Deep Learning for Forecasting

Tim Januschowski, Kashif Rasul, Lorenzo Stella

June 25, 2023

Your instructors

- Tim Januschowski. Director Pricing Platform, Zalando
- Kashif Rasul. Sr. Scientist, Morgan Stanley
- Lorenzo Stella. Sr. Applied Scientist, Amazon Web Services

Connect with us on Twitter, LinkedIn!

Your instructors' other activities

- Tim runs "Modern Forecasting in Practice": https://maven.com/tim-januschowski/modern-forecasting
- Kashif actively works with Hugging Face
- GluonTS – give it a star on github!

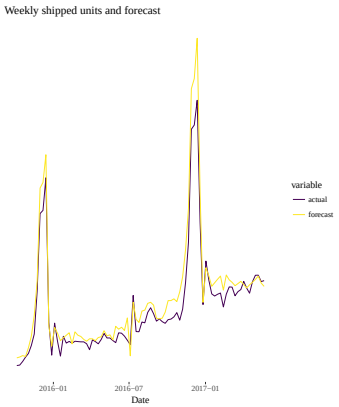
The repo for this workshop

Our workshop today

1. Tim: Intro to Deep Learning for Forecasting 45 mins
2. everyone: 10 min break
3. Kashif: Transformers 30 mins
4. everyone: 10 min break
5. Lorenzo & everyone: Put it into Practice 90 mins

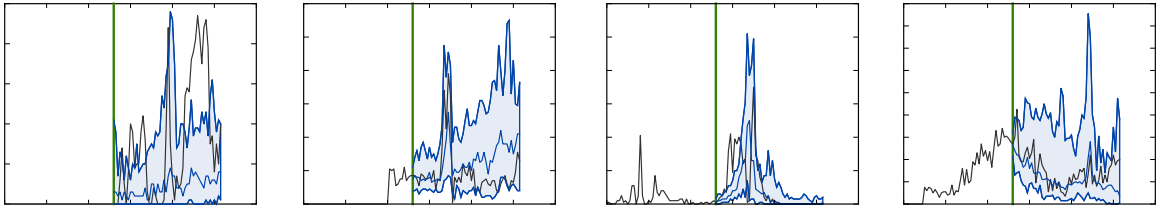
Introduction to Practical Forecasting Problems

Retail Demand Forecast, the strategic view



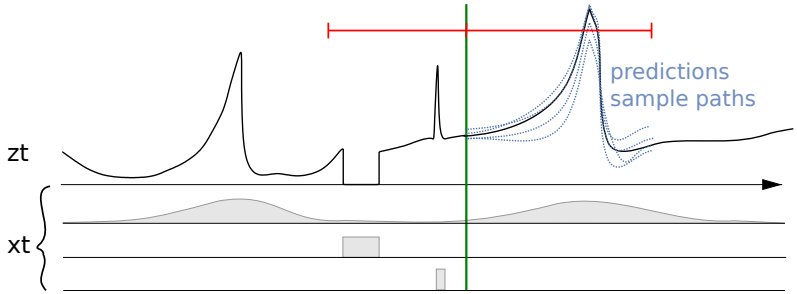
- Problem: predict overall Amazon retail demand years into the future
- Decision Problems: topology planning, market entry/segment analyses

Retail Demand Forecast, the operational view



- Problem: predict the demand for a each product available on Amazon
- Decision Problems: how many units to order when and where, when to mark products down

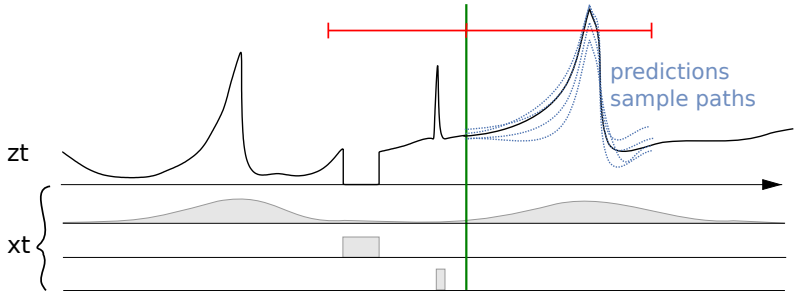
Forecasting Problems: General Setup



- Predict the future behavior of a (univariate) time series $z_{i,t}$ for item $i \in I$ given its past

$$z_{i,0}, \ldots, z_{i,T-2}, z_{i,T-1}, z_{i,T} \Longrightarrow P(z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h})$$

Forecasting Problems: General Setup



- Predict the future behavior of a (univariate) time series $z_{i,t}$ for item $i \in I$ given its past

$$z_{i,0}, \ldots, z_{i,T-2}, z_{i,T-1}, z_{i,T} \Longrightarrow P(z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h})$$

- Make optimal decisions

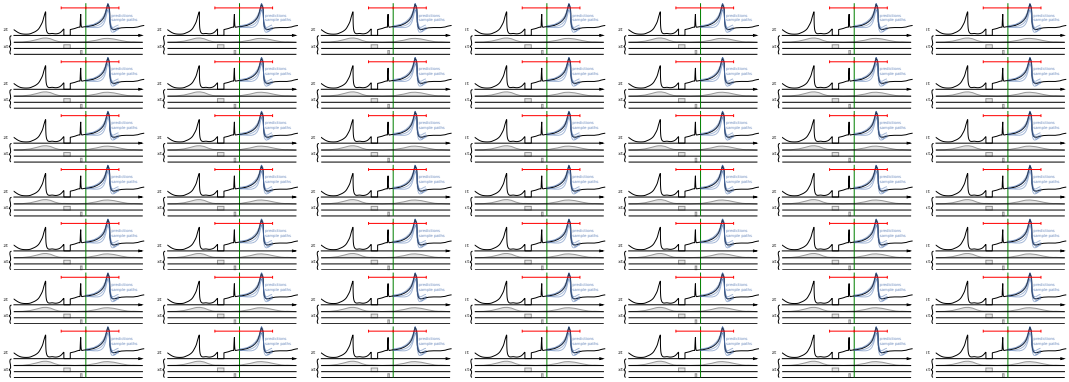
$$\text{best action} = \underset{a}{\operatorname{argmin}}\, \mathrm{E}_{\mathrm{P}}[\text{cost}(a, z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h})]$$

Forecasting Problems: Two Extremes

- small number of time series
- sufficient historical data
- limited meta data
- hand-crafted models
- statistician and econometrician have the right skills typically

We refer to these problems as strategic forecasting problems [Januschowski et al., 2019]

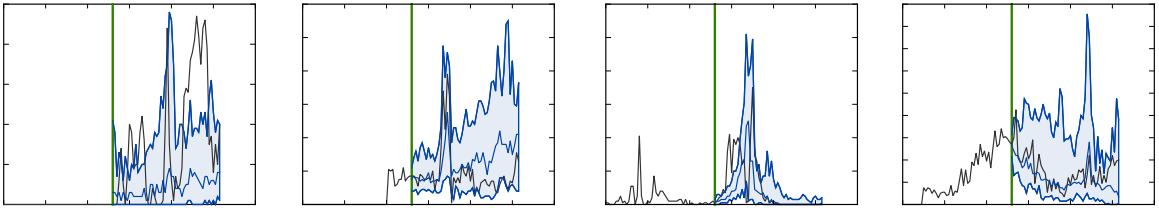
Forecasting Problems: Two Extremes



Examples: demand for products of a retailer, work force cohorts of a company in its locations, compute capacity needs per region and server type.

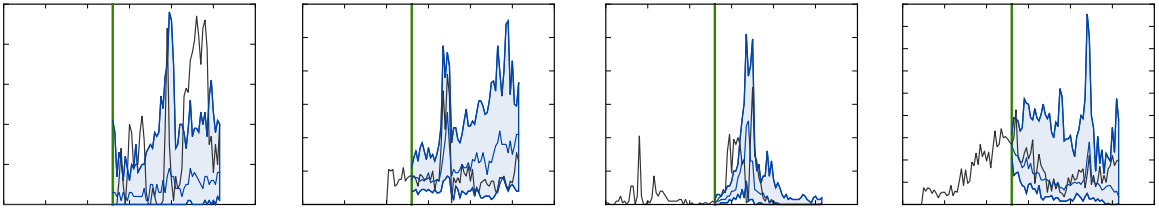
We refer to these problems as operational forecasting problems.

Operational forecasting problems: characteristics & approaches



- cold start/new items
- short cycles
- burstiness, sparsity
- high degree of automation of downstream systems
- ratio of people/time series $\ll 1$

Operational forecasting problems: characteristics & approaches

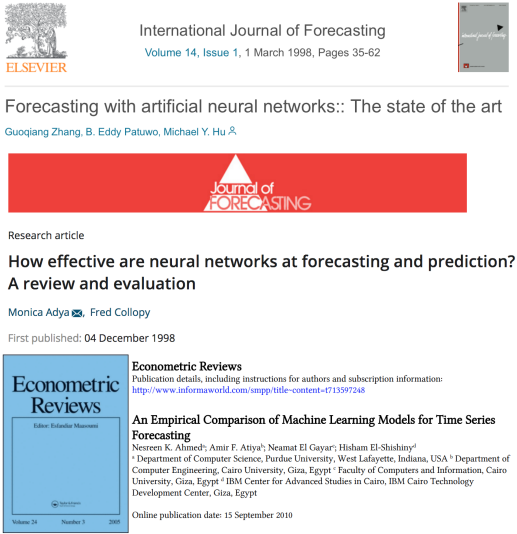


- cold start/new items
- short cycles
- burstiness, sparsity
- high degree of automation of downstream systems
- ratio of people/time series $\ll 1$

Two extremes:

- complex pipeline of simple models (adapt traditional models to new problems)
- simple pipeline including end-to-end learning with complex models like **neural networks**

Forecasting with Neural Networks – Timeline



ELSEVIER

International Journal of Forecasting

Volume 14, Issue 1, 1 March 1998, Pages 35-62

Forecasting with artificial neural networks:: The state of the art

Guoqiang Zhang, B. Eddy Patuwo, Michael Y. Hu

Journal of FORECASTING

Research article

How effective are neural networks at forecasting and prediction? A review and evaluation

Monica Adya, Fred Collopy

First published: 04 December 1998

Econometric Reviews

Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713597248

An Empirical Comparison of Machine Learning Models for Time Series Forecasting

Nesreen K. Ahmed[a]; Amir F. Atiya[b]; Neamat El Gayar[c]; Hisham El-Shishiny[d]

[a] Department of Computer Science, Purdue University, West Lafayette, Indiana, USA [b] Department of Computer Engineering, Cairo University, Giza, Egypt [c] Faculty of Computers and Information, Cairo University, Giza, Egypt [d] IBM Center for Advanced Studies in Cairo, IBM Cairo Technology Development Center, Giza, Egypt

