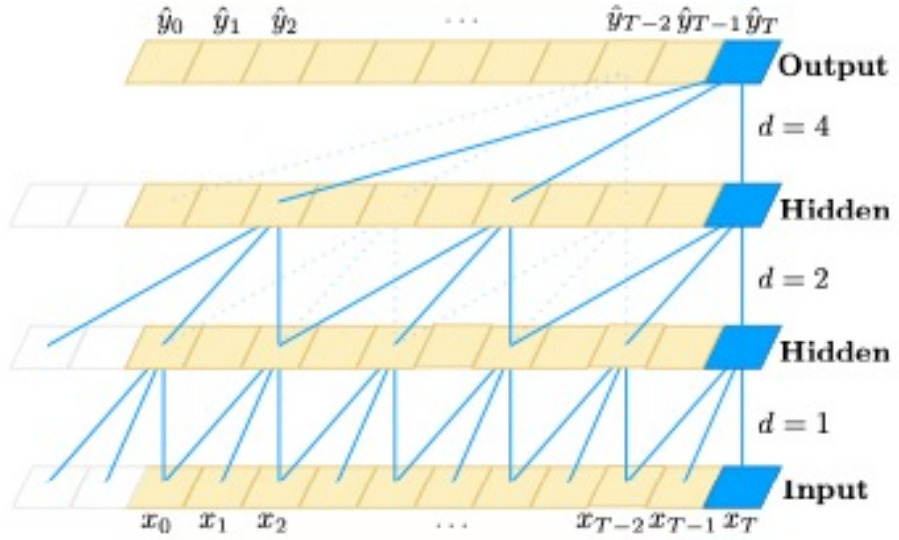
- *d*: dilation factor, k: filter size, $(s - d \cdot i)$ direction of the past.
- is equivalent to introducing a fixed step between every two adjacent filter taps.
- d = 1: regular convolution
- d>1: enables output at the top level to represent a wider range of inputs: expanding the receptive field of a ConvNet.



Dilated Convolutions

Longer memory - increase the receptive field of the TCN:
- choosing larger filter sizes *k*
- increasing the dilation factor *d*:
- effective history for this layer *(k − 1)d*.

In dilated convolutions, increase d exponentially with the depth of the network: $d = O(2^i)$. *i:* level in the network. Ensures
- each input treated by some filter in the effective history,
- allowing for an extremely large effective history using deep networks.



An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

Dilated Convolutions

Advantages TCNs for sequence modelling.

- *Parallelism*. Unlike in RNNs, convolutions can be done in parallel since the same filter is used in each layer.
- *Flexible receptive field size*: stacking more dilated (causal) convolutional layers, using larger dilation factors, increasing the filter size. better control of the model's memory size, and are easy to adapt to different domains.
- *Stable gradients*. Unlike recurrent architectures, TCN has a backpropagation path different from the temporal direction of the sequence. Thus avoids the problem of exploding/vanishing gradients
- *Low memory requirement for training*. For long input sequence, LSTMs and GRUs use a lot of memory for partial results for their multiple cell gates. TCN filters are shared across a layer, with the backpropagation path depending only on network depth.
- *Variable length inputs*. like RNNs, TCNs take in inputs of arbitrary lengths by sliding the 1D convolutional kernels.

Disadvantages

- *Data storage during evaluation*, RNNs only need hidden state h_t and the current input x_t in order to generate a prediction. In contrast, TCNs need the raw sequence up to the effective history length
- Potential parameter change for a transfer of domain.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

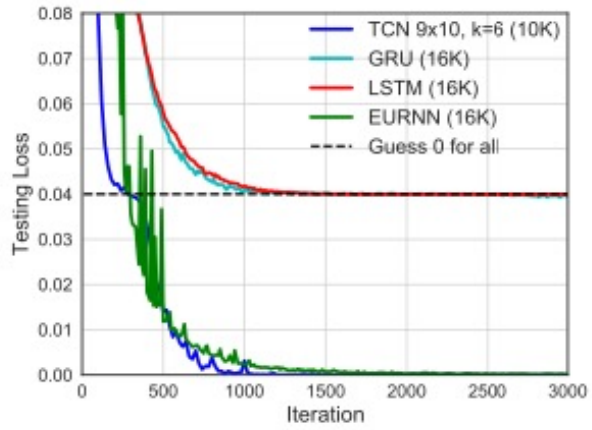
Experimental Evaluation

Sequence Modeling Task	Model Size (≈)	Models			
		LSTM	GRU	RNN	TCN
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

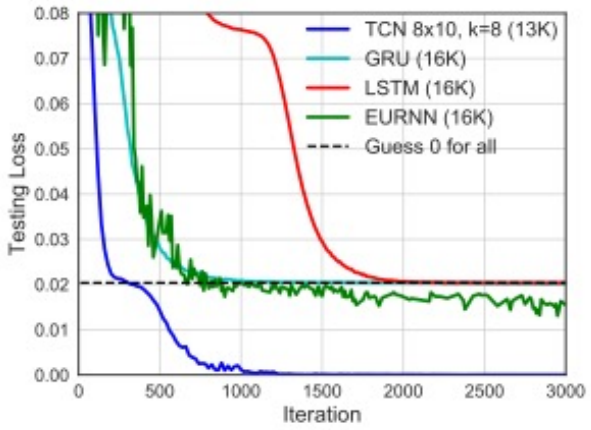
An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

Experimental Evaluation

- Copy memory task



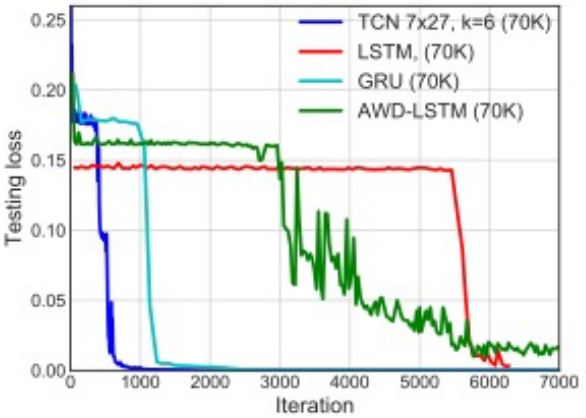
(a) $T = 500$



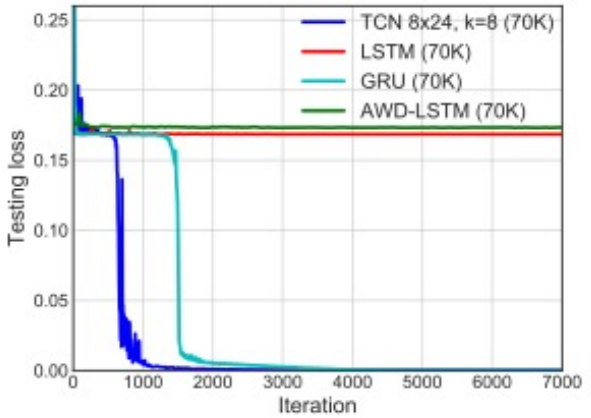
(b) $T = 1000$

- adding problem task

different sequence length



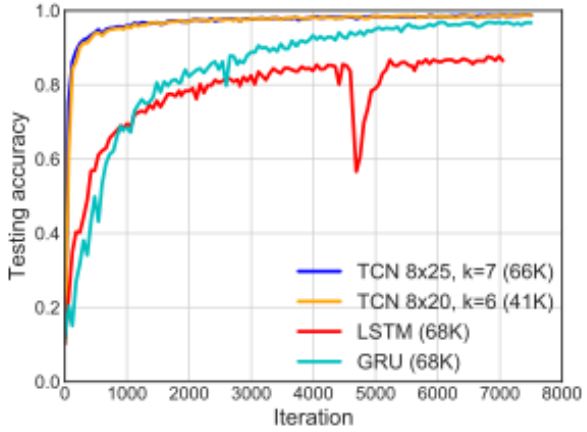
(a) $T = 200$



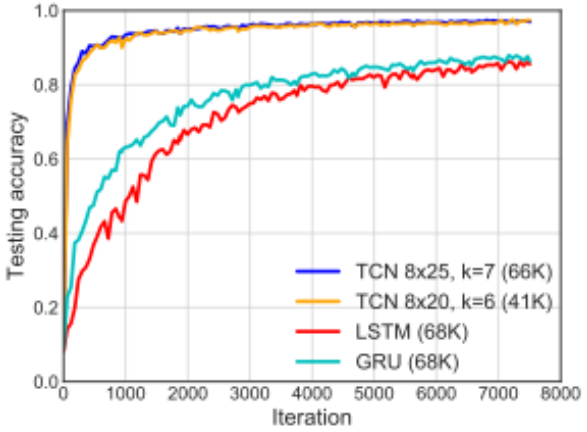
(b) $T = 600$

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

Experimental Evaluation

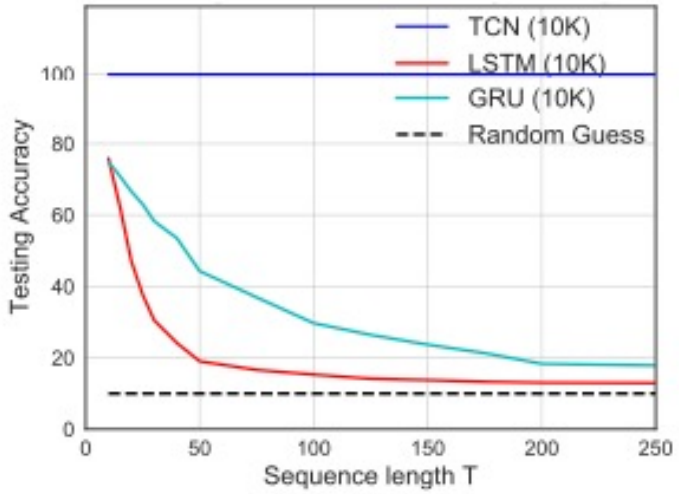


(a) Sequential MNIST



(b) P-MNIST

- Accuracy (copy memory task) for sequences of different lengths T



Transformer model for Forecasting [1]

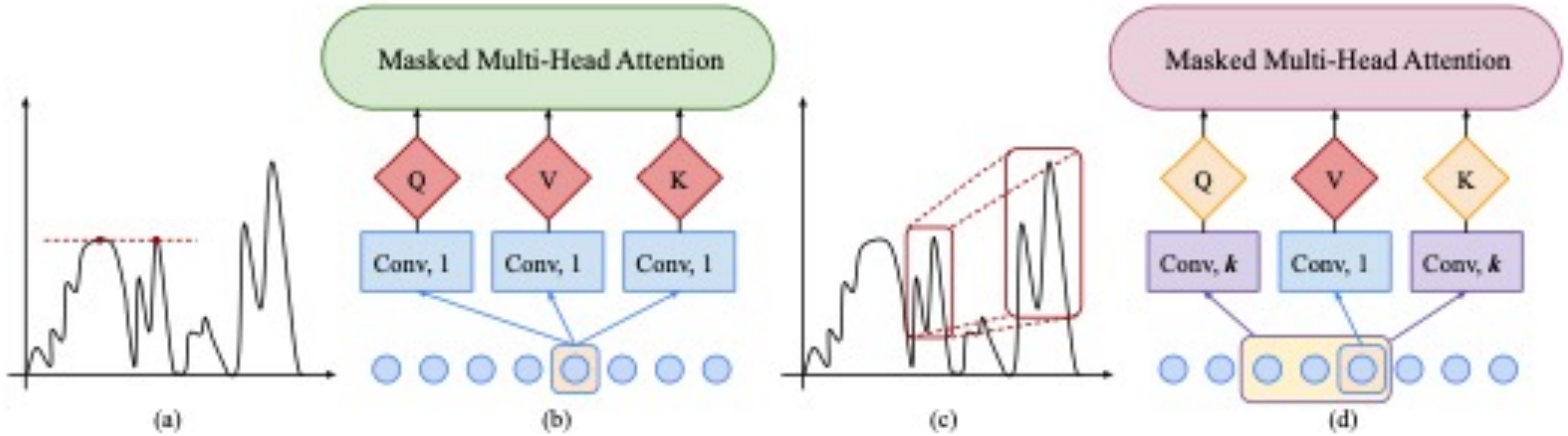
- Use of transformers in NLP impressive
- For time series and sequences two major weaknesses:
  - *locality-agnostics*: the dot- product self-attention in vanilla Transformer is insensitive to local context - prone to anomalies in time series;
  - *memory bottleneck*: space complexity of Transformer $\Theta(L^2)$ L: sequence length – thus modelling long time series infeasible.

[1] proposed

- convolutional self-attention by producing queries and keys with *causal convolution* so that local context can be better incorporated into attention mechanism.
- *LogSparse* Transformer with only $\Theta(L(\log L)^2)$ memory cost
- improving *forecasting accuracy* for time series with fine granularity
- strong long-term dependencies under constrained memory budget.
- experiments on synthetic data and real- world datasets show that it compares favorably to the state-of-the-art.

[1]"Enhancing the Locality and Breaking the Memory Bottleneck of Transformer onTime Series Forecasting, Li et al., 2019

Transformer model for Forecasting



vanilla vs. convolutional self-attention layers.

- “Conv, 1” and “Conv, k”: kernel size {1, k} with stride 1
- Vanilla self-attention in (b), may wrongly match point-wise inputs as in (a).
- Convolutional self-attention (d), convolutional layers of kernel size k with stride 1 to transform inputs (with proper paddings) into queries/keys.
- Thus locality awareness can correctly match the most relevant features based on shape matching in (c).

Transformer model for Forecasting

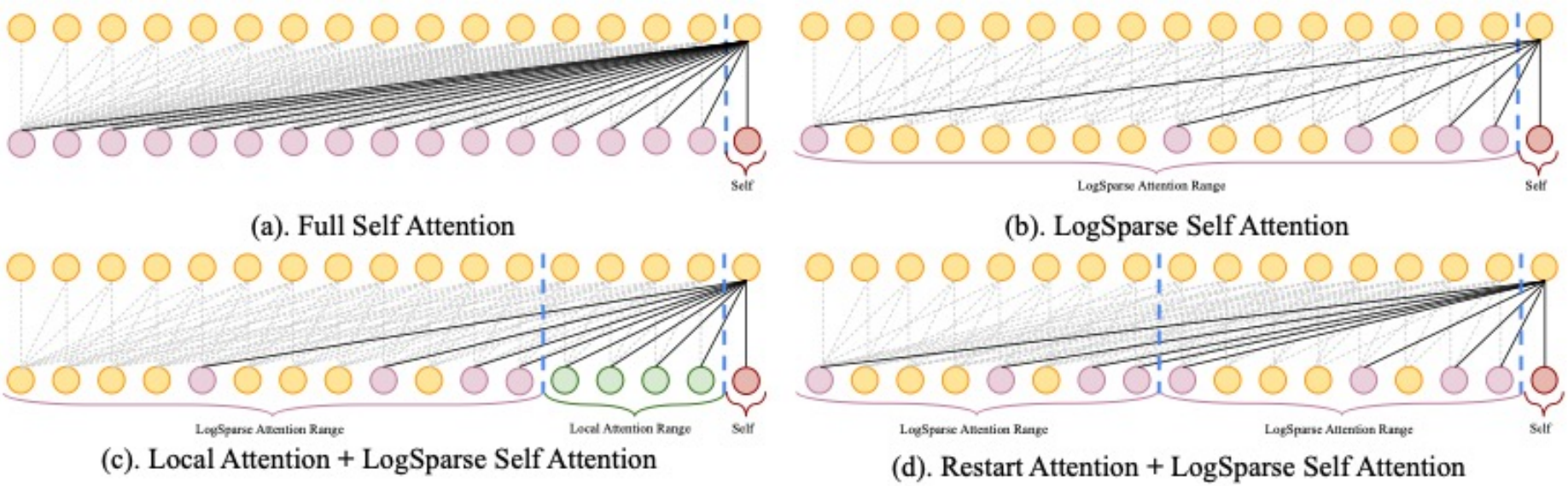


Illustration of different attention mechanism between adjacent layers in Transformer.

