Forecasting Big Time Series: Old and New

Tutorial for VLDB, 2018

Christos Faloutsos, Jan Gasthaus, **Tim Januschowski, Yuyang Wang**

CMU/**Amazon Research** and **AWS AI Labs**




https://lovvge.github.io/Forecasting-Tutorial-VLDB-2018/

August 28th, 2018

ML Forecasting Team@AWS AI Labs


Lorenzo Stella


Matthias Seeger


Jan Gasthaus


Bernie Wang


David Salinas


Tim Januschowski


Syama Rangapuram


Valentin Flunkert


Alexander Alexandrov


Michael Bohlke-Schneider

amazon.de

Amazon AI

Forecasting Big Time Series: Old and New

1. Introduction to Forecasting
2. Classical approaches (local, learning one time series at a time)
3. Modern approaches (globally finding patterns)
4. Building forecasting systems

Out of Scope, but we still want to point it out ...

Pattern finding, outlier/anomaly detection, modeling, forecasting and similarity indexing are closely related:

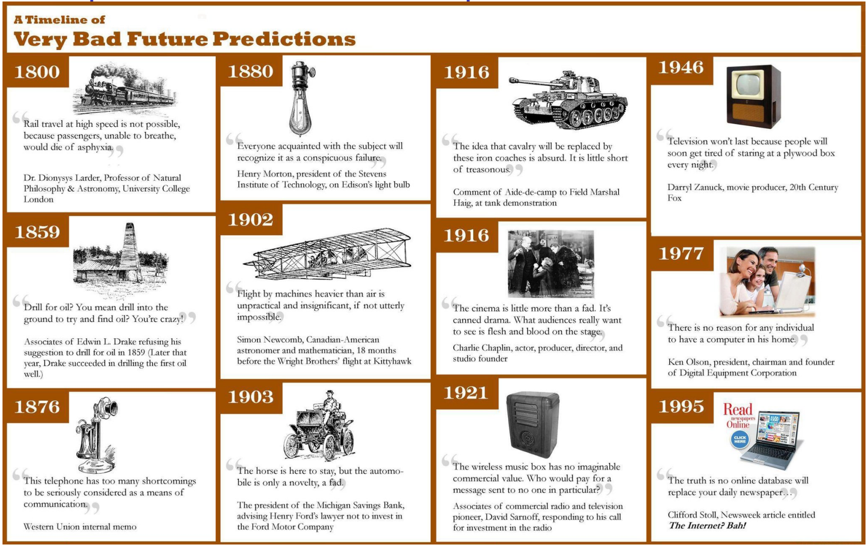
- For forecasting, we need to estimate
  - patterns/rules/models
  - similar past settings
- For outlier/anomaly detection, we can use forecasts
  - outlier = too far from our forecast

Reference



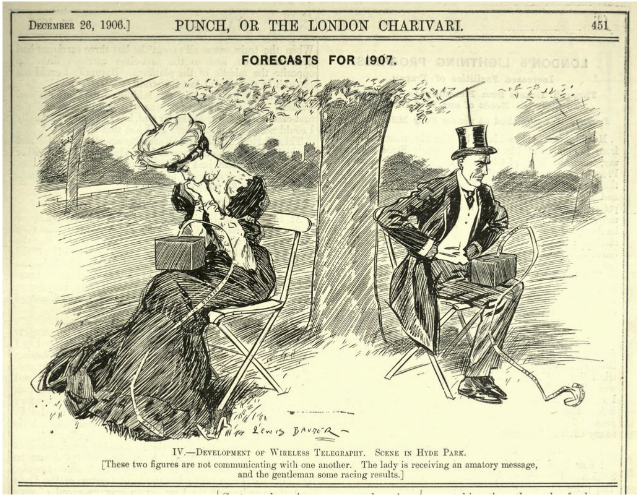
Smart Analytics for Big Time-series Data
Yasushi Sakurai, Yasuko Matsubara and Christos Faloutsos
http://www.cs.kumamoto-u.ac.jp/~yasukoTALKS/17-KDD-tut/

Within the Scope, but we still want to point it out ...



http://infographic.city/timeline-bad-future-predictions/

Within the Scope, but we still want to point it out ...



Forecast for 1907:
Telegraph kills live communication!

First Principle

Forecasts are always wrong.

https://medium.freecodecamp.org/worst-tech-predictions-of-the-past-100-years-c18654211375

Within the Scope, but we still want to point it out ...

Prediction is very difficult, especially about the future. – Niels Bohr

Within the Scope, but we still want to point it out ...

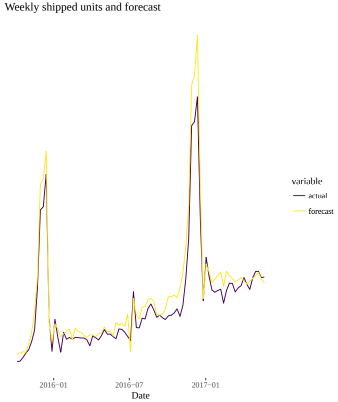
Prediction is very difficult, especially about the future. – Niels Bohr

Second Principle

Whenever possible, formulate the problem in a different way.