Online publication date: 15 September 2010

- 1963 Weather forecasting with adaptive linear neurons (Hu)
- 1986 Backpropagation (Rumelhart et al.)
- 1988 NNs using backpropagation applied to forecasting; positive results (Werbos)
- 199x Many authors applying mostly feed-forward models to various forecasting problem (single time series)
- 1998 Review articles: "The outcome of all of these studies has been somewhat mixed"; "While ANNs provide a great deal of promise, they also embody much uncertainty."
- 2000 M3 competition – simple methods declared the winner
- 200x Less work on NN-based forecasting methods
- 2012 AlexNet wins ImageNet competition – start of the Deep Learning revival (Krizhevsky et al.)
- 2014 Generating Sequences With RNNs (Graves); seq2seq architecture (Sutskever et al.)
- 2014- Modern deep learning techniques (RNNs, CNNs) get applied to forecasting (across time series)
- 2018 M4 competition: combination of NNs and classical techniques wins

Forecasting with Neural Networks – Timeline

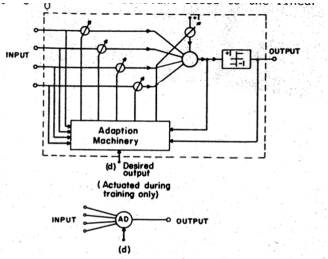


Fig. 1 The ADALINE Concept

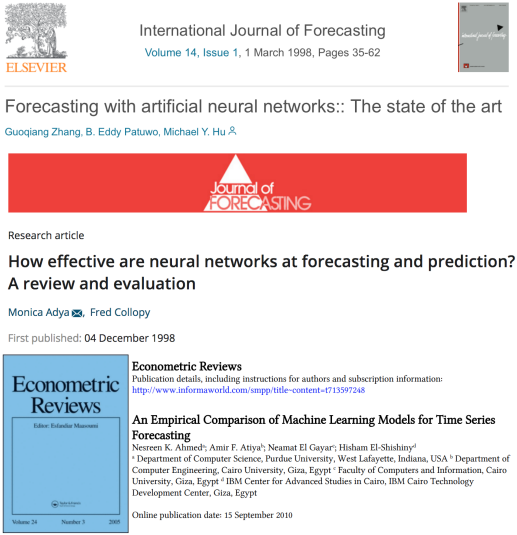
[Widrow, 1987]

1963 Weather forecasting with adaptive linear neurons (Hu)

Michael J. C. Hu, a graduate student from Hong Kong, is conducting experiments based on weather forecasting. He fed into an Adaline the patterns of atmospheric pressure readings on selected days from over 500 million square miles extending from Alaska to northern Mexico and from Hawaii to the Rockies. Correlated in the training were the patterns of actual weather in the Bay Area on corresponding days. Then he asked Adaline to make predictions based on the pressure data alone for 18 days on which it had not been trained. Adaline scored 78 per cent right on today forecasts, 89 per cent on tonight forecasts, and 83 per cent on tomorrow forecasts. For the same 18 days, U.S. Weather Bureau forecasters scored 78, 89, and 67 per cent. Hu is now working on a much more comprehensive study.

https://www-isl.stanford.edu/~widrow/papers/j1963adalinesmarter.pdf

Forecasting with Neural Networks – Timeline



- 1963 Weather forecasting with adaptive linear neurons (Hu)
- 1986 Backpropagation (Rumelhart et al.)
- 1988 NNs using backpropagation applied to forecasting; positive results (Werbos)
- 199x Many authors applying mostly feed-forward models to various forecasting problem (single time series)
- 1998 Review articles: "The outcome of all of these studies has been somewhat mixed"; "While ANNs provide a great deal of promise, they also embody much uncertainty."
- 2000 M3 competition – simple methods declared the winner
- 200x Less work on NN-based forecasting methods
- 2012 AlexNet wins ImageNet competition – start of the Deep Learning revival (Krizhevsky et al.)
- 2014 Generating Sequences With RNNs (Graves); seq2seq architecture (Sutskever et al.)
- 2014– Modern deep learning techniques (RNNs, CNNs) get applied to forecasting (across time series)
- 2018 M4 competition: combination of NNs and classical techniques wins

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

YES!

Deep Learning for Forecasting: Shouldn't it just work?

"Time series? Just use an RNN!"

- Is there anything special about forecasting?
- Shouldn't the hugely successful models from areas like NLP and CV *just work*?

YES!

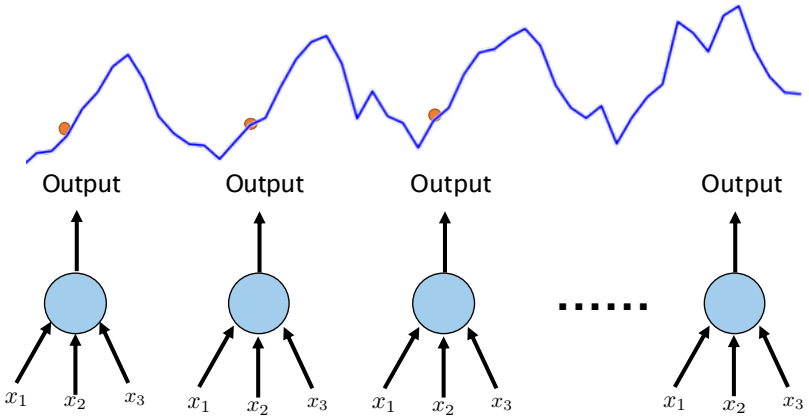
- ... mostly.
- Challenges: scaling, probability distribution outputs, sample efficiency, incorporating prior knowledge

Deep Learning for Forecasting: Outline

1. Non-linear regression, feed-forward models (aka MLPs)
2. Training neural network models
3. Basic model structures
4. Basic model blocks
   - Convolutional Neural Nets
   - Recurrent Neural Nets
5. Probabilistic Forecasting with Neural Nets
6. Deep Probabilistic Models (let's see if we get there, timewise)

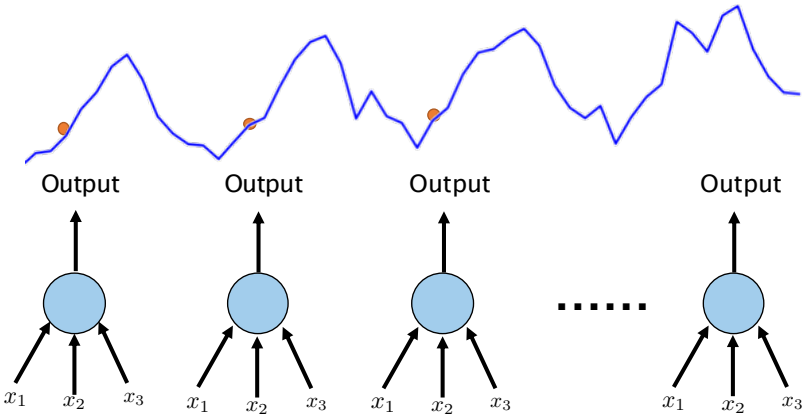
From Linear Regression to Feed-Forward Networks

From Linear Regression to Feed-Forward Neural Networks



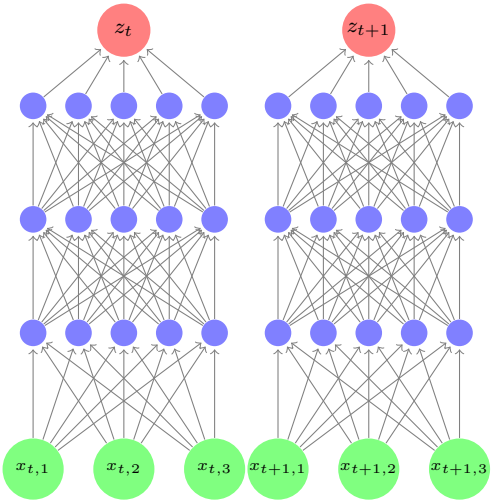
$$z_t = \boldsymbol{w}^T \boldsymbol{x}_t$$

From Linear Regression to Feed-Forward Neural Networks



$$z_t = \sigma(\boldsymbol{w}_l^T(\sigma(W_{l-1}^T(\sigma(W_{l-2}^T(\cdots W_0^T \boldsymbol{x}_t))))) := \text{DEEP-NET}(\boldsymbol{x}_t)$$

Feed-Forward Neural Networks (Multi-layer Perceptron (MLPs))

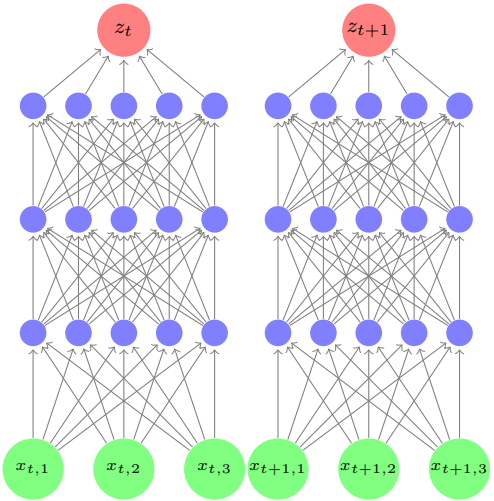


- Linear model + non-linear hidden layers
- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear *activation* function,

$$h_{l,j} = \sigma\left(\mathbf{w}_{l,j}^{\top}\mathbf{h}_{l-1} + b_{l,j}\right)$$

- FF Neural Networks are flexible general function estimators
- More (and larger) hidden layers $\rightarrow$ more complex functions

Feed-Forward Neural Networks (Multi-layer Perceptron (MLPs))



- Main advantage over linear models: Can learn complex input-output relationships

$\Rightarrow$ Less manual feature engineering

- Building blocks for more complex structures, for example, N-BEATS [Oreshkin et al., 2019].
- Main disadvantage: more data needed for training
- Careful tuning (e.g. of regularization, learning rate, etc.) might be necessary for good results
- Sensitive to scaling of inputs

Training Neural Networks

General recipe

Pick a class of functions $f(\mathbf{x}; \theta)$ and learn the parameters θ by minimizing some notion of error on a *training set*,

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_i L\left(\mathbf{z}_i, f(\mathbf{x}_i, \theta)\right)$$

- Optimization algorithm of choice: Stochastic Gradient Descent (SGD)
  1. For each iteration $k = 1, 2, 3, \ldots$
  2. Randomly pick a *minibatch* of examples $i_1, i_2, \ldots, i_B$
  3. Compute the batch loss $L_k = \sum_b L\left(\mathbf{z}_{i_b}, f(\mathbf{x}_{i_b}, \theta)\right)$
  4. Compute the gradient of the loss $g_k = \nabla_\theta L_k(\theta)$
  5. Update the parameters $\theta_k = \theta_{k-1} - \eta g_k$
  6. (Optional but recommended: Adjust the learning rate η)

Loss Functions

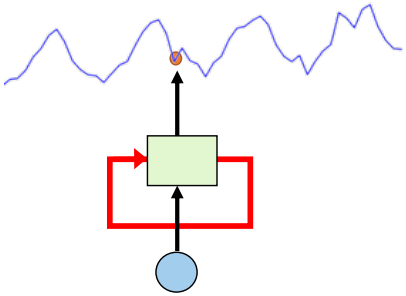
- In *supervised learning*, one key modelling choice is the *loss function*.
- In the forecasting context, the loss function compares a forecast to the truth (on historical training data where the truth is known).
- For point forecasts, a loss function compares two real numbers, e.g. $\widehat{z}_t$ vs. z_t,

$$e_t = (\widehat{z}_t - z_t)^2$$

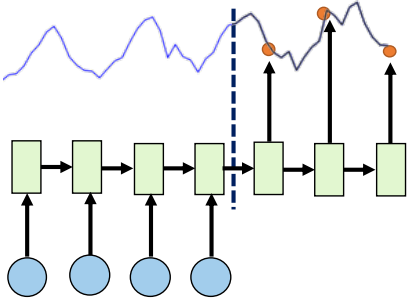
- For distribution forecasts, a loss function compares a forecast distribution F_t to a real number z_t, using a so-called *scoring rule*, e.g., negative log likelihood.