- Local Attention: allow each cell to densely attend to cells in its left window of size $O(\log_2 L)$ so that more local information/trend, can be leveraged for current step forecasting. Beyond the neighbor cells, can resume *LogSparse* attention strategy Figure (c).
- Restart Attention: divide the input length L into subsequences and set each subsequence length $L_{sub} \sim L$. For each of them, we apply the *LogSparse* attention strategy - Figure (d).

Experimental Evualuation

- Data Sets
  - electricity-f (fine): electricity consumption of 370 customers recorded every 15 minutes
  - electricity-c (coarse) dataset is the aggregated electricity-f by every 4 points, producing hourly electricity consumption.
  - traffic-f (fine): occupancy rates of 963 freeway in San Francisco recorded /20 minutes
  - traffic-c (coarse): hourly occupancy averaging every 3 points in traffic-f.
  - solar dataset6: solar power production records January to August 2006, sampled /hour from 137 PV plants in Alabama.
  - wind7 dataset contains daily estimates of 28 countries' energy potential (1986 – 2015) as % of a power plant's maximum output.
  - M4-Hourly: 414 hourly time series from M4 competition

Experimental Evualuation

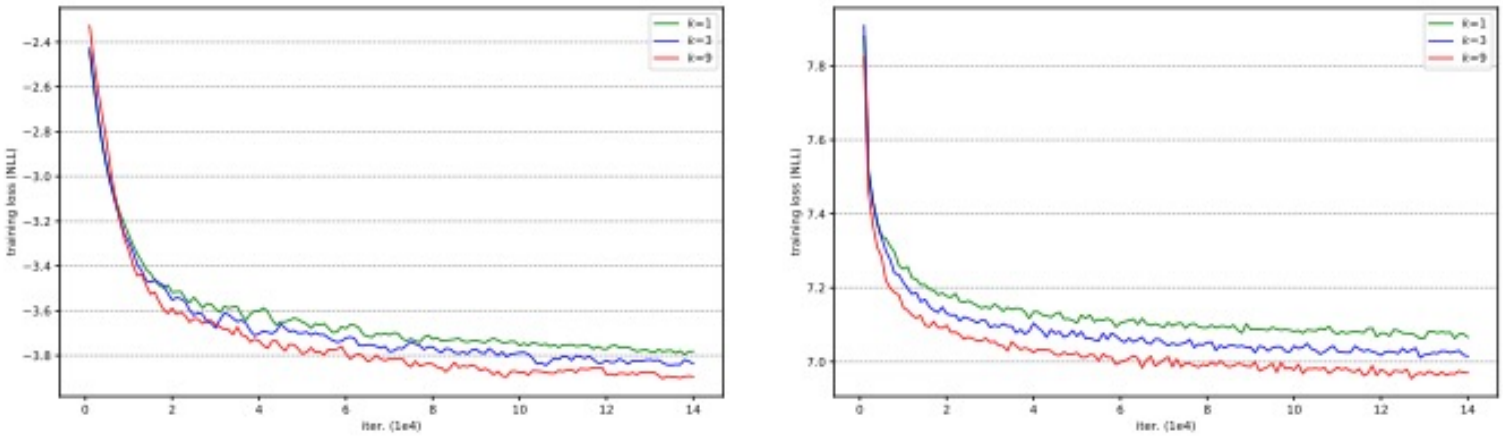
	ARIMA	ETS	TRMF	DeepAR	DeepState	Ours
e-c$_{1d}$	0.154/0.102	0.101/0.077	0.084/-	0.075$^\diamond$/0.040$^\diamond$	0.083$^\diamond$/0.056$^\diamond$	**0.059/0.034**
e-c$_{7d}$	0.283$^\diamond$/0.109$^\diamond$	0.121$^\diamond$/0.101$^\diamond$	0.087/-	0.082/0.053	0.085$^\diamond$/0.052$^\diamond$	**0.070/0.044**
t-c$_{1d}$	0.223/0.137	0.236/0.148	0.186/-	0.161$^\diamond$/0.099$^\diamond$	0.167$^\diamond$/0.113$^\diamond$	**0.122/0.081**
t-c$_{7d}$	0.492$^\diamond$/0.280$^\diamond$	0.509$^\diamond$/0.529$^\diamond$	0.202/-	0.179/0.105	0.168$^\diamond$/0.114$^\diamond$	**0.139/0.094**

- R0.5/R0.9-loss – from ρ-quantile loss:

$$R_\rho(\mathbf{x}, \hat{\mathbf{x}}) = \frac{2 \sum_{i,t} D_\rho(x_t^{(i)}, \hat{x}_t^{(i)})}{\sum_{i,t} |x_t^{(i)}|}, \quad D_\rho(x, \hat{x}) = (\rho - \mathbf{I}_{\{x \leq \hat{x}\}})(x - \hat{x})$$

- xˆ is the empirical ρ-quantile of the predictive distribution and I{x≤xˆ} is an indicator function, ρ ∈ (0, 1)

Experimental Evualuation



- Training curve comparison with kernel size k ∈ {1, 3, 9} traffic-c (left) and electricity-c (right) dataset.
- Being aware of larger local context size, the model can achieve lower training error and converge faster.

Table 2: Average $R_{0.5}/R_{0.9}$-loss of different kernel sizes for rolling-day prediction of 7 days.

	$k = 1$	$k = 2$	$k = 3$	$k = 6$	$k = 9$
electricity-c_{1d}	0.060/**0.030**	0.058/**0.030**	**0.057**/0.031	**0.057**/0.031	0.059/0.034
traffic-c_{1d}	0.134/0.089	0.124/0.085	0.123/0.083	0.123/0.083	**0.122/0.081**

Experimental Evualuation

sparse attention vs. full attention models

- Average R0.5/R0.9-loss comparisons, rolling-day prediction 7 days, kernel k = 6.
- with/without convolutional self-attention "Full" means models are trained with full attention while "Sparse" means they are trained with our sparse attention strategy.
- "+ Conv" means models are equipped with convolutional self-attention with

Constraint	Dataset	Full	Sparse	Full + Conv	Sparse + Conv
Memory	electricity-f_{1d}	0.083/0.051	0.084/**0.047**	**0.078**/0.048	0.079/0.049
	traffic-f_{1d}	0.161/0.109	0.150/0.098	0.149/0.102	**0.138/0.092**
Length	electricity-f_{1d}	0.082/0.047	0.084/0.047	**0.074/0.042**	0.079/0.049
	traffic-f_{1d}	0.147/0.096	0.150/0.098	0.139/**0.090**	**0.138**/0.092

- R0.5/R0.9-loss of datasets with various granularities.
- subscript of each dataset forecasting horizon (days).

	electricity-f_{1d}	traffic-f_{1d}	solar$_{1d}$	M4-Hourly$_{2d}$	wind$_{30d}$
TRMF	0.094/-	0.213/-	0.241/-	-/-	0.311/-
DeepAR	0.082/0.063	0.230/0.150	0.222/0.093	0.090°/0.030°	0.286/0.116
Ours	**0.074/0.042**	**0.139/0.090**	**0.210 /0.082**	**0.067 /0.025**	**0.284/0.108**

Refererences

- 25 years of time series forecasting, Jan G. De Gooijer, Rob J. Hyndman b, 2006
- Deep Learning for Time-Series Analysis, 2017: https://arxiv.org/abs/1701.01887
- Deep Learning for Time-Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Jason Brownlee, Machine Learning Mastery, 2018
- FORECASTING ECONOMIC AND FINANCIAL TIME SERIES: ARIMA VS. LSTM, SIMA SIAMI NAMIN, https://arxiv.org/pdf/1803.06386.pdf , 2018.
- Deep Learning for Time-Series Analysis, Gamboa J., 2017, https://arxiv.org/pdf/1701.01887.pdf
- G.Peter Zhang, B.Eddy Patuwo, Michael Y. Hu, "A simulation study of artificial neural networks for nonlinear time-series forecasting," Computers & Operations Research, Volume 28, Issue 4, 2001, pp. 381-396, ISSN 0305-0548, https://doi.org/10.1016/S0305-0548(99)00123-9.
- G.Peter Zhang, Min Qi, "Neural network forecasting for seasonal and trend time series," European Journal of Operational Research, Volume 160, Issue 2, 2005, pp. 501-514, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2003.08.037.
- https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/
- Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015
- Bryan Lim and Stefan Zohren, Time Series Forecasting With Deep Learning: A Survey, https://arxiv.org/abs/2004.13408

References

- MLP
  - Shiblee, M., Kalra, P. K., & Chandra, B. (2009). Time Series Prediction with Multilayer Perceptron (MLP): A New Generalized Error Based Approach. Lecture Notes in Computer Science, 37–44. doi:10.1007/978-3-642-03040-6_5
  - N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados and Yoshua Bengio, https://arxiv.org/pdf/1905.10437.pdf
- RNNs, LSTMs
  - K. A. Althelaya, E. M. El-Alfy and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, 2018, pp. 151-156, doi: 10.1109/IACS.2018.8355458.
  - DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks, David Salinas, Valentin Flunkert and Jan Gasthaus, https://arxiv.org/pdf/1704.04110.pdf
  - Deep State Space Models for Time Series Forecasting, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, https://papers.nips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdfdeep
- 1D CNN, Causal CNN, Dilated CNN
  - *Oord et al. WaveNet: A Generative Model for Raw Audio. https://arxiv.org/pdf/1609.03499.pdf
  - *S. Bai, J. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. https://arxiv.org/pdf/1803.01271.pdf
  - Chang et al. Dilated Recurrent Neural Networks.https://papers.nips.cc/paper/2017/file/32bb90e8976aab5298d5da10fe66f21d-Paper.pdf
- Hybrid Models (ARIMA-LSTM, LSTM-CNN, Attention-LSTM)
  - Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks, Wuokun et al, 2018, https://dl.acm.org/doi/pdf/10.1145/3209978.3210006
  - A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction, Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang and Garrison Cottrell, https://songdj.github.io/publication/ijcai-17-a/ijcai-17-a.pdf
  - Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting, Rajat Sen, Hsiang-Fu Yu and Inderjit Dhillon, https://papers.nips.cc/paper/2019/file/3a0844cee4fcf57de0c71e9ad3035478-Paper.pdf
  - Hybrid Neural Networks for Learning the Trend in Time Series, Lin et al. 2017, https://infoscience.epfl.ch/record/262447
  - Learning representations from eeg with deep recurrent-convolutional neural networks, Bashivan et al., arXiv preprint arXiv:1511.06448, 2015, https://arxiv.org/pdf/1511.06448.pdf
  - Wang, R., Peng, C., Gao, J. et al. A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series. Comp. Appl. Math. 39, 30 (2020). https://doi.org/10.1007/s40314-019-1006-2
  - G.Peter Zhang, "Time series forecasting using a hybrid ARIMA and neural network model", Neurocomputing, Volume 50, 2003, pp. 159-175, ISSN 0925-2312, https://doi.org/10.1016/S0925-2312(01)00702-0.
  - Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944
- Transformer for Time-series
  - Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting, Li et al., 2019, https://arxiv.org/pdf/1907.00235.pdf
- Graphs and GNNs
  - Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting, Yaguang Li, Rose Yu, Cyrus Shahabi and Yan Liu, https://arxiv.org/pdf/1707.01926.pdf
  - Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting, Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong and Qi Zhang, https://arxiv.org/pdf/2103.07719.pdf

- Time Series
- **Recommendations**
- Graph generation

AI for Recommendations

Recommendation widely applied online:

- E-commerce (Amazon)
- Content Sharing (Netflix, Youtube, Pinterest)
- Social Networking (Facebook, Linked-in Twitter
- Forums (Reddit, IMDB,
- Gaming (Tencent)
- ….

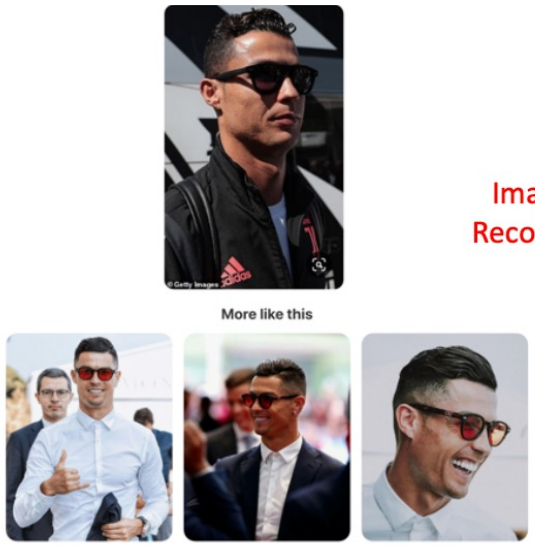
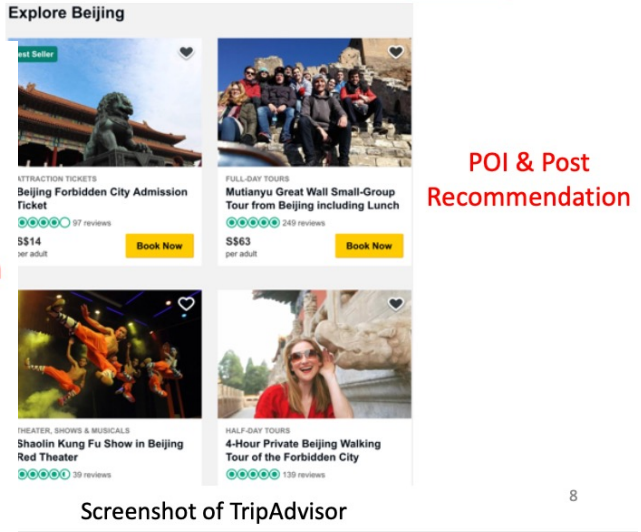
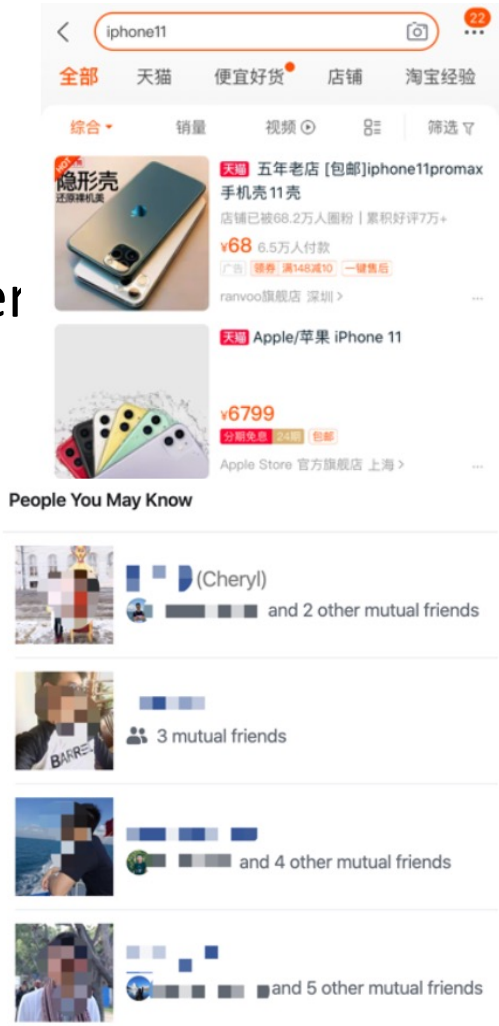