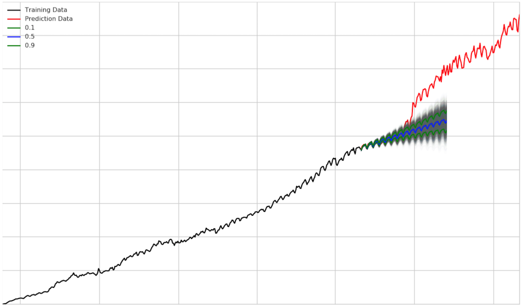
Introduction to Forecasting: Old and New

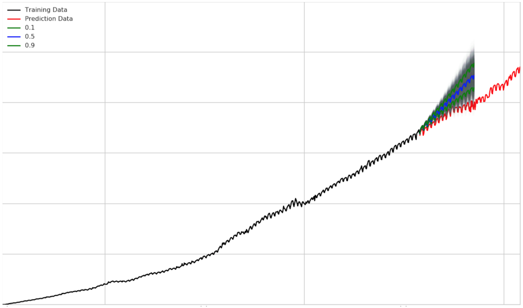
Forecasting Problems at Amazon I: Retail Demand



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

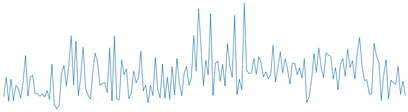
Forecasting Problems at Amazon II: AWS Compute Capacity

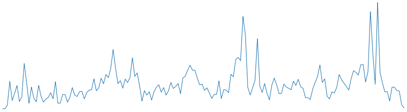




- Problem: predict AWS compute capacity demand
- Decision Problem: how many servers to order when and where

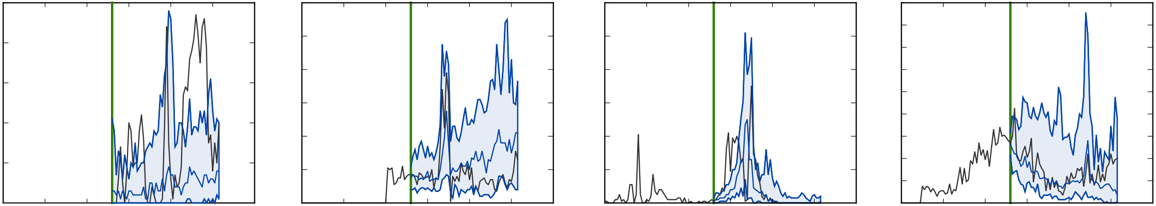
Forecasting Problems at Amazon III: Staff Planning





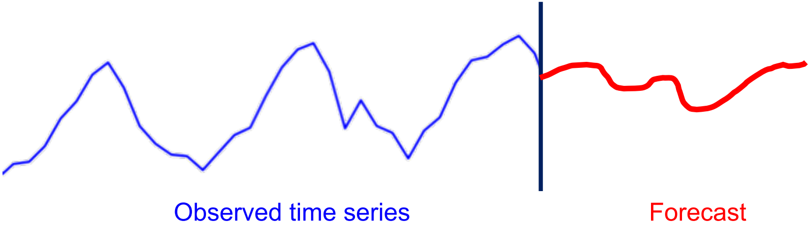
- Problem: predict attendance rate of fulfillment center staff
- Decision Problems: how to schedule staff and when to hire how much staff

Forecasting Problems at Amazon IV: Retail Product Forecasting



- 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

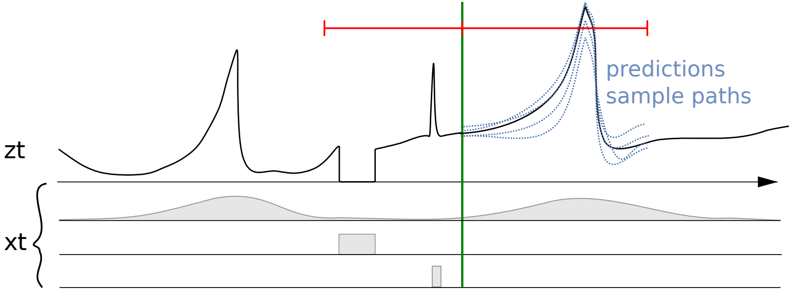
What is Forecasting?