Basic Model Structures

Basic Model Structures



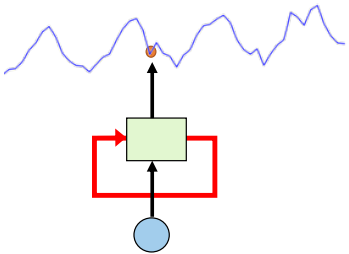
Canonical (One-to-One)



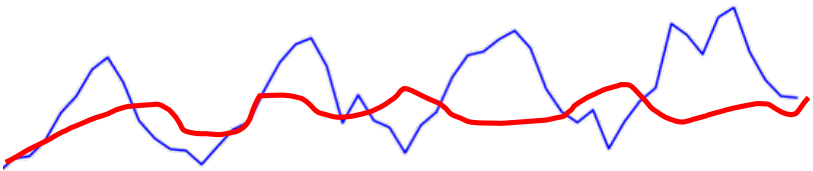
Seq2Seq (Many-to-Many)

One-to-One Structure (Generative Model)

$$f_t : x_t \mapsto z_t$$



Canonical (One-to-One)

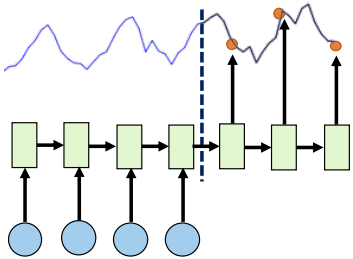


Training sequence

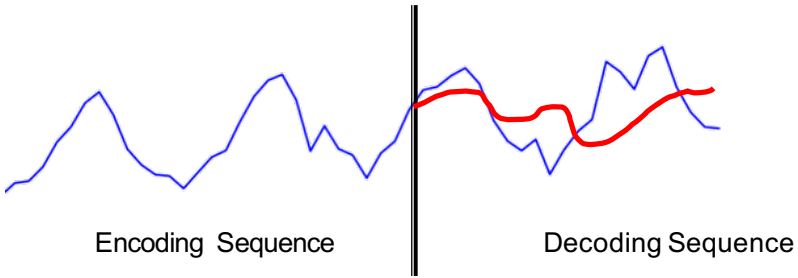
How well does the prediction reconstruct the the observed time series?

Sequence-to-Sequence / Many-to-Many Structure (Discriminative Model)

$$f : \{z_1, \cdots, z_{T_e}\} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



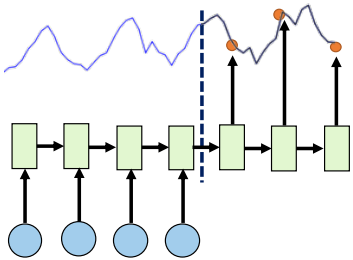
Seq2Seq (Many-to-Many)



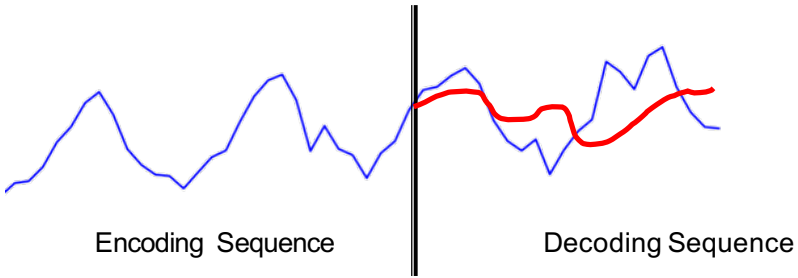
How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence?

Sequence-to-Sequence / Many-to-Many Structure (Discriminative Model)

$$f : \{z_1, \cdots, z_{T_e}\} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



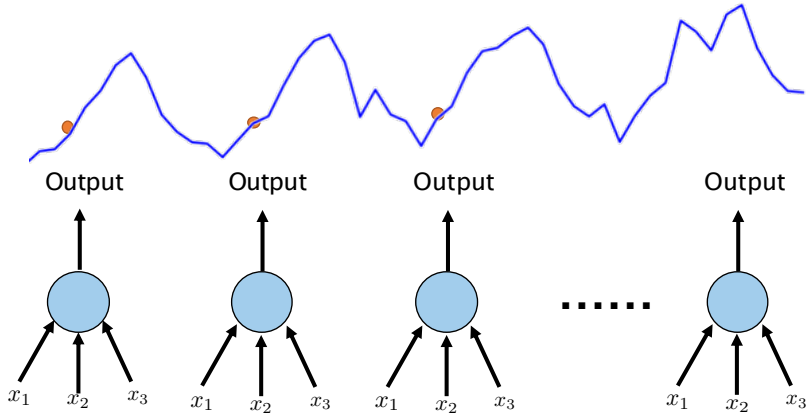
Seq2Seq (Many-to-Many)



How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence? Conceptually close to multivariate regression

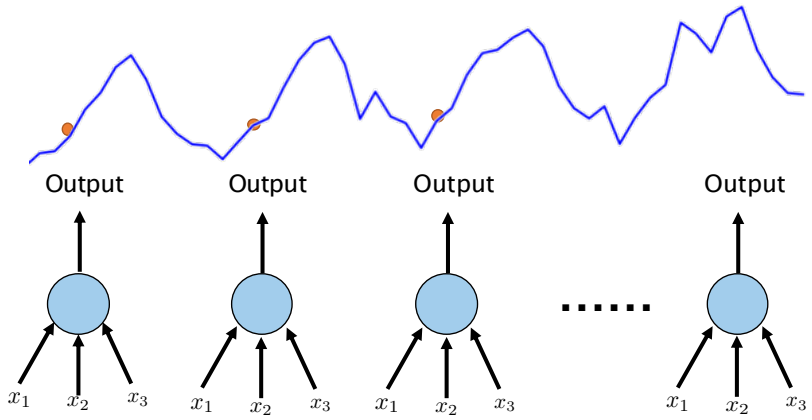
Basic Model Blocks: Recurrent Neural Networks

Recap: MLP for Forecasting



$$z_t = \text{DEEP-NET}(\boldsymbol{x}_t)$$

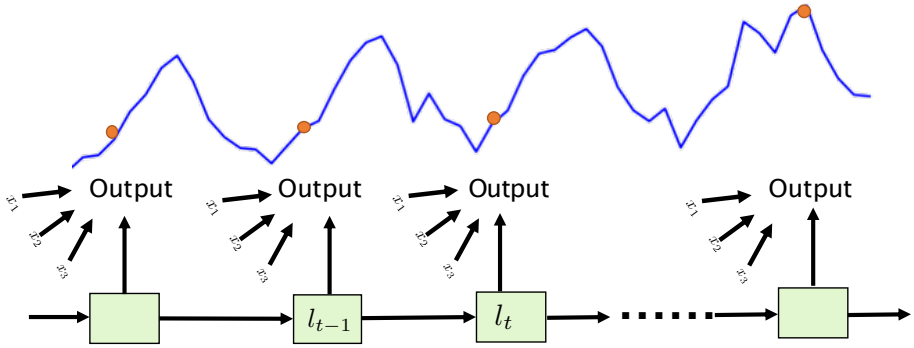
Recap: MLP for Forecasting



$$z_t = \text{DEEP-NET}(\boldsymbol{x}_t)$$

How about the sequential relationship?

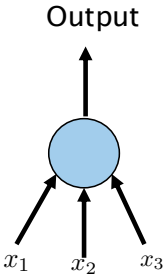
Recap: State-Space Models for Forecasting



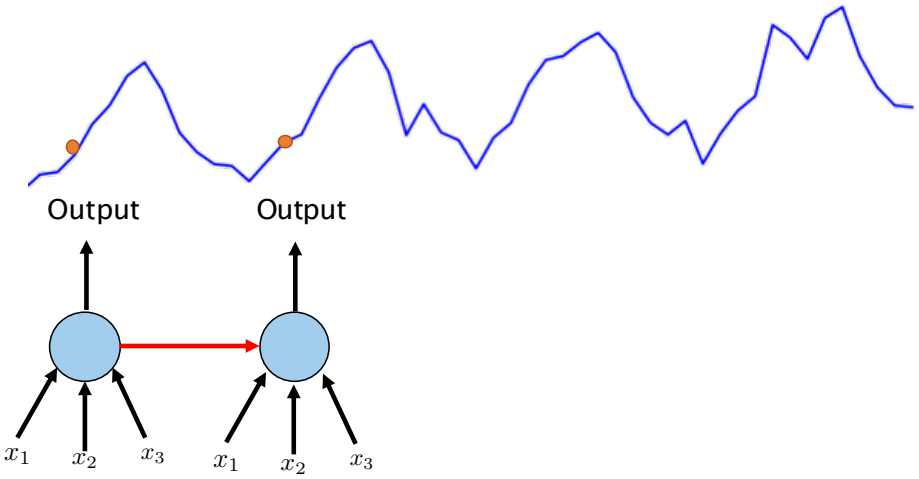
$$\boldsymbol{l}_t = \boldsymbol{l}_{t-1} + \alpha \cdot \epsilon_t$$
$$z_t = \boldsymbol{v}^T \boldsymbol{l}_t + \boldsymbol{w}^T \boldsymbol{x}_t + \epsilon_t$$

Can we do the same with NNs?