Image & Video Recommendation



Screenshot of TripAdvisor

POI & Post Recommendation



Ad & Product Recommendation

Friend Recommendation

Value of Recommendation Systems

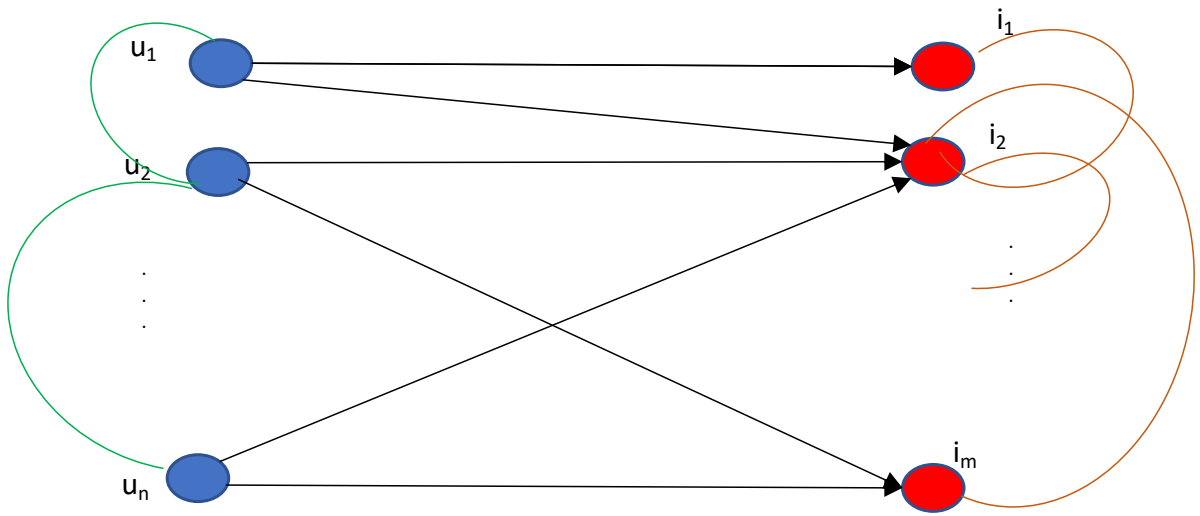
- Netflix: 80% of what people watch comes from some sort of recommendation.
  - 2015: *"We think the combined effect of personalization and recommendations save us more than $1B per year."*

  Netflix paper: https://dl.acm.org/doi/pdf/10.1145/2843948
- Amazon: 30%+ page views comes from recoms[Smith and Linden, 2017]
- YouTube Homepage: 60%+ clicks [Davidson et al. 2010]
- Airbnb, Search Ranking …: 99% of all booking conversions.

Early days

- Netflix: personalized video-recommendation system based on ratings and reviews by its customers.
- In 2006, offered a $1,000,000 prize to the first developer of a video-recommendation algorithm that could beat its existing algorithm - *Cinematch* by 10%.
- Collaborative filtering early Recommender Systems
  - Spotify
  - YouTube.
  - LinkedIn (Browsemaps).

Basic Info: Users – Items graphs



User-User
- Social Relations
- SameProfiles

...

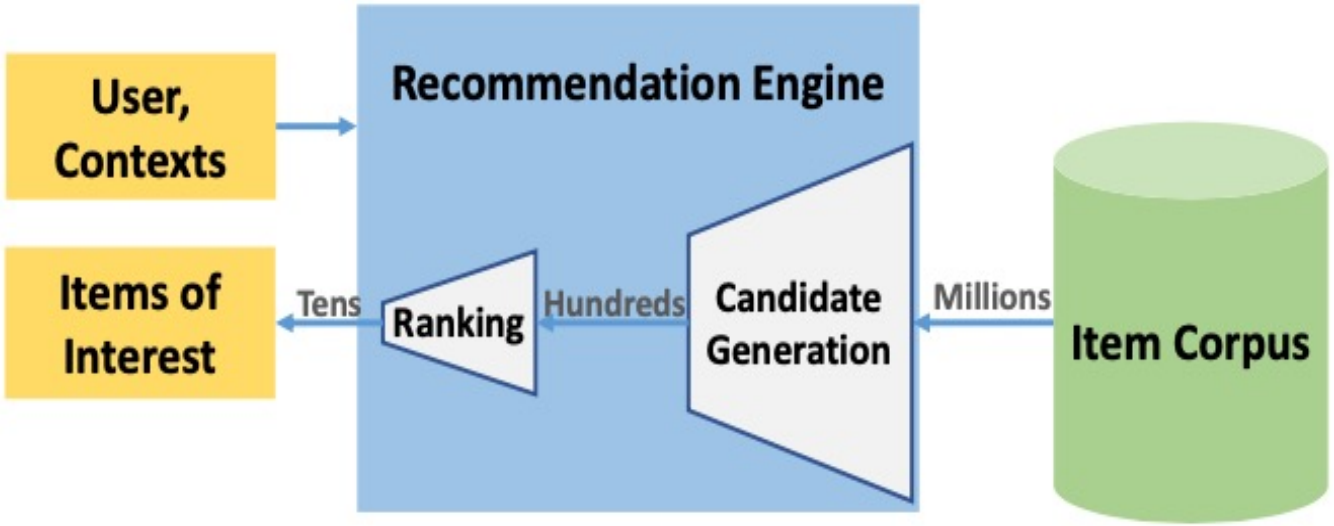
User-Item
- Implicit Feedback
- Explicit Feedback …

Item-Item
- Same Attributes
- External Knowledge ...

Recommendation Engine architecture

User Contexts

- Interaction history
- Demographics ...



Items

- Products, News, Movies, Videos, Friends ...

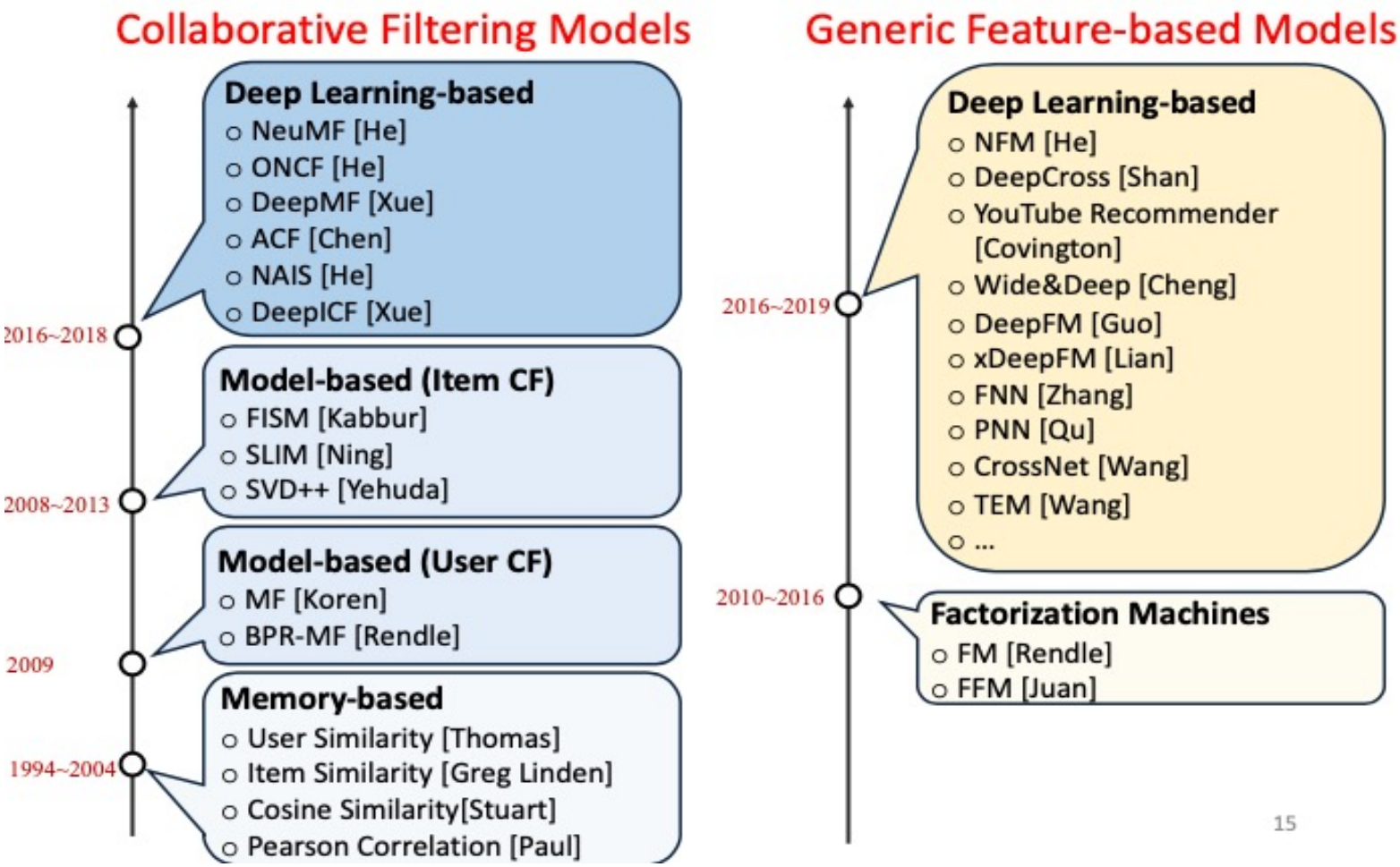
Recommender Data Model

- Set $U=\{u_1, \ldots, u_n\}$ of users
- Set $I=\{i_1, \ldots, i_m\}$ of items (e.g. products)
- Elements from U and I can be described by a vector respectively
  - $(a_1, \ldots, a_s)$ → attributes of user profile
  - $(b_1, \ldots, b_t)$ → description of items (meta data, features, …)
- Goal of recommendation process: recommend new items for an active user u
- Overview of process
  - User modeling (explicit or implicit, e.g. user rates items)
  - Personalization, generate list of recommended items

Issues of Recommender Systems

- Cold start and latency problems
- Sparseness of user-item matrix
- Diversity of recommendations
- Scalability
- Privacy and trust
- Robustness
- Utilization of domain knowledge
- Changing user interests (dynamics)
- Evaluation of recommender systems

Recommendations methods landscape



Learning and Reasoning on Graph for Recommendation, Xiang Wang , Xiangnan He, Tat-Seng Chua , tutorial CIKM 2019

Recommendations

Introduction

Methods - Background
- Collaborative filtering
- Matrix completion NMF

Advanced Methods
- Factorization machines –
- Deep Learning

Collaborative Filtering (CF)

- Basic idea: System recommends items which were preferred by similar users in the past
  - Based on ratings
    - Expressed preferences of the active user
    - And also other users → Collaborative approach
  - Works on user-item matrix
    - Memory-based or model-based
    - No item meta data etc.!
- Assumption: Similar taste in the past implies similar taste in future
- CF is formalization of “word of mouth“ among buddies

General Process

1. Users rate items
2. Find set S of users which have rated similar to the active user u in the past (→ neighborhood)
   - Similarity calculation
   - Select the k nearest users to the active user
3. Generate candidate items for recommendation
   - Items which were rated in neighborhood of u,
   - but were not rated by u yet
4. Predict rating of u for candidate items
   - Select and display n best items

Example (I)

	Hoop Dreams	Star Wars	Pretty Woman	Titanic	Blimp	Rocky XV
Joe	D	A	B	D	?	?
John	A	F	D		F	
Susan	A	A	A	A	A	A
Pat	D	A		C		
Jean	A	C	A	C		A
Ben	F	A				F
Nathan	D		A		A	

Source: http://www.dfki.de/~jameson/ijcai03-tutorial/

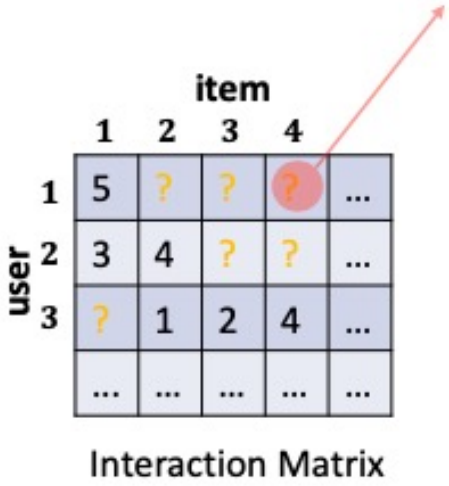
Example (II)

	Hoop Dreams	Star Wars	Pretty Woman	Titanic	Blimp	Rocky XV
Joe	D	A	B	D	?	?
John	A	F	D		F	
Susan	A	A	A	A	A	A
Pat	D	A		C		
Jean	A	C	A	C		A
Ben	F	A				F
Nathan	D		A		A	

Example (III)

	Hoop Dreams	Star Wars	Pretty Woman	Titanic	Blimp	Rocky XV
Joe	D	A	B	D	?	?
John	A	F	D		F	
Susan	A	A	A	A	A	A
Pat	D	A		C		
Jean	A	C	A	C		A
Ben	F	A				F
Nathan	D		A		A	

Memory based CF



Interaction Matrix

Problem: predict user u's rating on item i.

- User-based CF computes ratings of u's *similar users* on the target item i.

$$\hat{y}_{ui} = \sum_{u' \in S_u(u)} sim(u, u') \cdot y_{u'i}$$

Rating of a similar user on i

Similar users of u

- Item-based CF leverages the ratings of u on other *similar items* of i

$$\hat{y}_{ui} = \sum_{i' \in S_i(i)} sim(i, i') \cdot y_{ui'}$$

Rating of u on a similar item

Similar items of i

- Many similarity measures can be used, e.g., Jaccard, Cosine, Pearson Correlation. Recent advance learns the similarity from data.