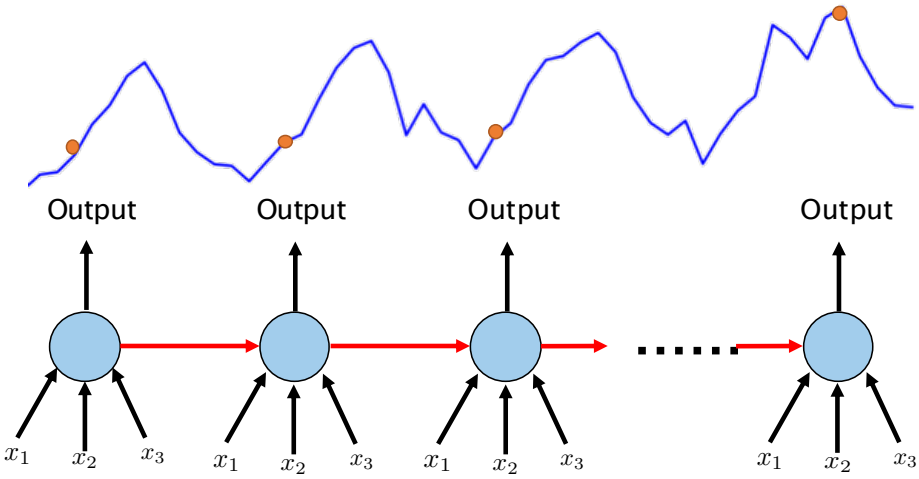
From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN

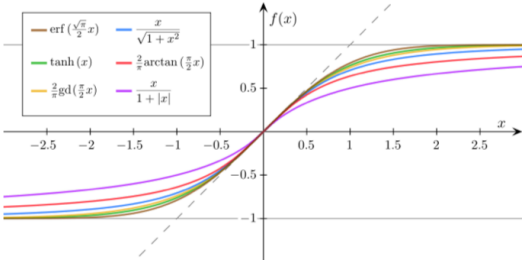


From Latent State (Exponential Smoothing) to Recurrent NN

Current hidden state h_t combines

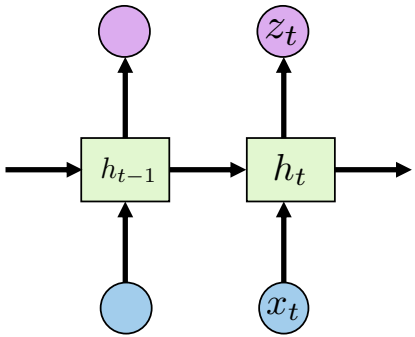
- the previous hidden state h_{t-1}
- input features x_t

and goes into



Source: Wikipedia

RECURRENT NEURAL NETWORK



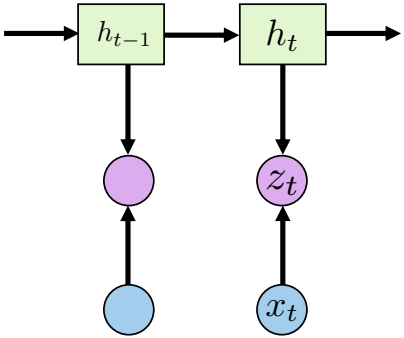
$$\boldsymbol{h}_t = \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$

$$z_t = \sigma(\theta h_t)$$

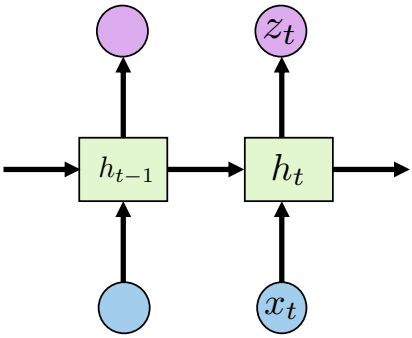
Central idea: Exponential Smoothing

today = yesterday's information + new knowledge

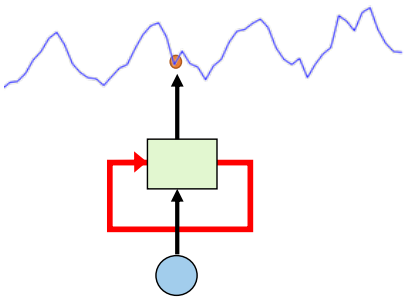
STATE-SPACE MODEL



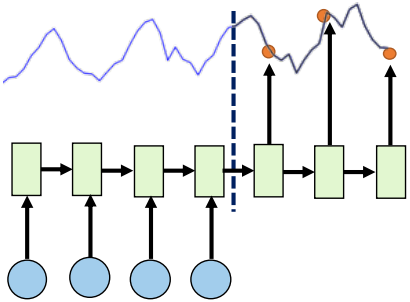
RECURRENT NEURAL NETWORK



Basic Model Structures (again)



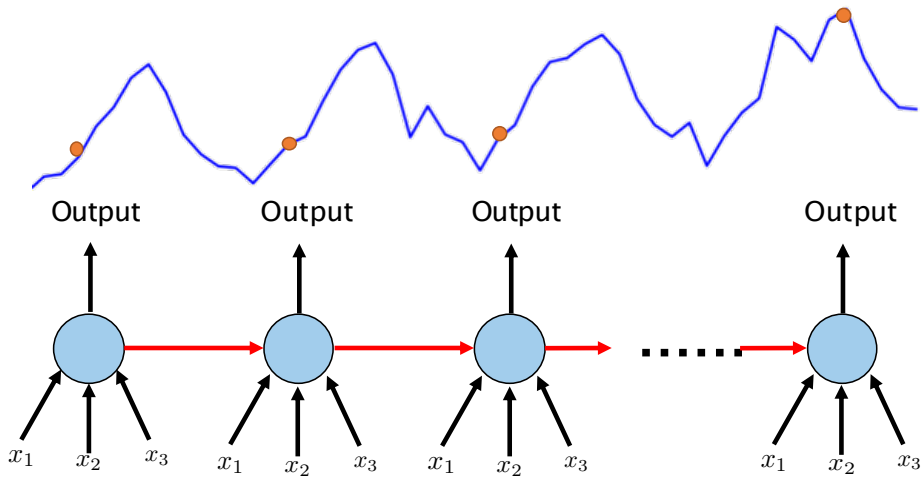
Canonical (One-to-One)



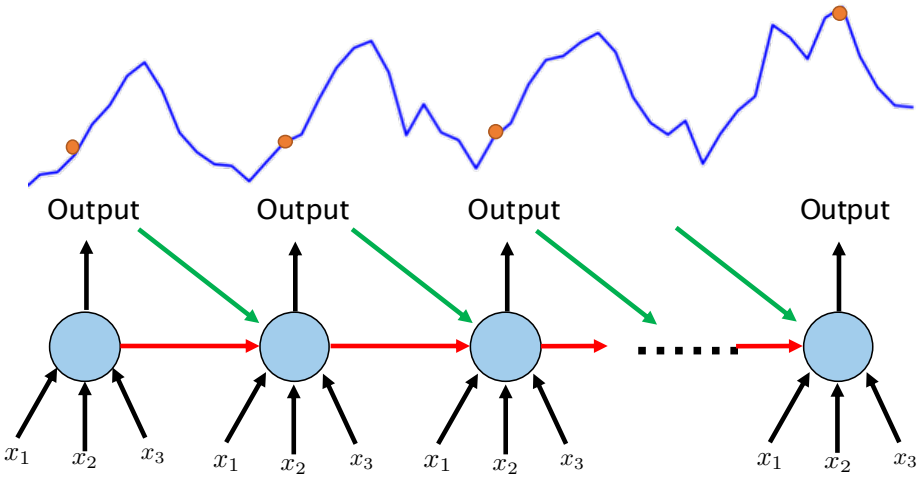
Seq2Seq (Many-to-Many)

- Canonical (One-to-One) RNN: DeepAR [Flunkert et al., 2017], AR-MDN [Mukherjee et al., 2018], Deep LSTM [Yu et al., 2017], ...
- Sequence-to-Sequence (Many-to-Many) models: Diffusion Convolutional RNNs [Li et al., 2018], MQ-RNN [Wen et al., 2017], ...

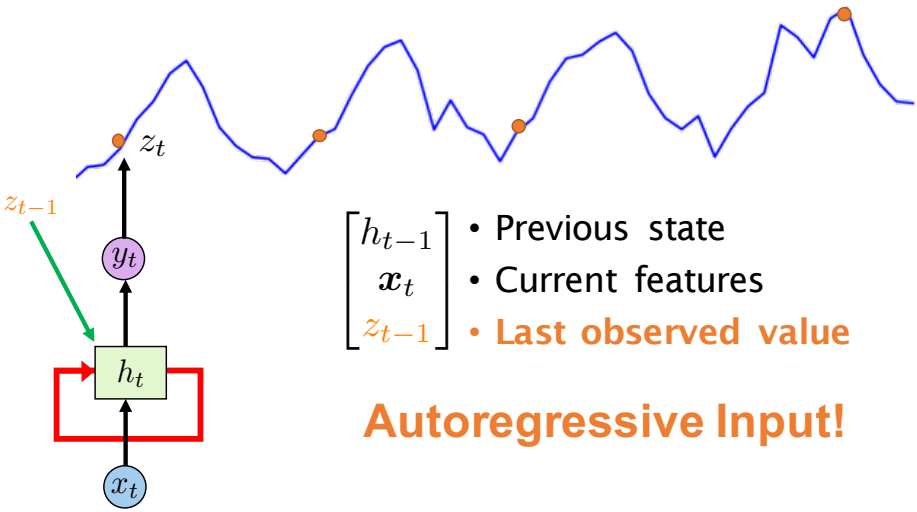
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: DeepAR [Flunkert et al., 2017]

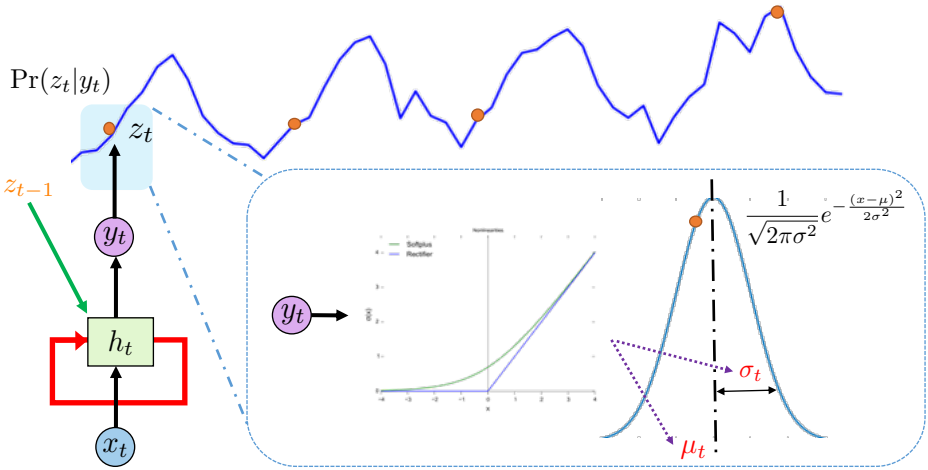


$$\begin{bmatrix} h_{t-1} \\ \boldsymbol{x}_t \\ z_{t-1} \end{bmatrix}$$

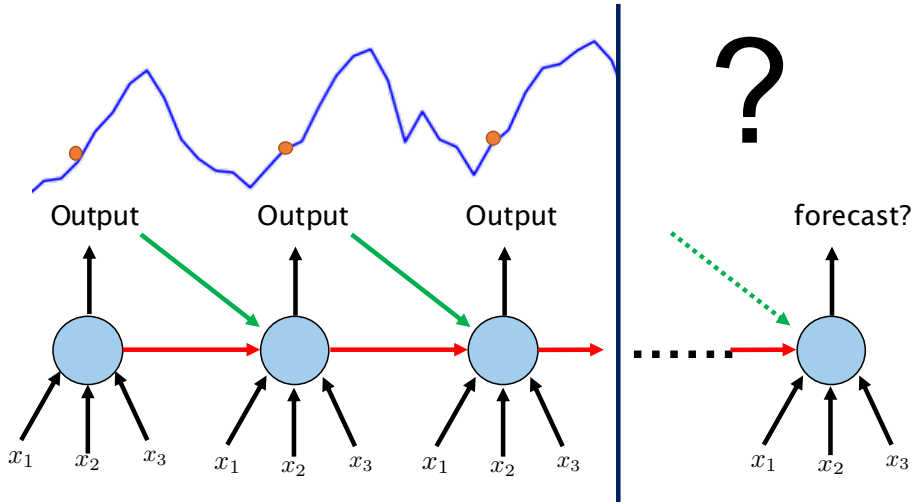
- Previous state
- Current features
- **Last observed value**

Autoregressive Input!

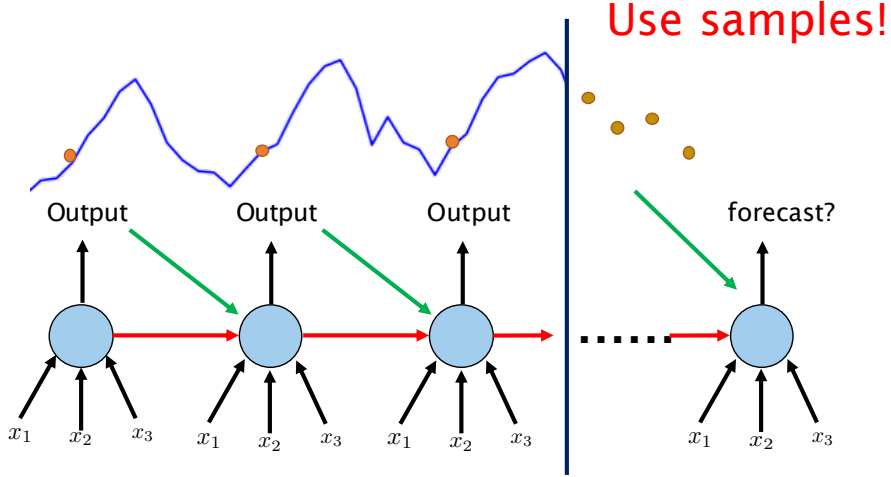
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: How do we do forecast?

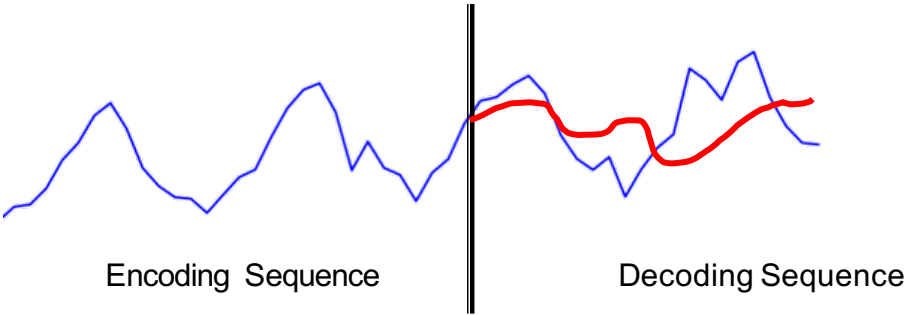


Canonical RNN Structure: How do we do forecast?

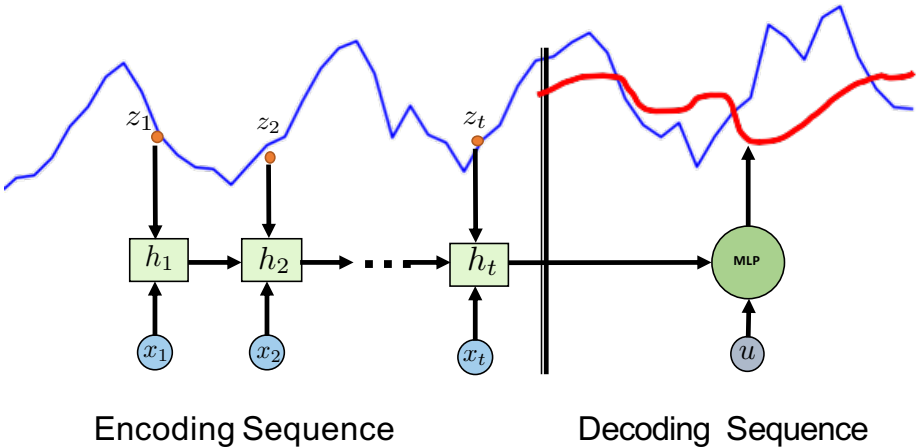


Sequence to Sequence (Seq2Seq) Structure: Many-to-Many

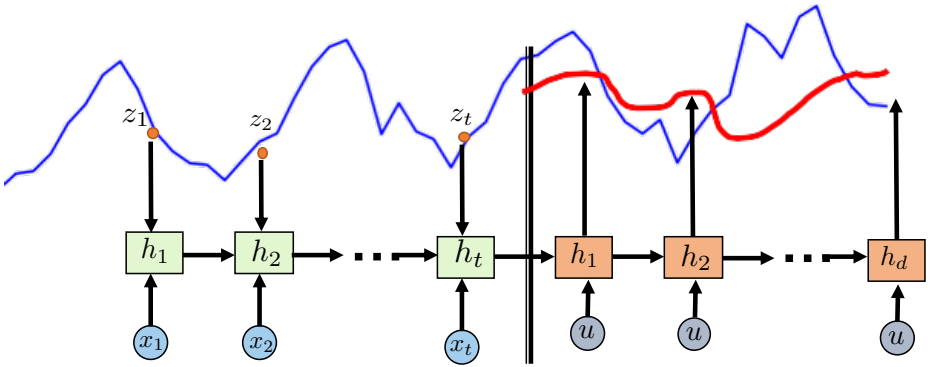
$$f_{encoder} : \{z_1, \cdots, z_{T_e}\} \mapsto \boldsymbol{h}_{T_e}$$
$$f_{decoder} : \boldsymbol{h}_{T_e} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



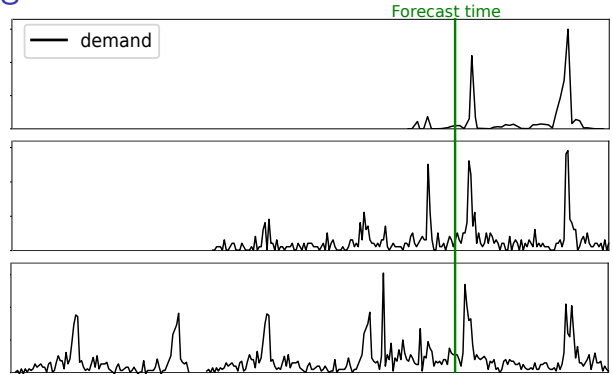
Seq2Seq: RNN-MLP [Wen et al., 2017]



Seq2Seq: RNN-RNN

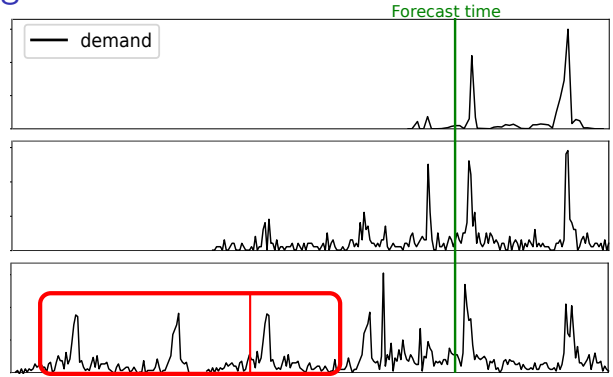


Seq2Seq: Training



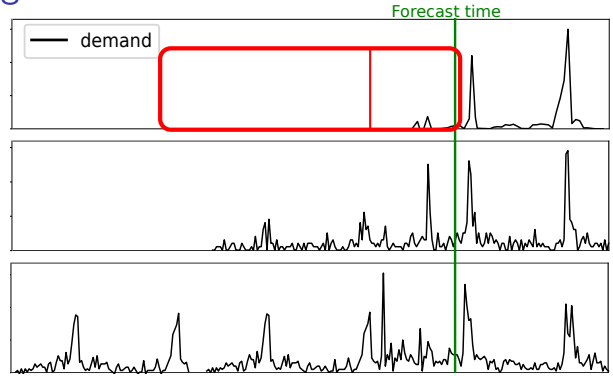
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s

Seq2Seq: Training



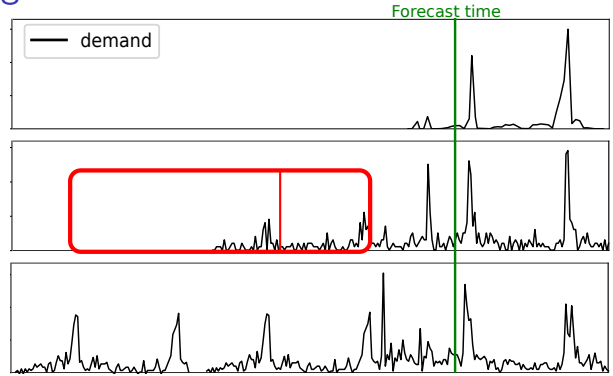
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



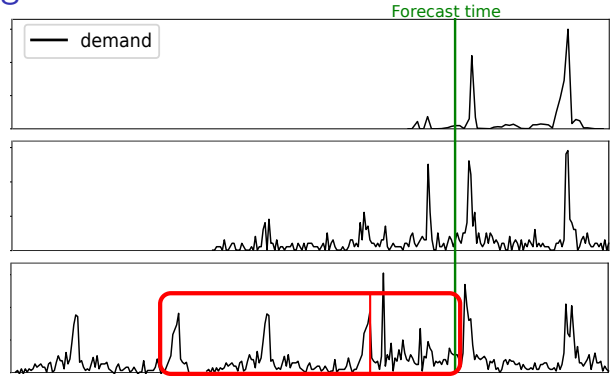
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



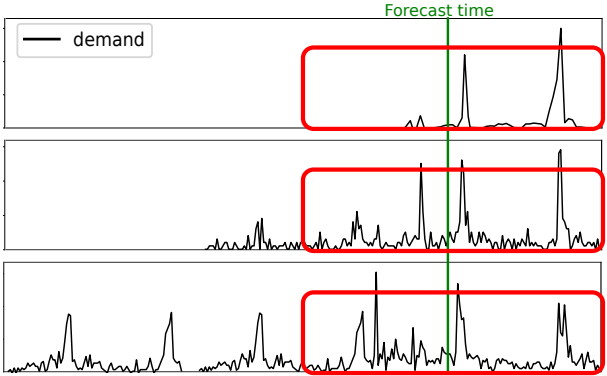
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training

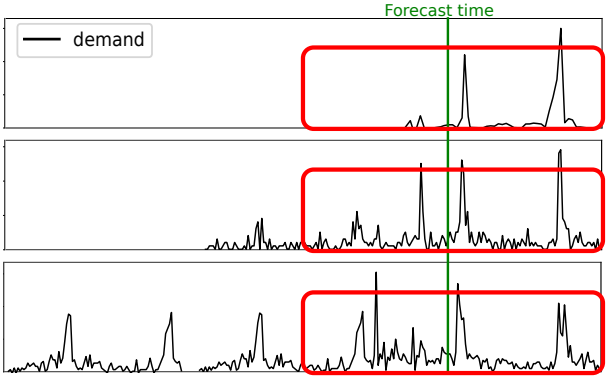


- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Prediction

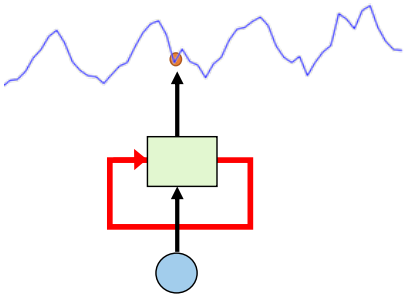


Seq2Seq: Prediction

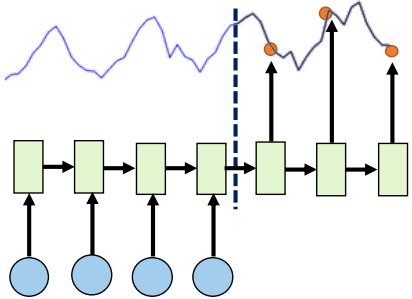


- target z_t is unobserved after the forecast time

Comparison: Canonical (One-to-One) vs. Seq2Seq (Many-to-Many)