Recommendations

Introduction

Methods - Background
- Collaborative filtering
- Matrix completion NMF

Advanced Methods
- Factorization machines –
- Deep Learning

Explaining data by factorization

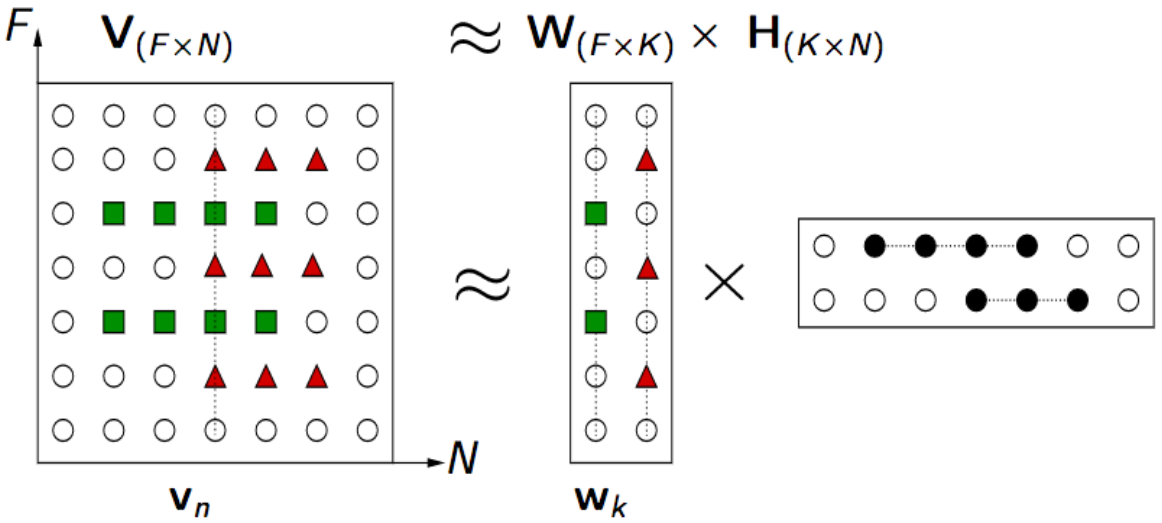


Illustration by C. Févotte

Low Rank approximation

- Consider the rating matrix R with elements R_{ui}
  If we have a set of users U and a set of items I,
- Rating Matrix is **sparse**
- then R is a $|U| \times |I|$ matrix
- We can approximate this by a low-rank approximation:
- $R \approx AB$
- where A is a $|U| \times K$ matrix and B is a $K \times |I|$ matrix
- K is the rank of AB (number of non-zero eigenvalues)

Low Rank approximation

- ratings R_{ui} for $(u,i) \in D$ where
  $|D| \ll |U| \times |I|$ (*number of ratings is typically much less than the total possible number of ratings*)
- In low rank approximation we are trying to fit to $|D|$ ratings of the matrix with $(|U| + |I|) \times K$ elements
- K controls the complexity of the model
- Large K will fit the data better but at the cost of possibly over-fitting

Non Negative Matrix factorization (NMF)

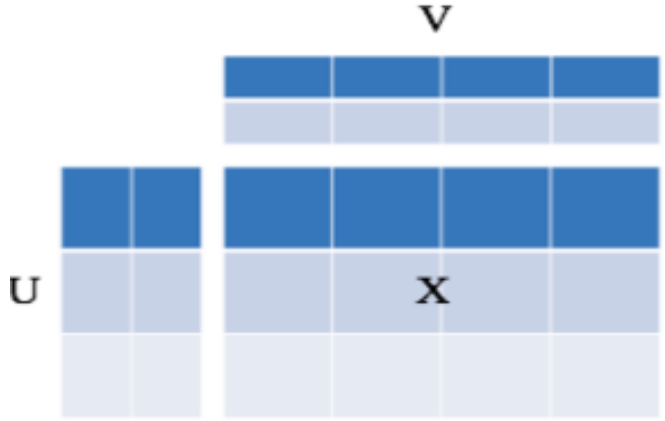
- Data is often nonnegative by nature
  - pixel intensities; occurrence counts; food or energy consumption; user scores; stock market values;
- Interpretability of the results, optimal processing of nonnegative data may call for processing under Non-negativity constraints

- Applying SVD results in factorized matrices with positive and negative elements may contradict the physical meaning of the result.
- - *Nonnegative matrix factorization (NMF)*
- find the reduced rank *nonnegative factors* to approximate a given nonnegative data matrix.

Non Negative Matrix factorization (NMF)

- Assume X (mxn) data matrix and $r << m, n$
- NMF aims to find non negative matrices

$$U \in R^{mxr}, V \in R^{rxn} : X \approx UV^T$$



$X \simeq UV^T$

- $U = [u_{fk}]\,, w_{fk} >= 0$
- $V = [v_{kn}]\,, h_{kn} >= 0$
- $k << f, n$

- To find U, V, optimization problem:

$$\min_{(U,V)} \left\| X - UV^T \right\|_2$$

- Alternative error function:

$$\min_{U,V} f(U,V) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(X_{ij} \log \frac{X_{ij}}{(UV^\top)_{ij}} - X_{ij} + \left(UV^\top\right)_{ij} \right)$$
$$\text{s.t. } U_{ia} \geq 0, V_{jb} \geq 0, \forall i, a, b, j$$

NMF - Alternating Least squares

1. Suppose we know U, *with* V unknown.

- for each j we could minimize $||X_{.j} - UV_{.j}^T||_2$
  - find $V_{.j}$ that minimizes with $X_{.j}$ and U known.
  - Frobenius norm: sum of squares,
  - minimization is a least-squares problem, i.e. linear regression
  - "predicting" $X.j$ from W.

$$V_{.j} = (U^T U)^{-1} U^T X.j$$

  - repeat for all columns $V_{.j}$

2. assume V, *with* U unknown: $X^T = VU^T$

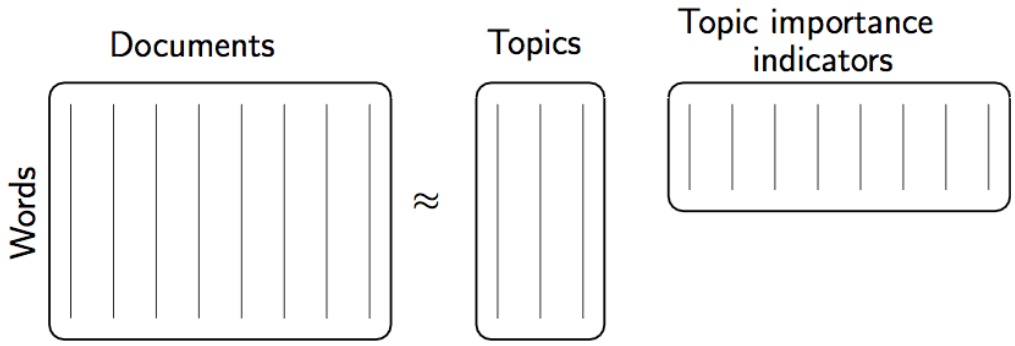
- Interchange roles of U, V in the above optimization
- Compute a row of U, repeat for all rows

NMF - Alternating Least squares

- Putting all this together
- first choose initial guesses, random numbers, for U and V
- alternate:
  - Compute U assuming V known
  - Compute V based on that new U
  - …
- may generate some negative values: simply truncate to 0

NMF issues, applications

- Uniqueness and Convergence
- U_{mxr} , r (rank) choice: via SVD…
- Applications
  - Topic detection
  - Source separation (music , speech)
  - Clustering
  - **Recommendations**



Recommendations

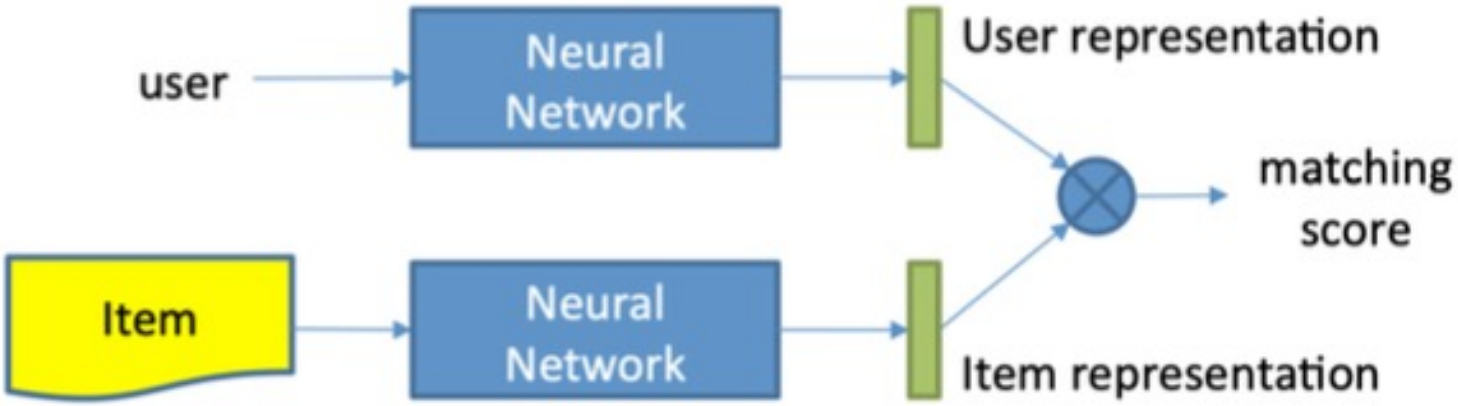
Introduction

Methods - Background
- Collaborative filtering
- Matrix completion NMF

Advanced Methods
- Deep Learning

Deep Learning for CF

- deep learning to embedd users, item and their interaction in an embedding space

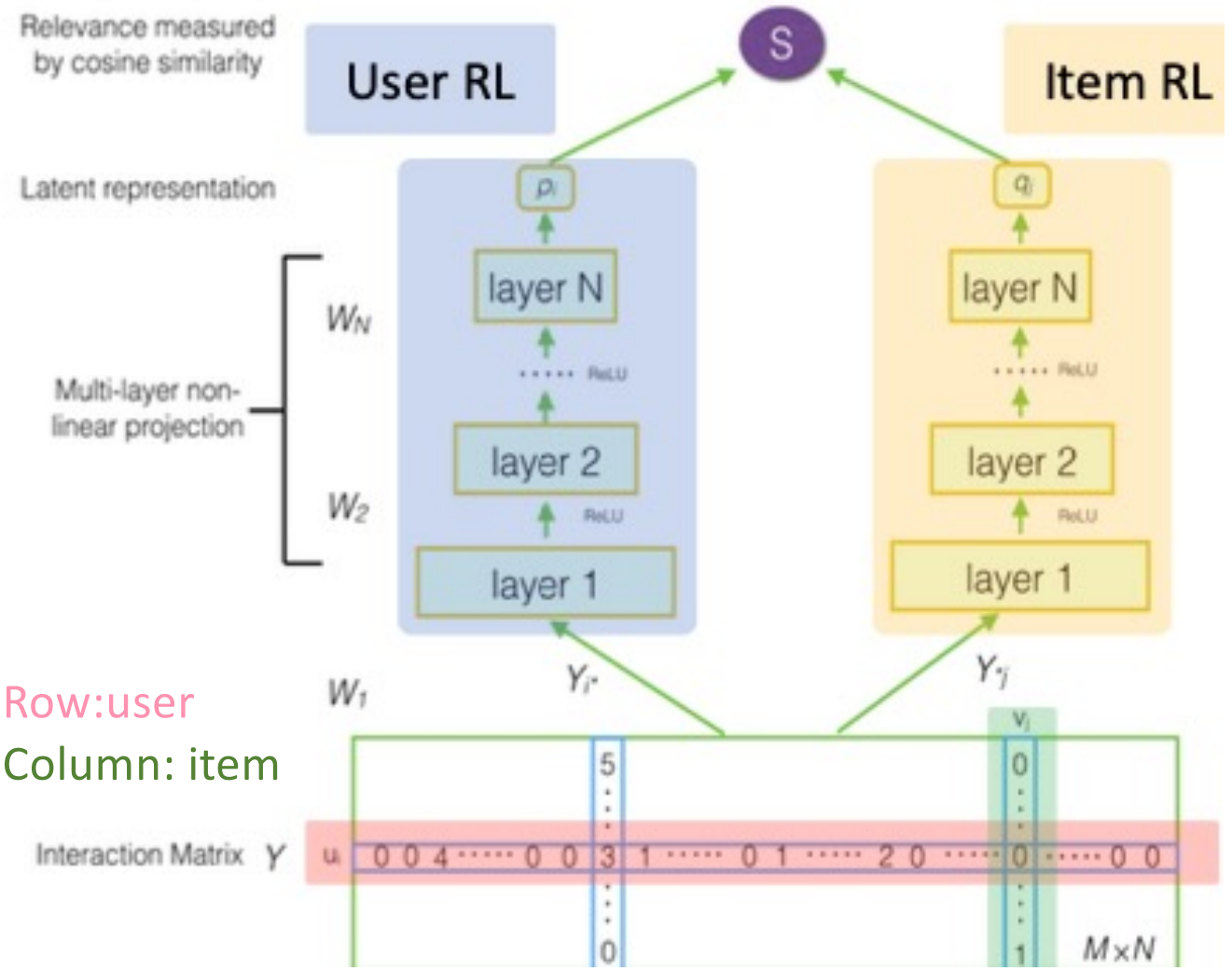


Model	Input Data	Representation Learning	Interaction Learning
DeepMF [Xue, IJCAI'17]	User: Historical items Item: User group	Multi-Layer Perceptron	Inner product
AutoRec [Sedhain, WWW'15]	User: Historical items Item: ID	Multi-Layer Perceptron	Inner Product
CDAE [Wu, WSDM'16]	User: Historical items + ID Item: ID	Multi-Layer Perceptron	Inner Product

23

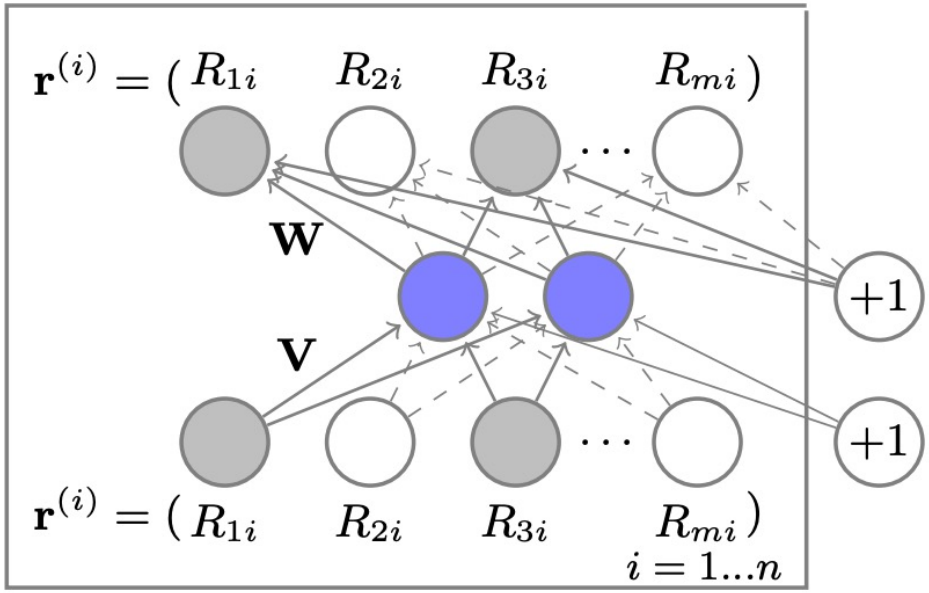
Deep Matrix Factorization (Xue, IJCAI'17)

- Representation Learning Multi-layer perceptron
  - Deep Neural Networks are adopted to learn representations of
- users & items



AutoRec: Autoencoders Meet Collaborative Filtering

- m users, n items,
- $R \in R^{m \times n}$: *partially observed user-item rating* matrix .
- user $u \in U = \{1 \ldots m\}$: $r(u) = (R^{ux1}, \ldots R^{un}) \in R^n$.
- item $i \in I = \{1 \ldots n\}$ $r(i) = (R^{1i}, \ldots R^{mi}) \in R^m$.
- item-based (user-based) autoencoder
- input each partially observed $r(i)$ $(r(u))$,
- project it into a low-dimensional latent (hidden) space
- reconstruct$)$ $r(i)$ $(r(u))$ in the output space to **predict missing ratings** for purposes of recommendation



$$\min_{\theta} \sum_{\mathbf{r} \in \mathbf{S}} ||\mathbf{r} - h(\mathbf{r}; \theta)||_2^2,$$

$h(r; \theta)$ is the reconstruction of input $r \in R^d$

$$h(\mathbf{r}; \theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

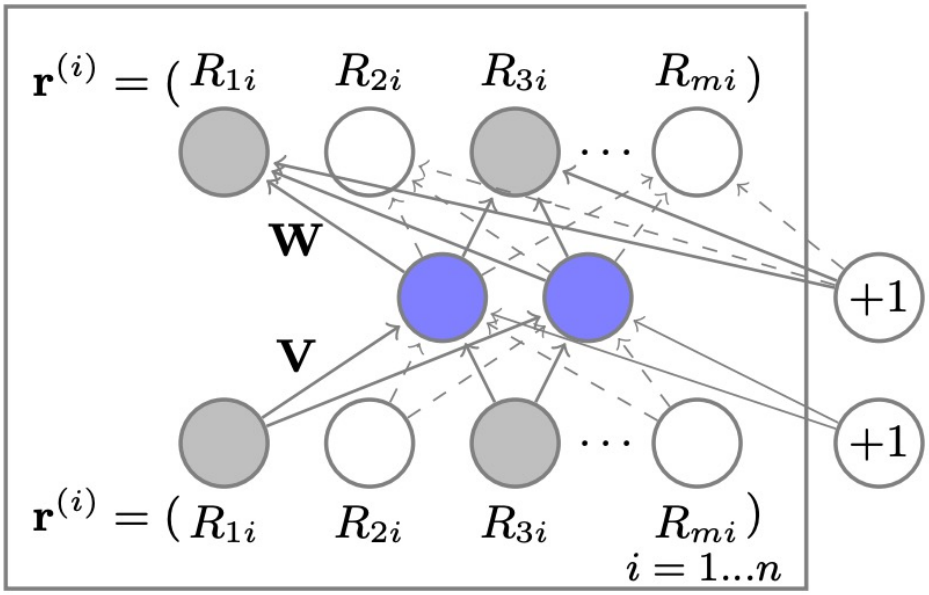
Auto rec

- the objective function for the Item-based AutoRec (I-AutoRec) model is, for regularisation λ > 0

$$\min_{\theta} \sum_{i=1}^{n} ||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta))||_{\mathcal{O}}^2 + \frac{\lambda}{2} \cdot (||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2)$$

- predicted rating for user u and item i

$$\hat{R}_{ui} = (h(\mathbf{r}^{(i)}; \hat{\theta}))_u$$



Auto rec - Experiments

	ML-1M	ML-10M
U-RBM	0.881	0.823
I-RBM	0.854	0.825
U-AutoRec	0.874	0.867
I-AutoRec	**0.831**	**0.782**

- RMSE of I/U-AutoRec and RBM models.