Canonical (One-to-One)



Seq2Seq (Many-to-Many)

Canonical

- input features need to be available during prediction phase
- no need to re-train for different prediction length (forecast horizon)

Seq2Seq

- can have disjoint encoding and decoding features
- needs re-training when changing the decoder length

Convolutional Neural Networks

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- Typical CNN model architectures combine convolutional layers with other layer types
- CNNs with 2D convolutions are extremely successful in computer vision applications

  $\Longrightarrow$ encode spatial invariance
- 1D convolutions are a promising alternative to RNNs for sequential data

  $\Longrightarrow$ encode temporal invariance, "stationarity"

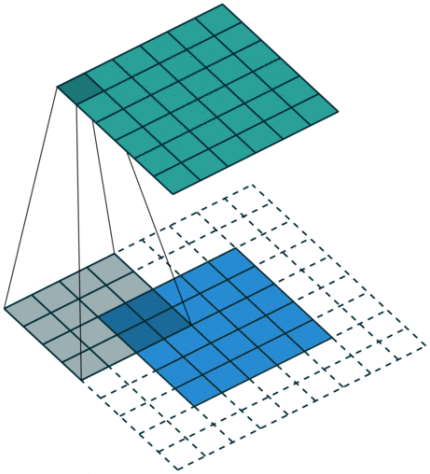
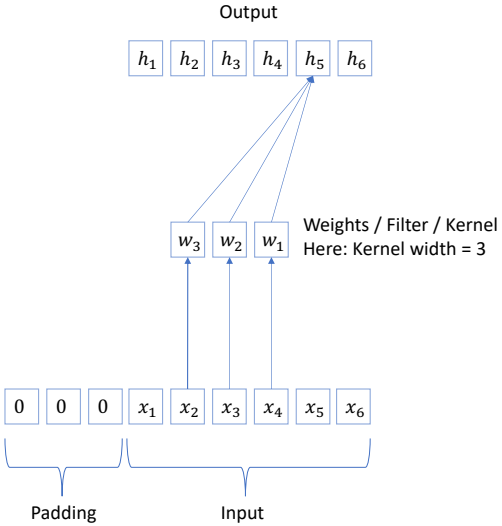


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning;
https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers

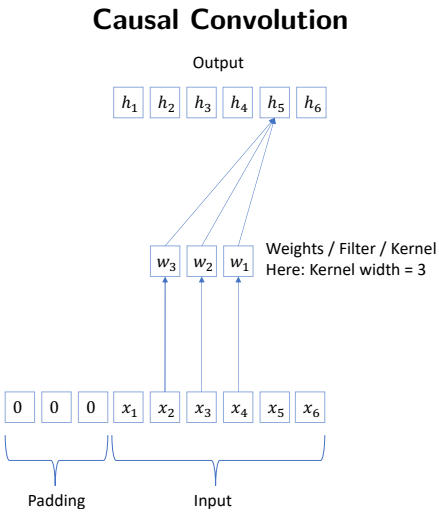


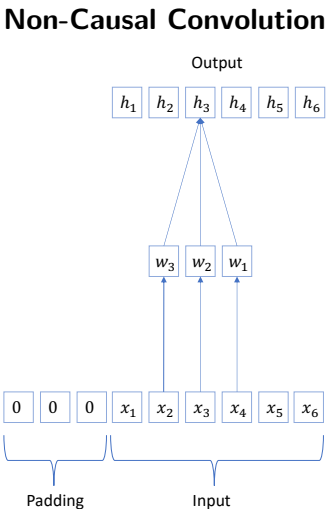
- The output h_j of a neuron j in a convolution layer is a discrete convolution of the inputs $\mathbf{x}$ with the layer's weigths/filter $\mathbf{w}$.
- For a one-dimensional convolution with a kernel with width D we have

$$h_j = \sum_{d=1}^{D} w_d x_{j-d}$$

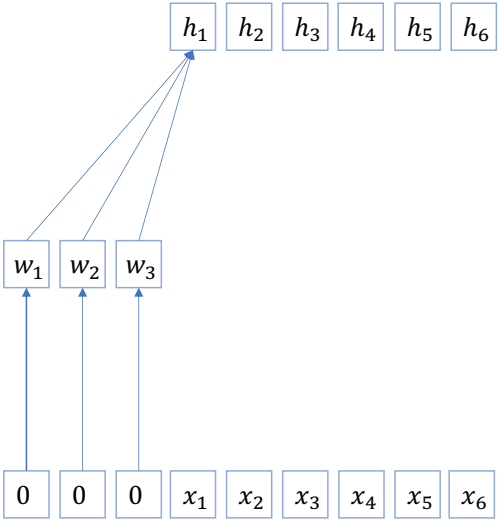
- Padding is used to shift the input relative to the output and change the behavior around the edges (causal vs. non-causal)

Causal vs. Non-Causal Convolution

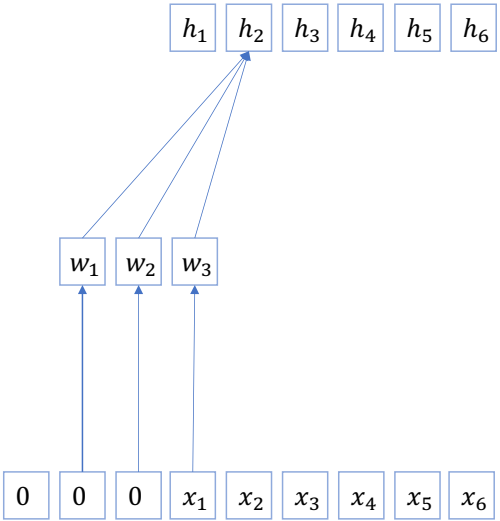




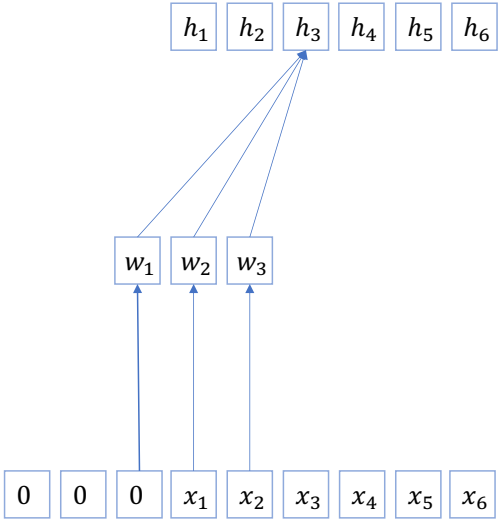
1D Causal Convolution



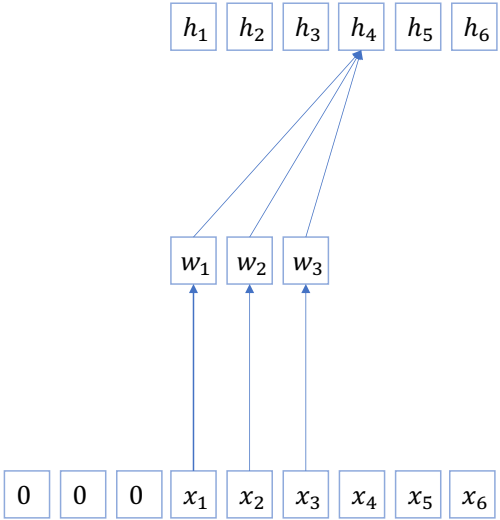
1D Causal Convolution



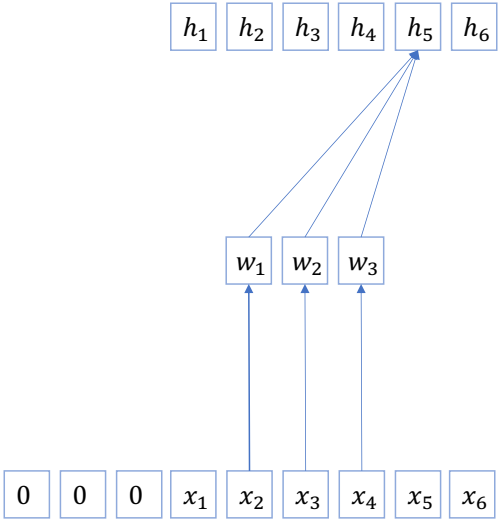
1D Causal Convolution



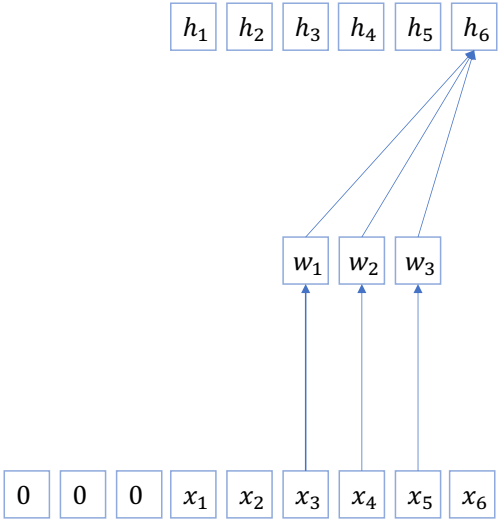
1D Causal Convolution



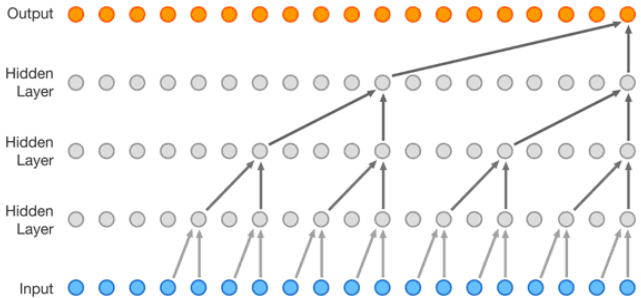
1D Causal Convolution



1D Causal Convolution



Canonical: Dilated Causal Convolution and WaveNet [Van Den Oord et al., 2016]



- Dilation increases receptive field
- Forecast is generated in an autoregressive fashion
- More complex structures including gating and residual links [Van Den Oord et al., 2016]

Canonical: Dilated Causal Convolution and WaveNet [Van Den Oord et al., 2016]

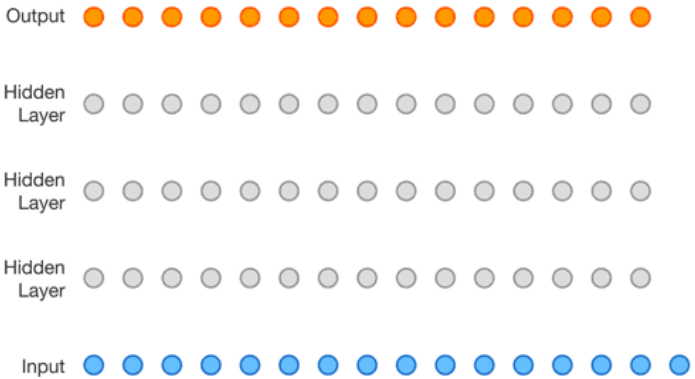
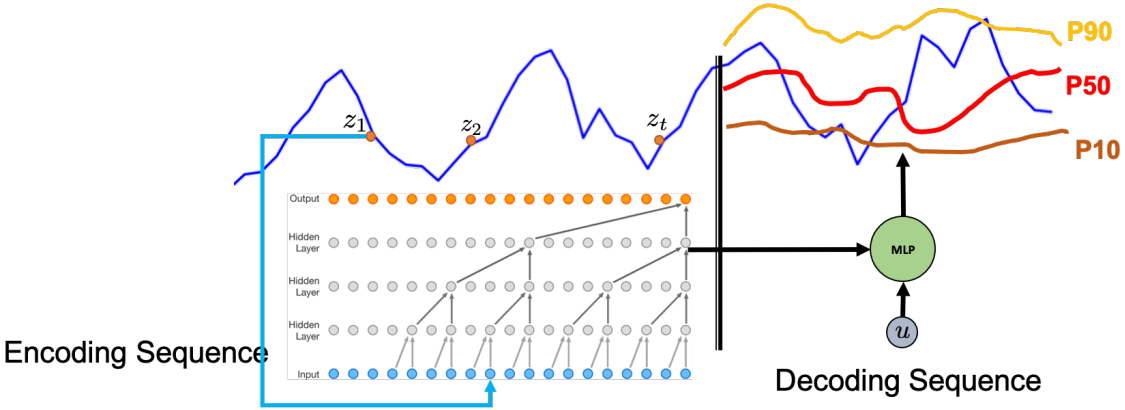


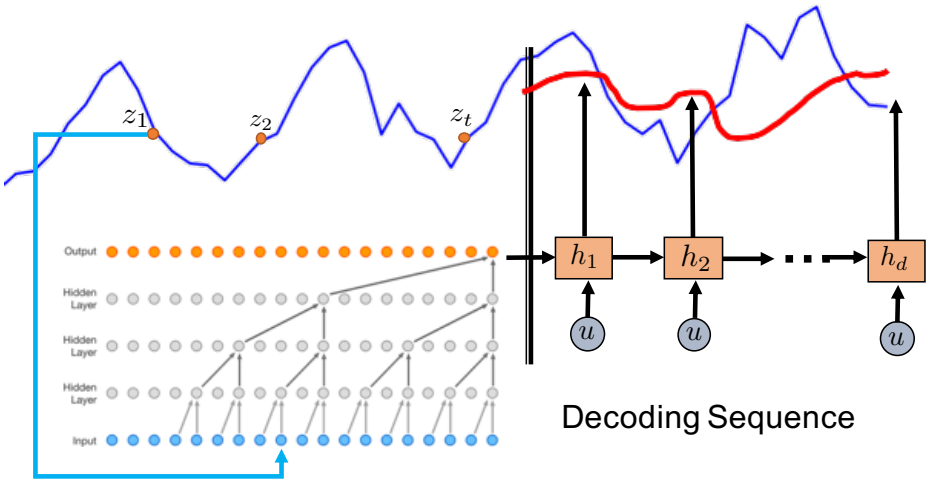
Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Seq2Seq: RNN-MLP [Wen et al., 2017]



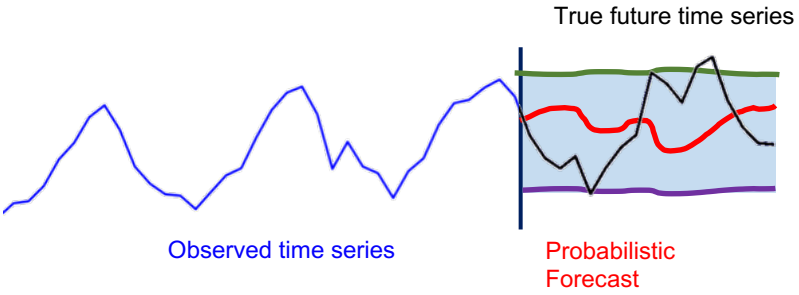
- Powers Amazon's supply chain: https://www.amazon.science/latest-news/the-history-of-amazons-forecasting-algorithm
- Modern approach (learnable) of doing multi-scale analysis.

Seq2Seq: Causal CNN-RNN



Generating Probabilistic Forecasts from Neural Nets

Probabilistic Forecasts from Neural Nets



- How can we get probabilistic forecasts from neural network models?

How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
  - $\Rightarrow$ Conformal predictions [Shafer and Vovk, 2008]

How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
  - $\Rightarrow$ Conformal predictions [Shafer and Vovk, 2008]
- Parametric output distributions, e.g. Gaussian
  - Easy to train by maximizing log-likelihood $\sum_t \log p(z_t)$
  - Strong assumptions about the data
  - Cannot directly capture complex distributions
  - Small number of parameters, e.g. μ and σ for a Gaussian distribution

How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
  - $\Rightarrow$ Conformal predictions [Shafer and Vovk, 2008]
- Parametric output distributions, e.g. Gaussian
  - Easy to train by maximizing log-likelihood $\sum_t \log p(z_t)$
  - Strong assumptions about the data
  - Cannot directly capture complex distributions
  - Small number of parameters, e.g. μ and σ for a Gaussian distribution
- Quantile regression [Koenker and Bassett Jr, 1978]

How to Get Probabilistic Forecasts

- Post-hoc: Generate a point forecast first and then use the residuals to estimate a distribution
  - Simple to implement, but more complex to maintain
  - Distribution not taken into account for estimating the model
  - $\Rightarrow$ Conformal predictions [Shafer and Vovk, 2008]
- Parametric output distributions, e.g. Gaussian
  - Easy to train by maximizing log-likelihood $\sum_t \log p(z_t)$
  - Strong assumptions about the data
  - Cannot directly capture complex distributions
  - Small number of parameters, e.g. μ and σ for a Gaussian distribution
- Quantile regression [Koenker and Bassett Jr, 1978]
- Nonparametric & Semiparametric models
  - No strong assumptions about the data
  - Can capture complex distributions
  - Large number of parameters ($\Longrightarrow$ use a NN)

Two Model Components

Most (if not all) recent deep *probabilistic* forecasting models have two components:

1. A "feature extractor" / "embedder"

$$\mathbf{h}_t = f_\theta(z_{1:t-1})$$

   Examples: RNN (many variants), 1-D (causal) CNNs (e.g. WaveNet, TCN), Transformer, combinations thereof.

2. A probabilistic output model

$$\mathrm{P}(z_t|\mathbf{h}_t) \quad \text{or} \quad \mathrm{P}(z_t, z_{t+1}, \ldots, z_{t+\tau}|\mathbf{h}_t)$$

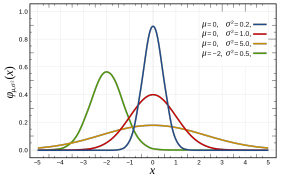
   Examples: (semi-) parametric; quantile regression (& SQF); normalizing flows; energy-based models

How to model $\mathrm{P}(z_t|\mathbf{h}_t)$ – Parametric Distributions

Gaussian

$$\mathrm{P}(z_t|\mathbf{h}_t) = \mathcal{N}(z_t|\mu(\mathbf{h}_t), \sigma^2(\mathbf{h}_t))$$

$$\mu(\mathbf{h}_t) = \mathbf{w}_\mu^\top \mathbf{h}_t \qquad \sigma(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\sigma^\top \mathbf{h}_t)$$

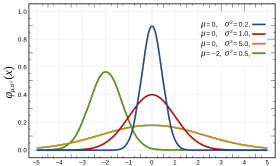


How to model $\mathrm{P}(z_t|\mathbf{h}_t)$ – Parametric Distributions

Gaussian

$$\mathrm{P}(z_t|\mathbf{h}_t) = \mathcal{N}(z_t|\mu(\mathbf{h}_t), \sigma^2(\mathbf{h}_t))$$

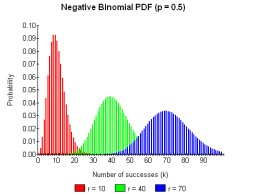
$$\mu(\mathbf{h}_t) = \mathbf{w}_\mu^\top \mathbf{h}_t \qquad \sigma(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\sigma^\top \mathbf{h}_t)$$



Negative Binomial

$$\mathrm{P}(z_t|\mathbf{h}_t) = \mathrm{NB}(z_t|\mu(\mathbf{h}_t), \alpha(\mathbf{h}_t))$$

$$\mu(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\mu^\top \mathbf{h}_t) \qquad \alpha(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\alpha^\top \mathbf{h}_t)$$

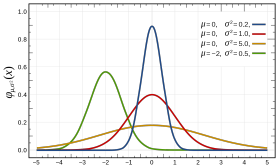


How to model $\mathrm{P}(z_t|\mathbf{h}_t)$ – Parametric Distributions

Gaussian

$$\mathrm{P}(z_t|\mathbf{h}_t) = \mathcal{N}(z_t|\mu(\mathbf{h}_t), \sigma^2(\mathbf{h}_t))$$

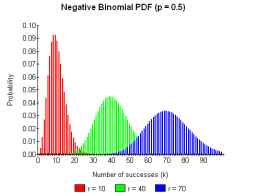
$$\mu(\mathbf{h}_t) = \mathbf{w}_\mu^\top \mathbf{h}_t \qquad \sigma(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\sigma^\top \mathbf{h}_t)$$



Negative Binomial

$$\mathrm{P}(z_t|\mathbf{h}_t) = \mathrm{NB}(z_t|\mu(\mathbf{h}_t), \alpha(\mathbf{h}_t))$$

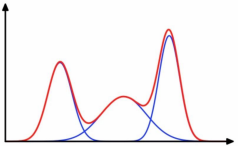
$$\mu(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\mu^\top \mathbf{h}_t) \qquad \alpha(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_\alpha^\top \mathbf{h}_t)$$



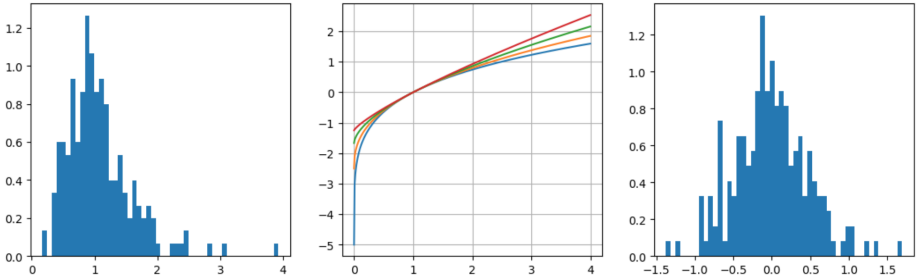
Mixture of Gaussians

$$\mathrm{P}(z_t|\mathbf{h}_t) = \sum_{k=1}^{K} \pi_k \mathcal{N}(z_t|\mu_k(\mathbf{h}_t), \sigma_k^2(\mathbf{h}_t))$$

$$\boldsymbol{\pi} = \mathsf{softmax}(W_\pi \mathbf{h}_t) \qquad \mu_k(\mathbf{h}_t) = \mathbf{w}_{\mu_k}^\top \mathbf{h}_t \qquad \sigma_k(\mathbf{h}_t) = \mathsf{softplus}(\mathbf{w}_{\sigma_k}^\top \mathbf{h}_t)$$