$f(\cdot)$	$g(\cdot)$	RMSE
Identity	Identity	0.872
Sigmoid	Identity	0.852
Identity	Sigmoid	**0.831**
Sigmoid	Sigmoid	0.836

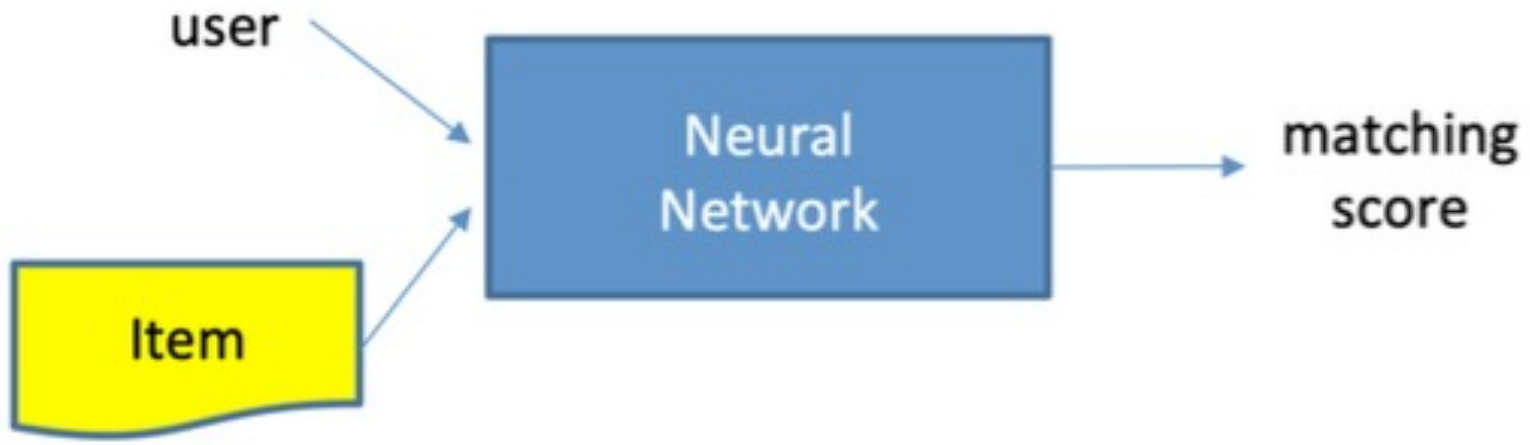
RMSE for I-AutoRec different activation functions, Movielens 1M dataset.

	ML-1M	ML-10M	Netflix
BiasedMF	0.845	0.803	0.844
I-RBM	0.854	0.825	-
U-RBM	0.881	0.823	0.845
LLORMA	0.833	**0.782**	0.834
I-AutoRec	**0.831**	**0.782**	**0.823**

- I-AutoRec vs baselines on MovieLens and Netflix datasets.

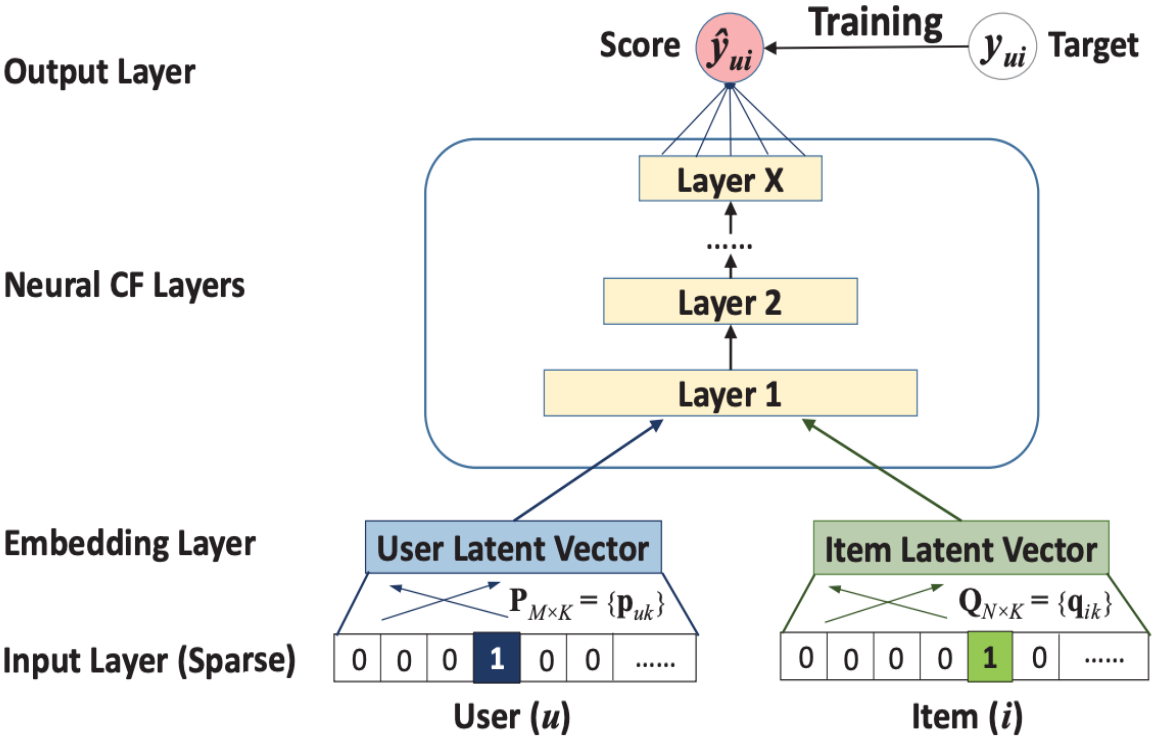
Deep Learning for CF II

- **Methods of interaction function learning**
  - Capture complex patterns of user-item relationships



Model	Input Data	Representation Learning	Interaction Learning
NCF [He, WWW'17]	User: ID Item: ID	ID embedding	Multi-Layer Perceptron
NNCF [Bai, CIKM'17]	User: User neighbors Item: Item neighbors	Embeddings	Multi-Layer Perceptron
ONCF [He, IJCAI'28]	User: ID Item: ID	ID embedding	Convolutional Neural Network

Neural Matrix Factorization (He, WWW'17)



Interaction Modelling

- MF + MLP over users and items
- MF uses inner product to capture the low-rank relation
- DNN to learn the matching function

Recommendation evaluation metrics

- top-K Hit Ratio (HR@K):

measures whether the test item appears on Top-K list,

- Normalized Discounted Cumulative Gain (NDCG)

Replaces hit with allocating higher scores to hits with top ranks

HR@K: a recall-based measure, as,

$$HR@K = \frac{\#hits@K}{|GT|}$$

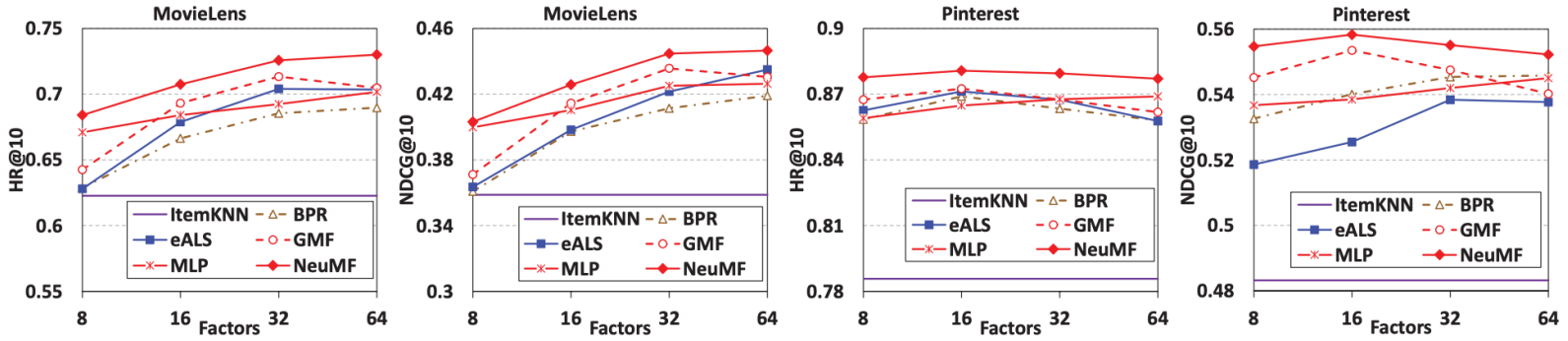
NDCG: is a ranking-based measure, as,

$$NDCG@K = Z_k \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

- GT: the test list set,
- rel_i : graded relevance value of the item at position i. $rel_i \in \{0, 1\}$, which depends on whether i is in the test dataset.
- Z_K": normalization.

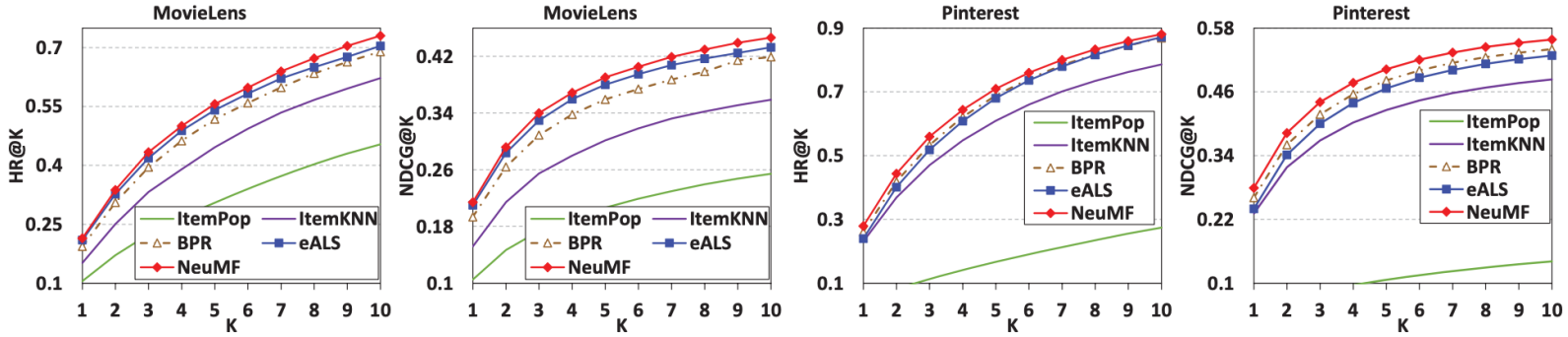
NeurMF - Experiments

Dataset	Interaction#	Item#	User#	Sparsity
MovieLens	1,000,209	3,706	6,040	95.53%
Pinterest	1,500,809	9,916	55,187	99.73%



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

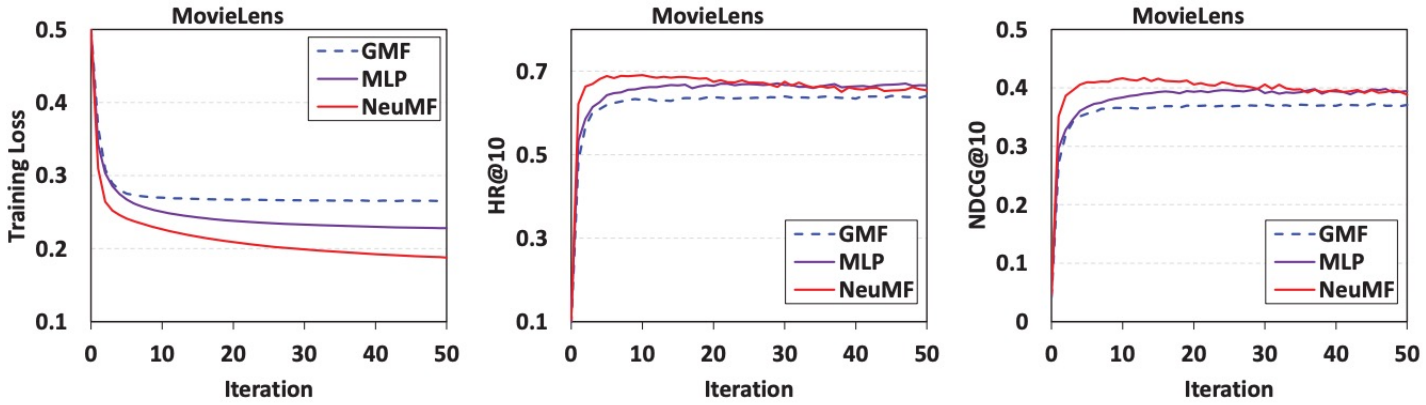
Figure 4: Performance of HR@10 and NDCG@10 *w.r.t.* the number of predictive factors on the two datasets.



(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

Figure 5: Evaluation of Top-K item recommendation where K ranges from 1 to 10 on the two datasets.

NeurMF - Experiments



(a) Training Loss (b) HR@10 (c) NDCG@10

Figure 6: Training loss and recommendation performance of NCF methods *w.r.t.* the number of iterations on MovieLens (factors=8).

DL helps…

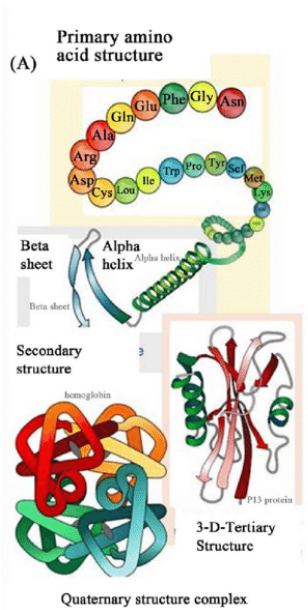
Table 3: HR@10 of MLP with different layers.

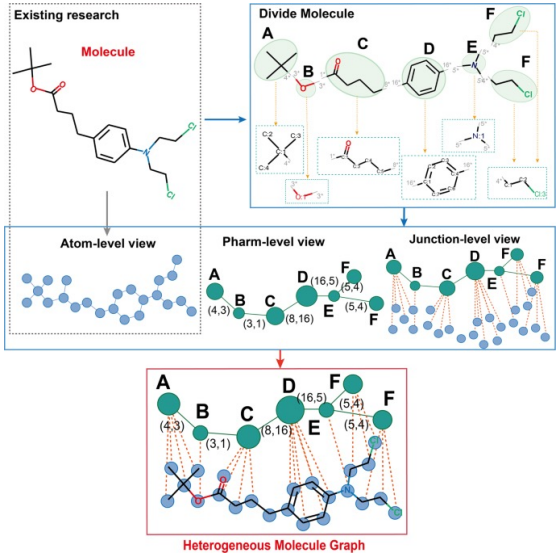
Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.452	0.628	0.655	0.671	**0.678**
16	0.454	0.663	0.674	0.684	**0.690**
32	0.453	0.682	0.687	0.692	**0.699**
64	0.453	0.687	0.696	0.702	**0.707**
Pinterest					
8	0.275	0.848	0.855	0.859	**0.862**
16	0.274	0.855	0.861	0.865	**0.867**
32	0.273	0.861	0.863	**0.868**	0.867
64	0.274	0.864	0.867	0.869	**0.873**

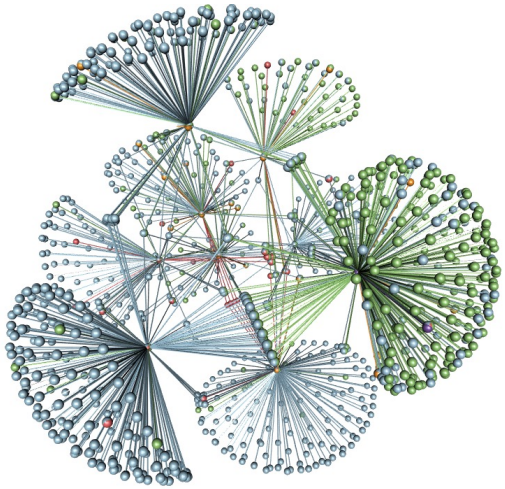
- Time Series
- Recommendations
- **Graph generation**

Graphs are ubiquitous

- Chemistry – Bio/Pharma
  - Space of molecules: 10^{60}
  - New proteins, molecules generation







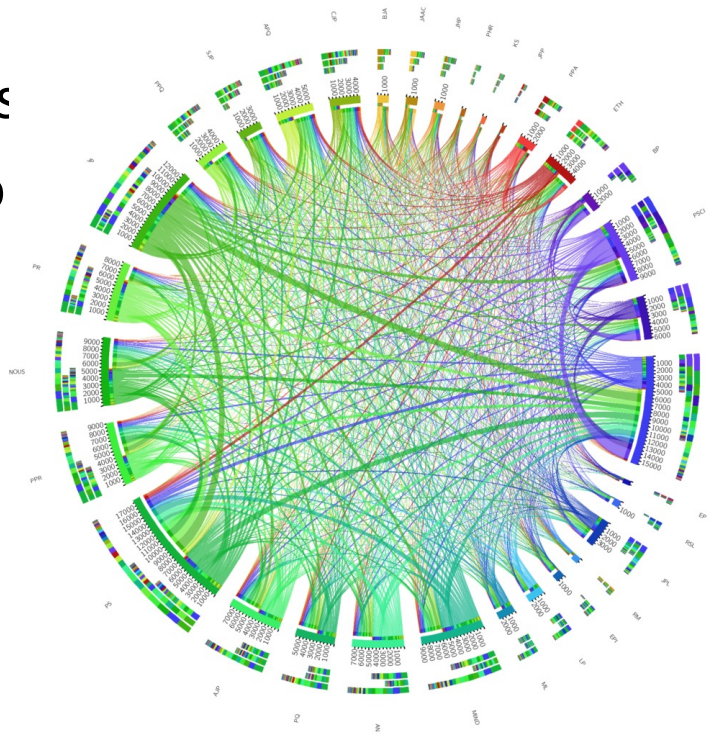
Guided Folding of Life's Proteins in Integrate Cells with Holographic Memory and GM-Biophysical Steering, Dirk K F Meijer, Hans J. H. Geesink, 2018, Open Journal of Biophysics 8(03):117-154 DOI:10.4236/ojbiphy.2018.83010,

Jiang, Y., Jin, S., Jin, X. *et al.* **Pharmacophoric-constrained heterogeneous graph transformer model for molecular property prediction**. *Commun Chem* **6**, 60 (2023). https://doi.org/10.1038/s42004-023-00857-x

Multiplex Human HIV-1 protein-protein interaction network https://commons.wikimedia.org/wiki/File:Multiplex_Human_HIV-1_protein-protein_interaction_network_%28edge-colored_visualization%29.png

Graphs are ub

- Social Networks
- Internet/Teleco
- Citation graphs

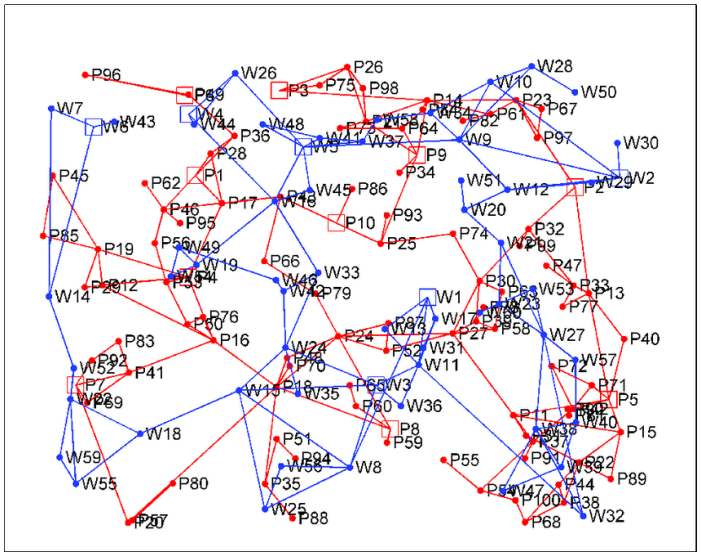


https://threatpost.com/researchers-graph-social-networks-spot-spammers-061711/75346/

http://tar.weatherson.org/2017/05/04/citation-graphs-and-methodology/

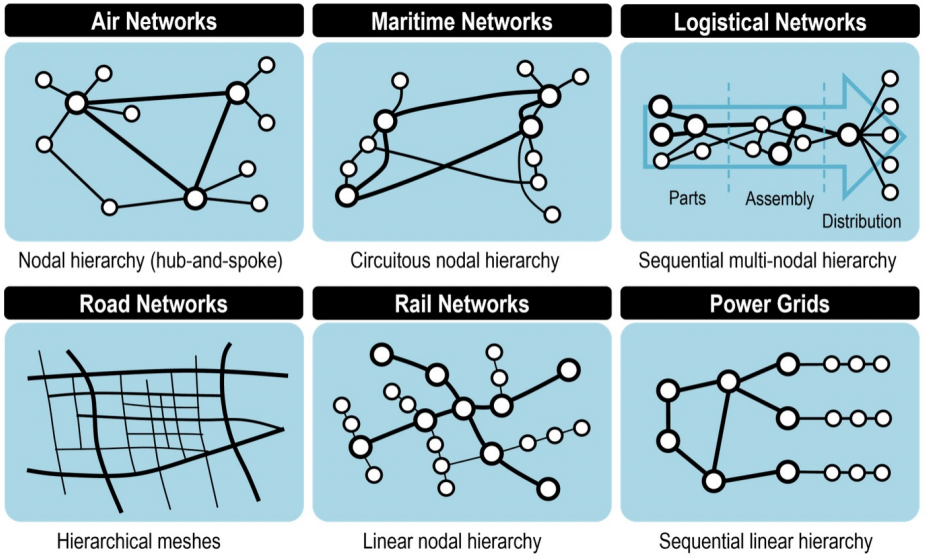
Graphs are ubiquitous

power/water distribution networks



doi.org/10.4324/9780429346323

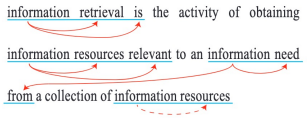
Transport/road networks



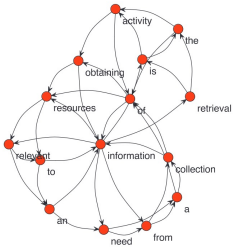
https://doi.org/10.1371/journal.pone.0195727.g005

Graphs in NLP

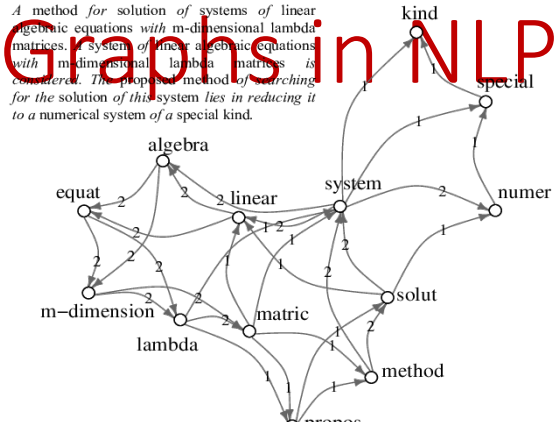
- Graph of Words
  - Information retrieval[1]
  - Keyword extraction [2]
  - Event detection [4]
  - Summarization
  - Document classification [3][1]

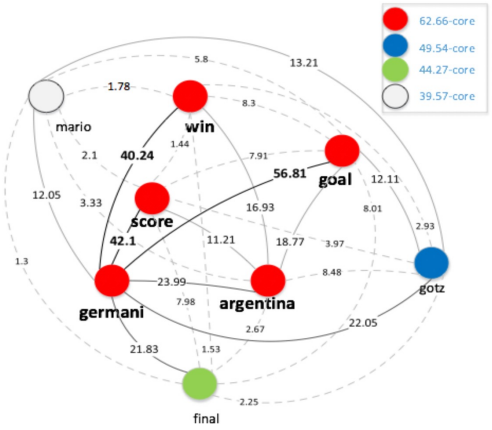


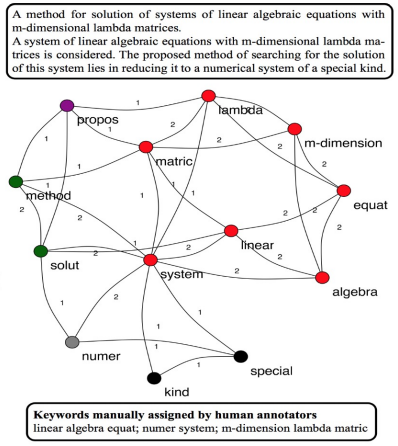
Bag of words: ((activity,1), (collection,1) (information,4), (relevant,1), (resources, 2), (retrieval, 1)..)

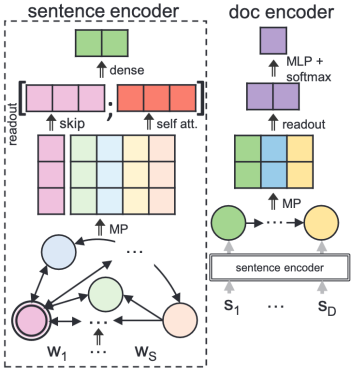


Captures: frequency, order and distance, …









[1] Graph-of-word and TW-IDF: new approach to ad hoc IR, F.Rousseau, Michalis Vazirgiannis - **CIKM '13**: https://doi.org/10.1145/2505515.2505671, Best paper mention award

[2]Main Core Retention on Graph-of-words for Single-Document Keyword Extraction, F. Rousseau,M. Vazirgiannis. **ECIR2015**

[3]Text Categorization as a Graph Classification Problem, F Rousseau, E Kiagias, M Vazirgiannis, **ACL 2015**

[4] Degeneracy-based real-time sub-event detection in twitter stream, P Meladianos, et. al. **AAAI - ICWSM 2015**

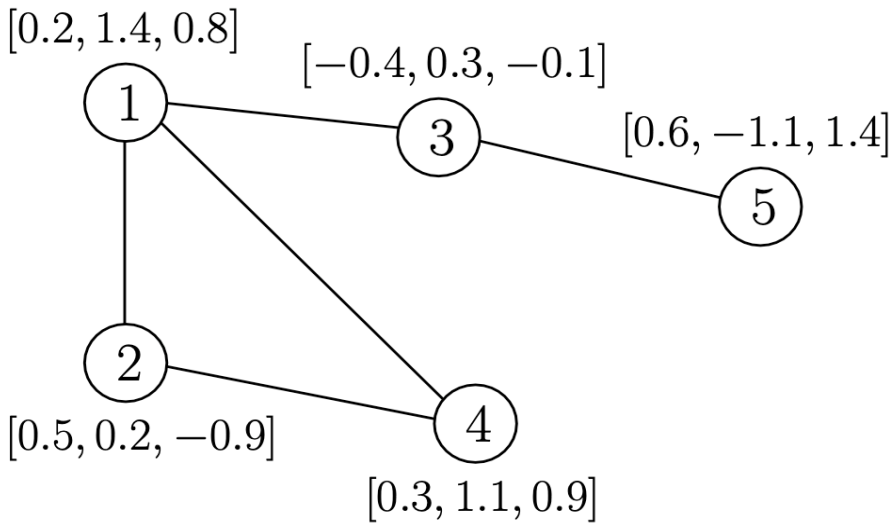
[5]Message Passing Attention Networks for Document Understanding, G. Nikolentzos, A. Tixier, M.Vazirgiannis , **AAAI2020**, https://doi.org/10.1609/aaai.v34i05.6376

Why Graph ML is important & different than sequential ML

- handles complex and rich data structures (graphs, networks, trees, and hypergraphs) not easily represented by vectors or matrices
- capture relational information and dependencies among nodes, and capitalise on the graph structure and properties to enhance the learning process – capture *longer term dependencies*
- can *leverage GNNs to learn powerful/expressive graph representations* useful for downstream tasks: node classification, link prediction, graph generation, and graph matching.
- Graph pretrained models tend to have fewer parameters than traditional DL models, especially those based on transformers.