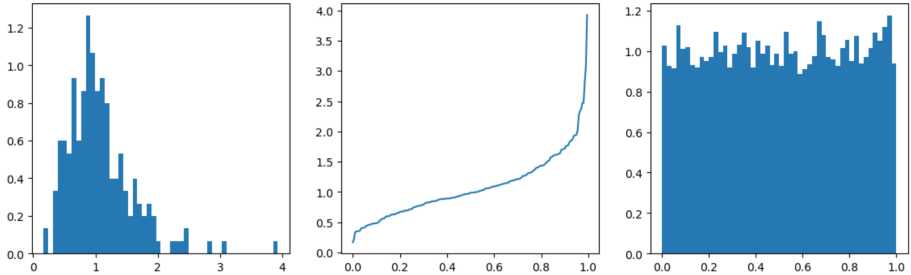
Input/Output Transformations: Box-Cox Transform



- Transform the data to make it closer to a normal distribution

$$y = \frac{x^{\lambda} - 1}{\lambda}$$

Input/Output Transformations: Probabilty Integral Transform



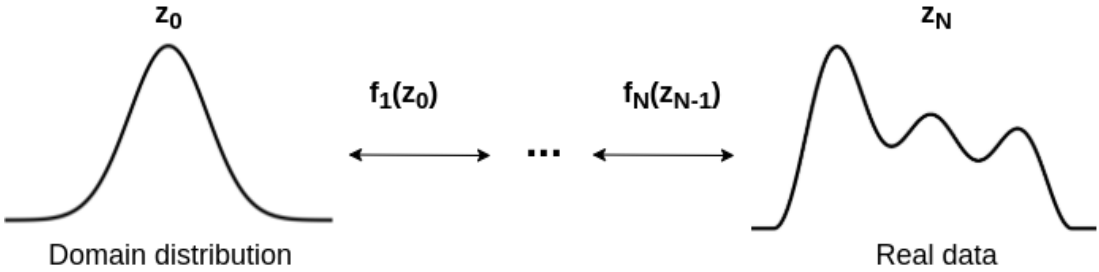
- Estimate the empirical CDF $\hat{F}$ and use to map the data to the quantile domain

$$y = \hat{F}(x)$$

Normalizing Flows

- Normalizing Flow: *invertible*, differentiable transformation

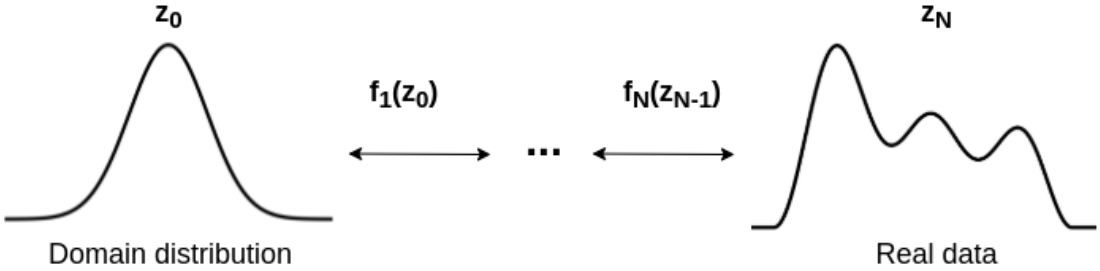
$$\mathrm{NF}(Z_0) = f_N \circ \dots f_1(Z_0)$$



Normalizing Flows

- Normalizing Flow: *invertible*, differentiable transformation

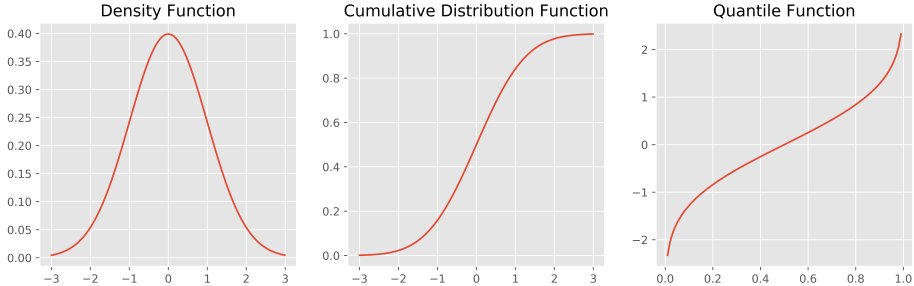
$$\mathrm{NF}(Z_0) = f_N \circ \dots f_1(Z_0)$$



- Transformation of a random variable Z: $Y = f(Z)$ (f is invertible)

$$p_Y(y) = p_Z(f^{-1}(y)) \left|\det[\mathrm{Jac}_y(f^{-1})]\right|$$

Distribution Representations



Quantile Regression [Koenker & Basset, 1978]

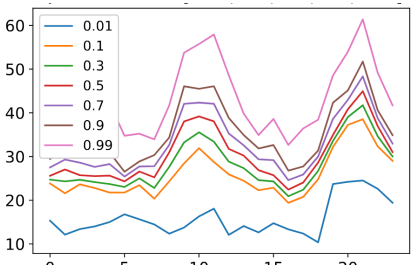
$$q_\alpha = F^{-1}(\alpha) = \text{argmin}_q \mathbb{E}_{F(z)}[\Lambda_\alpha(q, z)]$$
$$\approx \text{argmin}_q \frac{1}{N} \sum_i \Lambda_\alpha(q, z_i),$$

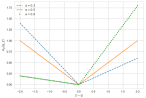
where

$$\Lambda_\alpha(q, z) = \begin{cases} (\alpha - 1)(z - q), & z < q, \\ \alpha(z - q) & z \geq q. \end{cases}$$

is the *quantile loss* (also called *check* or *tick* loss). We parametrize the quantile prediction based on $\mathbf{h}_t$:

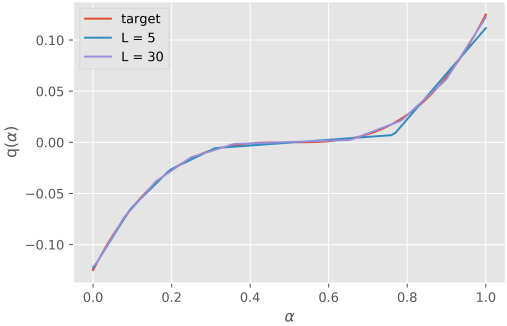
$$\hat{q}_\alpha = \mathbf{w}_\alpha^T \mathbf{h}_t$$





Spline Quantile Function RNN [Gasthaus et al, 2019]

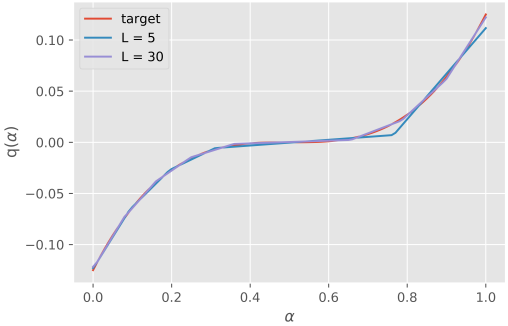
Instead of focussing on a single $\alpha^\star$, can we model the whole quantile function $F^{-1}(\alpha)$?



Expand in function basis, e.g. splines

Spline Quantile Function RNN [Gasthaus et al, 2019]

Isotonic linear splines



$$q(\alpha; \gamma, \mathbf{b}, \mathbf{d}) = \gamma + \sum_{l=1}^{L} (b_l - b_{l-1})(\alpha - d_l)_+$$

$$b_l(\mathbf{h}_t) = \text{softplus}(\mathbf{w}_{b_l}^\top \mathbf{h}_t) \qquad \tilde{\mathbf{d}} = \text{softmax}(W_d \mathbf{h}_t), d_l = \sum_{k=1}^{l} \tilde{d}_k$$

Spline Quantile Function RNN [Gasthaus et al, 2019]

Loss function: CRPS [Matheson & Winkler, 1976]

$$\text{CRPS}(q(\alpha;\theta), z) = \int_0^1 2\Lambda_\alpha(q(\alpha;\theta), z)d\alpha$$

Spline Quantile Function RNN [Gasthaus et al, 2019]

Loss function: CRPS [Matheson & Winkler, 1976]

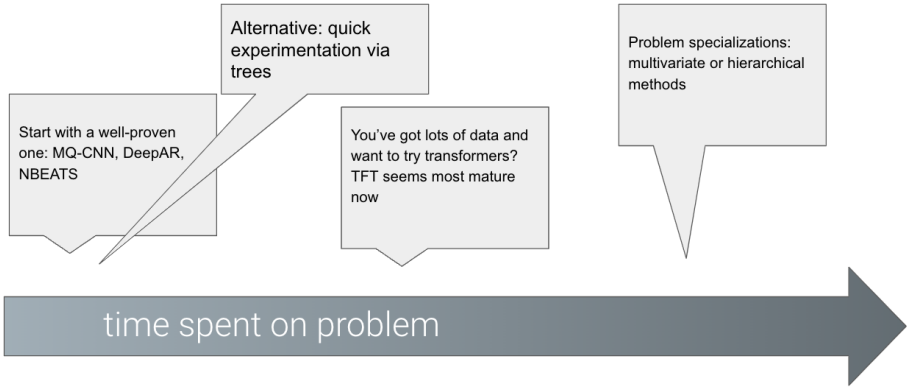
$$\text{CRPS}(q(\alpha;\theta), z) = \int_0^1 2\Lambda_\alpha(q(\alpha;\theta), z)d\alpha$$

For linear splines

$$\int_0^1 2\Lambda_\alpha(q(\alpha;\gamma,\mathbf{b},\mathbf{d}), z)d\alpha = (2\tilde{a}-1)z + (1-2\tilde{a})\gamma + \sum_{l=0}^{L} b_l \left(\frac{1-d_l^3}{3} - d_l - \max(\tilde{a}, d_l)^2 + 2\max(\tilde{a}, d_l)d_l \right)$$

with $\tilde{a}$ satisfying $q(\tilde{a};\gamma,\mathbf{b},\mathbf{d}) = z$.

So ... Neural Networks, huh?



Use them, if

- you want to squeeze out the last bit of performance
- you want to develop your own model flexibly

10 mins break

References

Flunkert, V., Salinas, D., Gasthaus, J., and Januschowski, T. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting, arXiv:1704.04110*.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., and Callot, L. (2019). Criteria for classifying forecasting methods. *International Journal of Forecasting*.

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.

Mukherjee, S., Shankar, D., Ghosh, A., Tathawadekar, N., Kompalli, P., Sarawagi, S., and Chaudhury, K. (2018). Armdn: Associative and recurrent mixture density networks for eretail demand forecasting. *arXiv preprint arXiv:1803.03800*.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.

Shafer, G. and Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *SSW*, page 125.

Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. *NIPS Workshop on Time Series, arXiv:1711.11053*.

References (cont.)

Widrow, B. (1987). The original adaptive neural net bloom-balancer. In *IEEE Int. Symp. Circuits Syst.*, volume 2, pages 351–357.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., and Liu, Y. (2017). Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 777–785. SIAM.