Graph preliminaries

An attributed graph is a graph with attributes on vertices. Each vertex $v \in V$ is annotated with a feature vector h_v



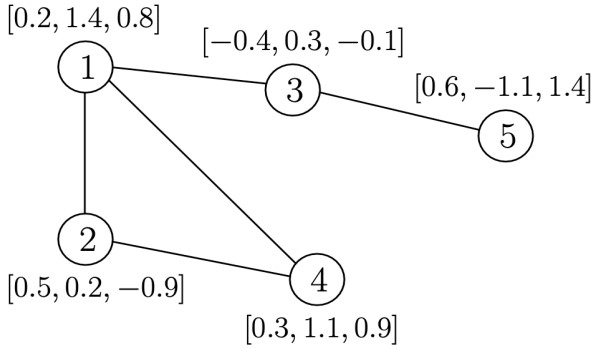
G

$h_1, \ldots, h_5 \in \mathbb{R}^3$

$h_1 = [0.2, 1.4, 0.8]^\top \quad h_3 = [-0.4, 0.3, -0.1]^\top$

Graph Machine learning tasks

An attributed graph is a graph with attributes on vertices. Each vertex $v \in V$ is annotated with a feature vector h_v



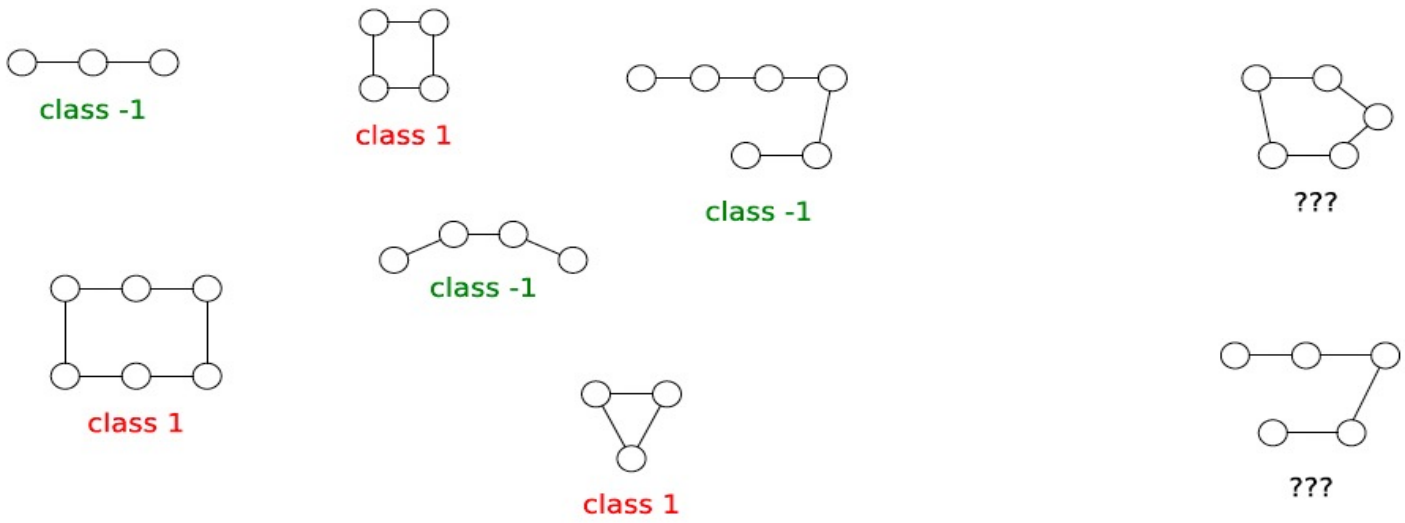
G

$h_1, \ldots, h_5 \in \mathbb{R}^3$

$h_1 = [0.2, 1.4, 0.8]^\top \quad h_3 = [-0.4, 0.3, -0.1]^\top$

- *Node classification*: given a graph with labels on some nodes, provide a high quality labelling for the rest of nodes
- *Graph clustering*: given a graph, group its vertices into clusters in such a way that there are *many edges within each cluster* and relatively few between the clusters (community detection)
- *Link Prediction*: given a pair of vertices, predict if they should be linked with an edge
- ***Graph classification***: given a set of graphs with known class labels for some of them, decide to which class the rest of the graphs belong.
- ***Graph Regression…***

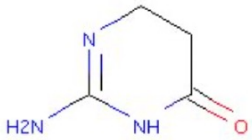
Graph classification



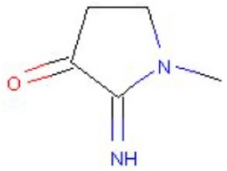
- Input data $G \in \mathcal{X}$
- Output $y \in \{-1, 1\}$
- Training set $\mathcal{D} = \{(G_1, y_1), \ldots, (G_n, y_n)\}$
- Goal: estimate a function $f : \mathcal{X} \to \mathbb{R}$ to predict y from $f(x)$

Graph regression - example

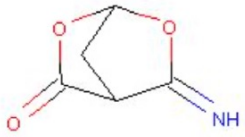
12 targets corresponding to molecular properties: ['mu', 'alpha', 'HOMO', 'LUMO', 'gap', 'R2', 'ZPVE', 'U0', 'U', 'H', 'G', 'Cv']



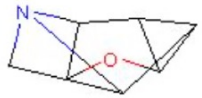
SMILES: NC1=NCCC(=O)N1
Targets: [2.54 64.1 -0.236 -2.79e-03 2.34e-01 900.7 0.12 -396.0 -396.0 -396.0 -396.0 26.9]



SMILES: CN1CCC(=O)C1=N
Targets: [4.218 68.69 -0.224 -0.056 0.168 914.65 0.131 -379.959 -379.951 -379.95 -379.992 27.934]

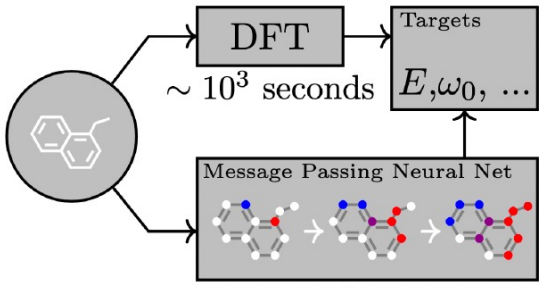


SMILES: N=C1OC2CC1C(=O)O2
Targets: [4.274 61.94 -0.282 -0.026 0.256 887.402 0.104 -473.876 -473.87 -473.869 -473.907 24.823]



SMILES: C1N2C3C4C5OC13C2C5
Targets: [? ? ? ? ? ? ? ? ? ? ? ?]

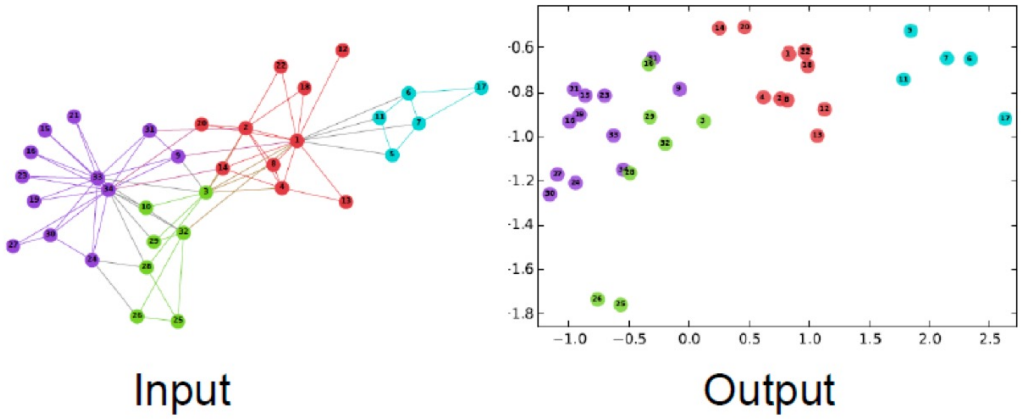
Perform **graph regression** to predict the values of the properties



[Gilmer et al., ICML'17]

Graph Neural Networks

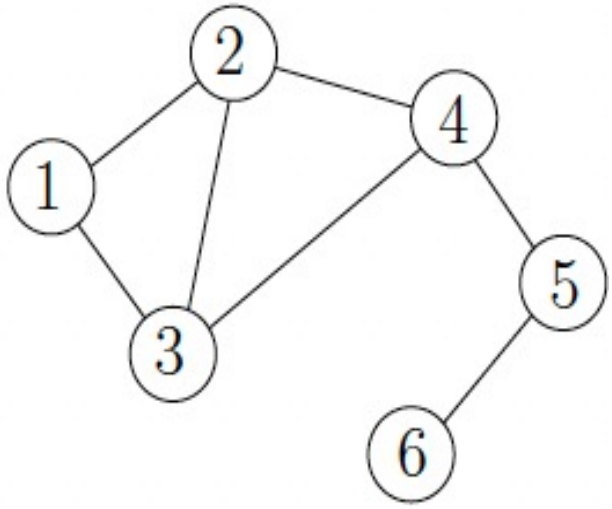
- Node Embeddings



- dimensionality d<<|V|
- similar vertices embedded close to each other in the low-dimensional space

Node embeddings - Example of message passing layer

$$h_1^{(t+1)} = f(W_0^{(t)} h_1^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_3^{(t)})$$
$$h_2^{(t+1)} = f(W_0^{(t)} h_2^{(t)} + W_1^{(t)} h_1^{(t)} + W_1^{(t)} h_3^{(t)} + W_1^{(t)} h_4^{(t)})$$
$$h_3^{(t+1)} = f(W_0^{(t)} h_3^{(t)} + W_1^{(t)} h_1^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_4^{(t)})$$
$$h_4^{(t+1)} = f(W_0^{(t)} h_4^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_3^{(t)} + W_1^{(t)} h_5^{(t)})$$
$$h_5^{(t+1)} = f(W_0^{(t)} h_5^{(t)} + W_1^{(t)} h_4^{(t)} + W_1^{(t)} h_6^{(t)})$$
$$h_6^{(t+1)} = f(W_0^{(t)} h_6^{(t)} + W_1^{(t)} h_5^{(t)})$$



Node embeddings – message passing

- series of *message passing* layers usually followed by one or more fully-connected layers
- message passing runs for T time steps and updates each vertex representation h_v^t based on its previous representation and the representations of its neighbors:

$$\mathbf{m}_v^{(t+1)} = \text{AGGREGATE}\left(\left\{\mathbf{h}_u^{(t)} \middle| u \in \mathcal{N}(v)\right\}\right)$$

$$\mathbf{h}_v^{(t+1)} = \text{COMBINE}\left(\mathbf{h}_v^{(t)}, \mathbf{m}_v^{(t+1)}\right)$$

- where $N(v)$ is the set of neighbours of v, and $AGGREGATE$ and $COMBINE$ are message functions and vertex update functions respectively

- $*$ $AGGREGATE$ function operates over an *unordered set of vectors* : *invariant to permutations of the neighbours*

Graph Convolutional Network (GCN)

Each message passing layer of the GCN model is defined as:

$$h_v^{(t+1)} = \mathrm{ReLU}\left(W^{(t)} \frac{1}{1+d(v)} h_v^{(t)} + \sum_{u \in \mathcal{N}(v)} W^{(t)} \frac{1}{\sqrt{(1+d(v))(1+d(u))}} h_u^{(t)}\right)$$

where $d(v)$ is the degree of node v

In matrix form, the above is equivalent to:

$$H^{(t+1)} = \mathrm{ReLU}\left(\hat{A}\, H^{(t)}\, W^{(t)}\right)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I$ and $\tilde{D}$ is a diagonal matrix such that $\tilde{D}_{ii} = \sum_{j=1}^{n} \tilde{A}_{ij}$

[Kipf and Welling, ICLR'17]

Need for Graph Generators

- Graph generator models can produce graphs with given properties or typology for various applications
- Modeling and studying networks in biology, engineering, and social sciences.
  - simulate the evolution of social networks,
  - the structure of protein-protein interactions,
  - topology of power grids.
- Discovering new graph structures and properties.
  - generate novel chemical and molecular structures,
  - design new materials,
  - explore the space of possible graphs with certain characteristics.
- Completing and enhancing existing graphs.
  - fill in missing nodes and edges,
  - add new features and attributes,
  - improve the quality and diversity of graph data.

Graph Generators – heuristic based models

- Erdős–Rényi Random Graph Model
- Barabasi Albert graph generator
- Kronecker graphs – pattern recursion
- Stochastic Block models
- ….

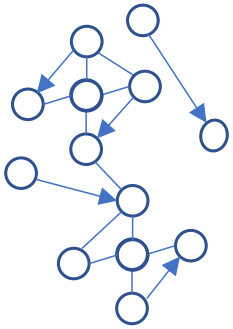
Need for **Deep** Graph Generators

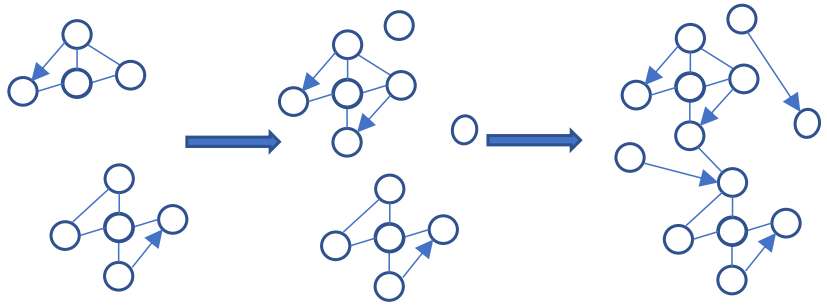
- Traditional graph generation models (i.e. *Erdős-Rényi, Barabási-Albert model, Kronecker graphs, Stochastic block models*) based on *assumptions / heuristics* oversimplifying the underlying distributions of graphs.
- Deep models for graph-structured data enable effective complex graph generation
  - *can learn the generative model directly from observed data*, without relying on hand-engineered processes or pre-defined statistical properties.
  - *capture the complex joint probability of all nodes and edges in the graph*, and generate realistic graphs that match the structural characteristics of the target distribution.
  - incorporate various advanced methods: (i.e. attention mechanisms, reinforcement learning) to enhance the quality and diversity of the generated graphs.

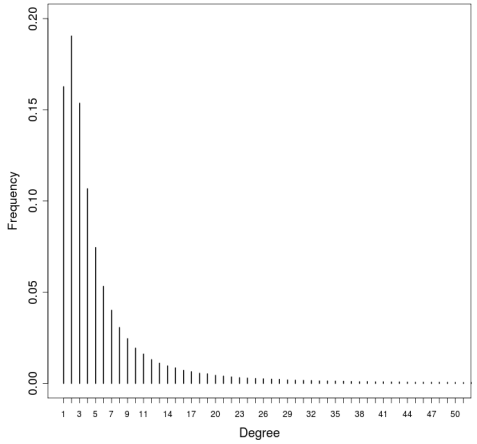
Graph Generative Models

Graph generation challenging task:

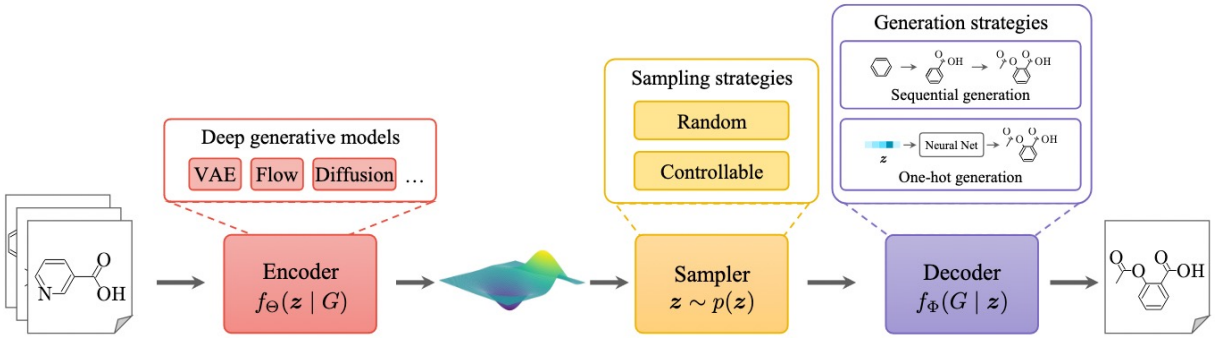
- higher-order (and non symmetric) relationships
- Sparsity and no deterministic order in processing nodes,
- long-tailed distribution of relationships: some are frequent others very rare in real-life graphs.
- dynamic and temporal: change over time, different states with time.







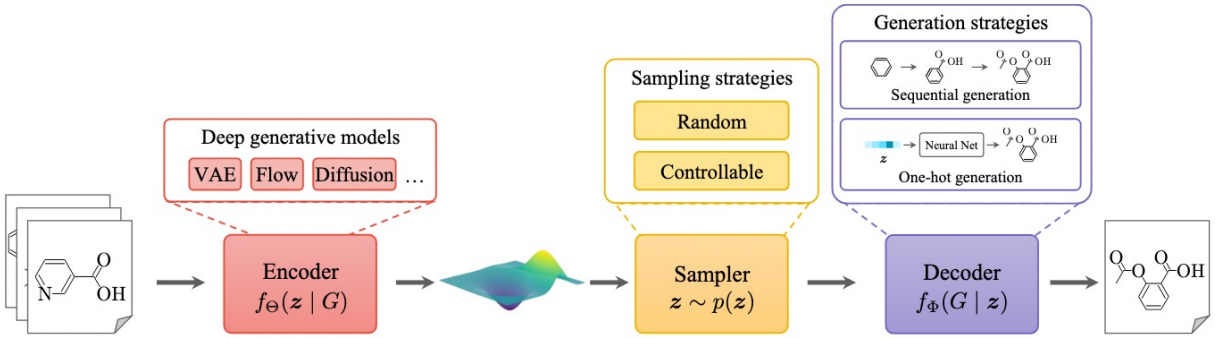
Overview of deep graph generation



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- *encoder* maps observed graphs into a stochastic distribution;
- *sampler* draws latent representations from that distribution;
- *decoder* receives latent codes and produces graphs

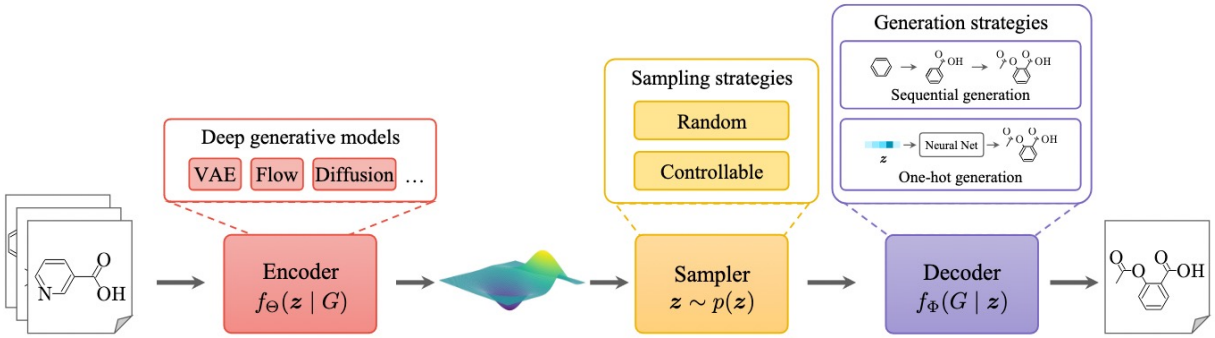
Overview of deep graph generation - Encoder



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- Encoder - The encoding function $f_\Theta(z / G)$ represent discrete graph objects as dense, continuous vectors.
- employ probabilistic generative models (e.g., variational graph neural networks) as the encoder.
- encoder function f_Θ outputs the parameters of a stochastic distribution following a prior distribution $p(z)$.

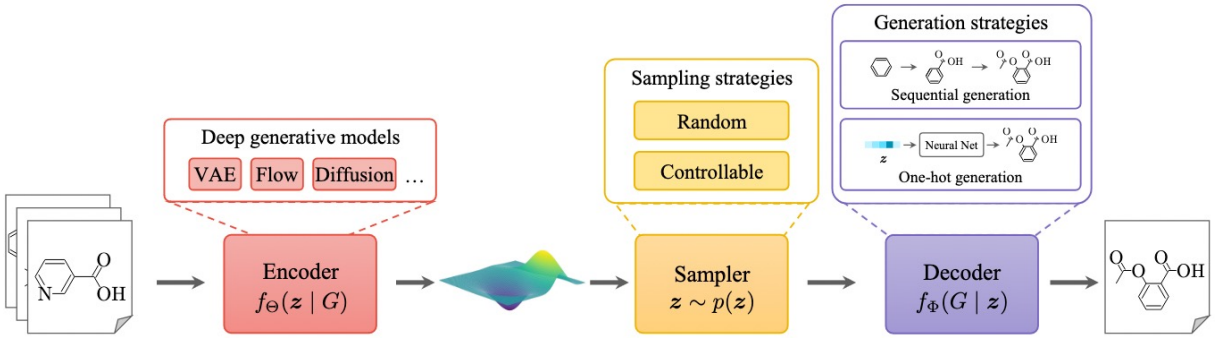
Overview of deep graph generation - Sampler



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- sample latent representations from learned distribution $z \sim p(z)$.
- two sampling strategies: random sampling and controllable sampling.
- Random: randomly sampling latent codes from the learned distribution.
- controllable: sample latent code in an attempt to generate new graphs with desired properties.
  - In practice, controllable sampling usually depends on different types of deep generative models and requires an additional optimization term beyond random generation.

Overview of deep graph generation - Decoder

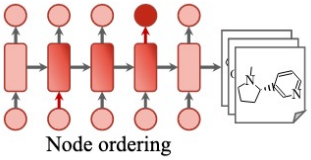


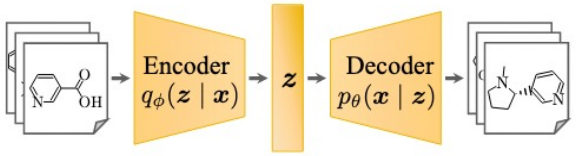
A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

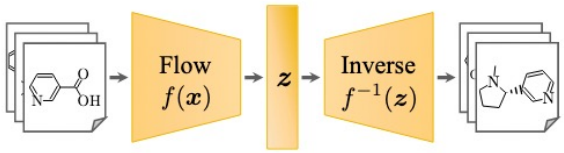
- The decoder receives the latent representations sampled from the learned distribution and generates graph structures.
- decoder is more complicated due to the discrete, non-Euclidean nature of graph objects.
- Decoder types:
  - sequential generation: generating graphs in consecutive steps, one node/edge at a time.
  - one-shot generation - generating node/edge feature matrices in single step.
- not all methods include all components – i.e. (GANs) do not include a specific encoder

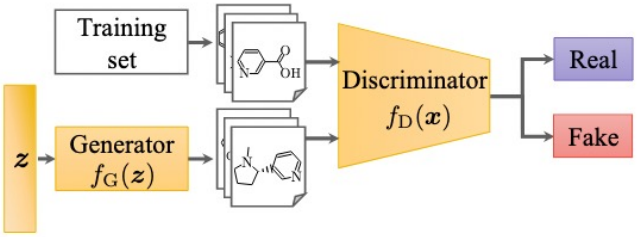
Graph generative models for deep graph generation

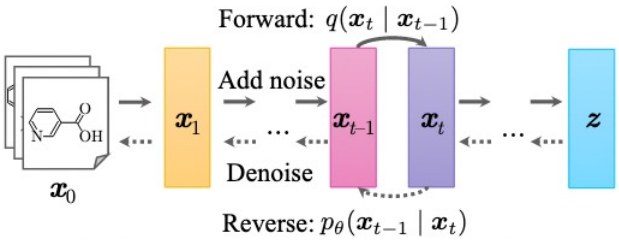
- **auto-regressive models**
- variational autoencoders
- normalizing flows
- generative adversarial networks
- diffusion models







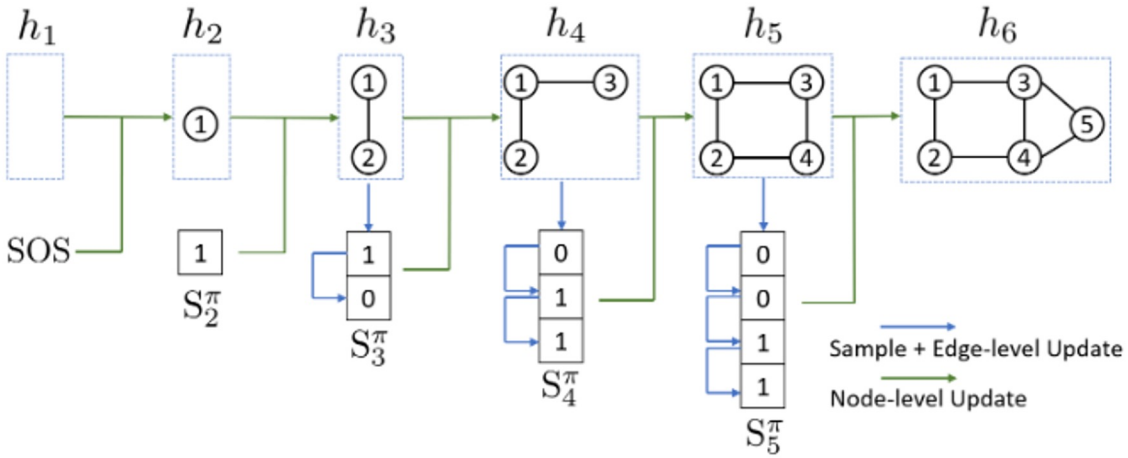




A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

GraphRNN [You et al., 2018]

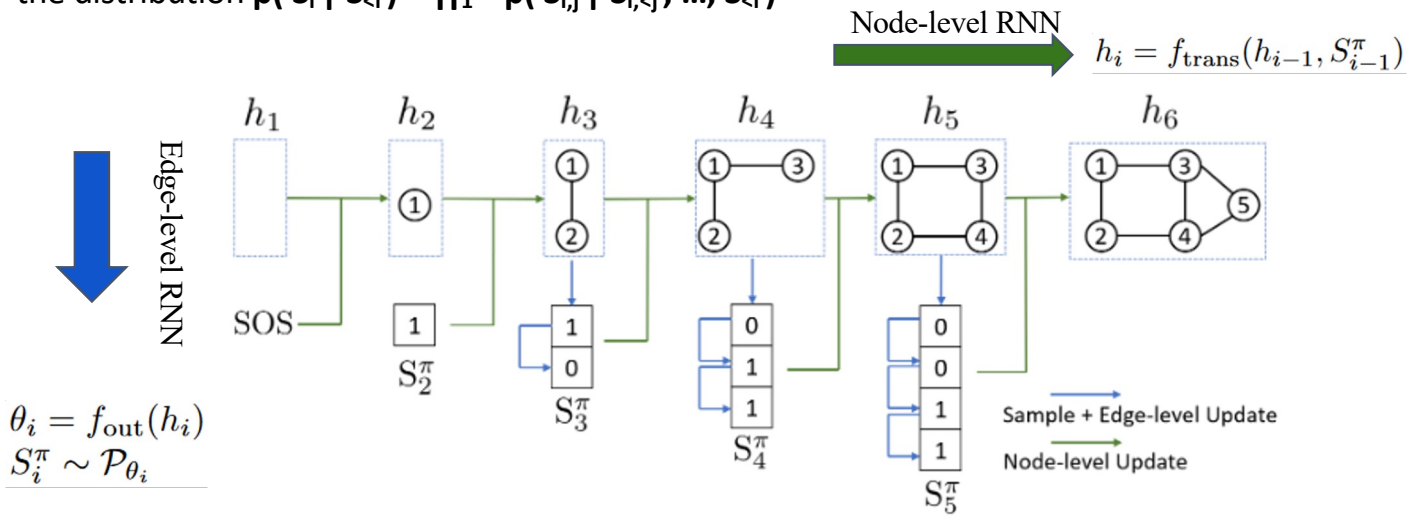
- **Autoregressive model** for graph generation
  - Key insight: Graph **G** with node permutation $\boldsymbol{\pi}$ can be uniquely mapped into a sequence of node and edge additions $\mathbf{S}^{\boldsymbol{\pi}}$
  - Model the generation process with two RNNs
    - Node-level: generate a state for a new node
    - Edge-level: generate edges for the new node based its hidden state



GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models
Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, Jure Leskovec

GraphRNN - Method

- ➔ Omitting symbol of node permutation **π**, Graph **G~p(G)** can be represented as a sequence of adjacency vectors **(S_1 , …, S_n)**
- ➔ **S_i** is a **(i-1)** dimensional *vector represents edges between **i** and previous nodes*: **$\{0, 1\}^{i-1}$**
- ➔ **p(G)** is related to **p(S)** with **$p(S) = \prod_1^{n+1} p(S_i \mid S_1, \ldots, S_{i-1})$**. This product can be modelled by a RNN (node-level) with state transition possible modelled by another RNN (edge-level), which models the distribution **$p(S_i \mid S_{<i}) = \prod_1^{i-1} p(S_{i,j} \mid S_{i,<j}, \ldots, S_{<i})$**



Evaluation for GraphRNN

Visual comparison

- ➔ First row: Training set
- ➔ Third row: Kronecker graph, Mixed-Membership Stochastic Block model and Barabasi–Albert graph

Evaluation for GraphRNN

	Community (160,1945)			Ego (399,1071)			Grid (361,684)			Protein (500,1575)		
	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit
E-R	0.021	1.243	0.049	0.508	1.288	0.232	1.011	0.018	0.900	0.145	1.779	1.135
B-A	0.268	0.322	0.047	0.275	0.973	0.095	1.860	0	0.720	1.401	1.706	0.920
Kronecker	0.259	1.685	0.069	0.108	0.975	0.052	1.074	0.008	0.080	0.084	0.441	0.288
MMSB	0.166	1.59	0.054	0.304	0.245	0.048	1.881	0.131	1.239	0.236	0.495	0.775
GraphRNN-S	0.055	0.016	0.041	0.090	**0.006**	0.043	0.029	10^{-5}	0.011	0.057	**0.102**	**0.037**
GraphRNN	**0.014**	**0.002**	**0.039**	**0.077**	0.316	**0.030**	$\mathbf{10^{-5}}$	**0**	$\mathbf{10^{-4}}$	**0.034**	0.935	0.217

$$\mathrm{MMD}^2(p||q) = \mathbb{E}_{x,y\sim p}[k(x,y)] + \mathbb{E}_{x,y\sim q}[k(x,y)] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)].$$

	Community-small (20,83)					Ego-small (18,69)				
	Degree	Clustering	Orbit	Train NLL	Test NLL	Degree	Clustering	Orbit	Train NLL	Test NLL
GraphVAE	0.35	0.98	0.54	13.55	25.48	0.13	0.17	0.05	12.45	14.28
DeepGMG	0.22	0.95	0.40	106.09	112.19	0.04	0.10	0.02	21.17	22.40
GraphRNN-S	**0.02**	0.15	**0.01**	31.24	35.94	0.002	**0.05**	**0.0009**	8.51	9.88
GraphRNN	0.03	**0.03**	**0.01**	28.95	35.10	**0.0003**	**0.05**	**0.0009**	9.05	10.61

Maximum Mean Discrepancy (MMD)

➔ Compare all moments of the empirical distributions using an exponential kernel with Wasserstein distance

80% decrease of **MMD** over traditional baselines: E-R, B-A, Kronecker, MMSB

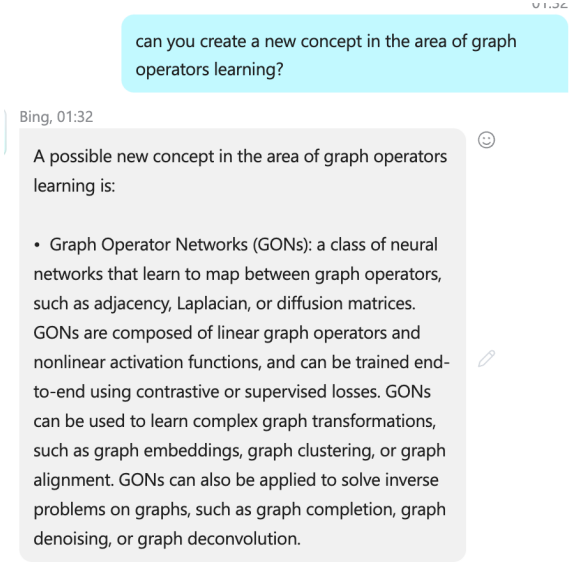
90% decrease of **MMD** over deep learning baselines

22% smaller average **NLL** gap compared to deep learning baselines

How Intelligent is AI (personal opinion)

- LLMs know everything that has ever existed in digital form…
- The size of the models (>400B parameters) creates non linear phenomena in capacity
- Already the best art is AI [1]
- Can create new concepts: YES..
- Future…we opened Pandoras box…

Jason Allen - “Space Opera Theatre ”



1: https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligenc artists.html

THANK YOU!

- DASCIM web page: http://www.lix.polytechnique.fr/dascim
- Google Scholar: https://bit.ly/2rwmvQU
- Twitter: @mvazirg