Australian beer shipments: weekly time series (07 Jan 1970 to 05 Dec 1973)

http://www.broo.com.au/australian-beers

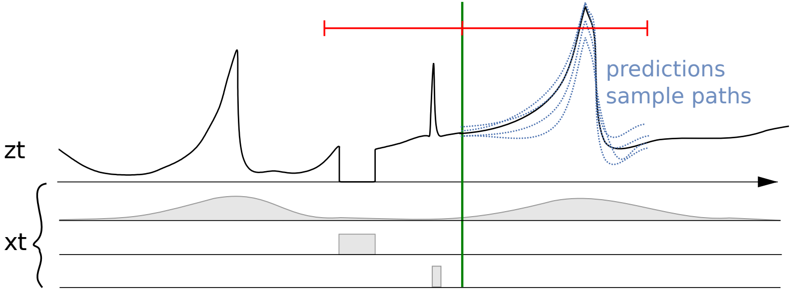
Forecasting Problems: General Setup



- 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

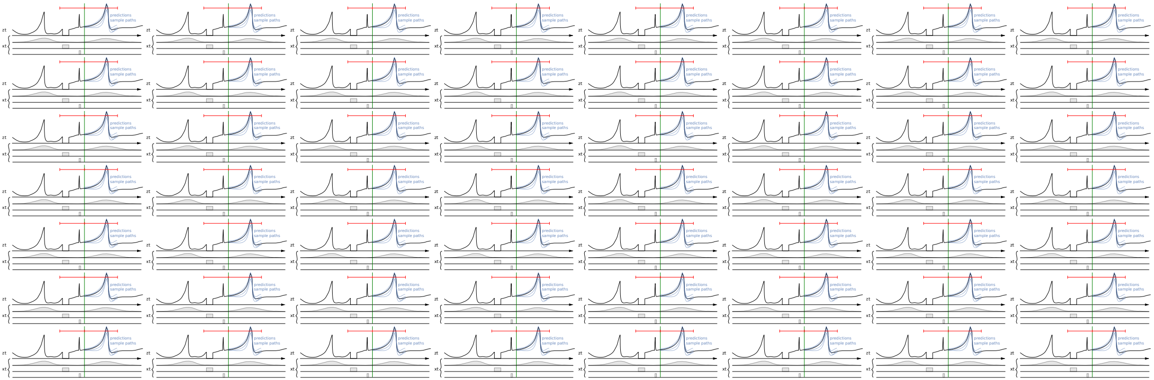
$$\text{best action} = \underset{a}{\operatorname{argmin}} \, \mathrm{E}_{\mathrm{P}}[\text{cost}(a, z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h})]$$

Forecasting Problems: Old and New

- small number of time series
- sufficient historical data
- limited meta data
- hand-crafted models
- statistician and econometrician heavy

We refer to these problems as strategic forecasting problems.

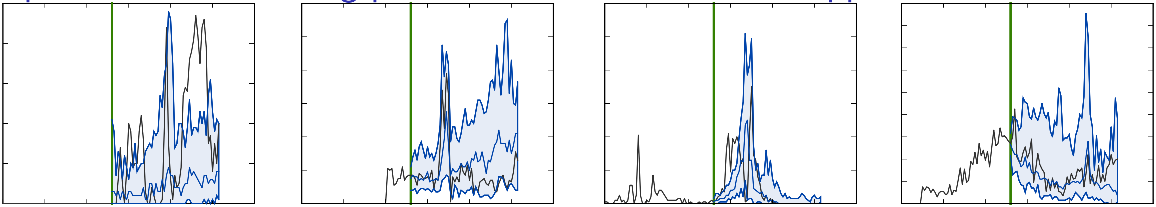
Forecasting Problems: Old and New



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

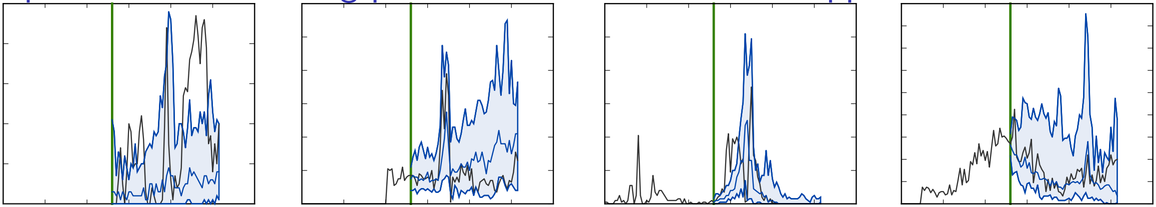
We refer to these problems as operational forecasting problems.

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

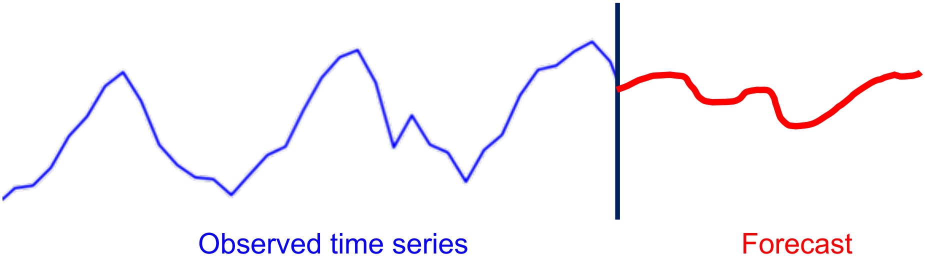


- cold start/new items
- short cycles
- burstiness, sparsity
- high degree of automation of downstream systems
- ratio of people/time series $\ll 1$

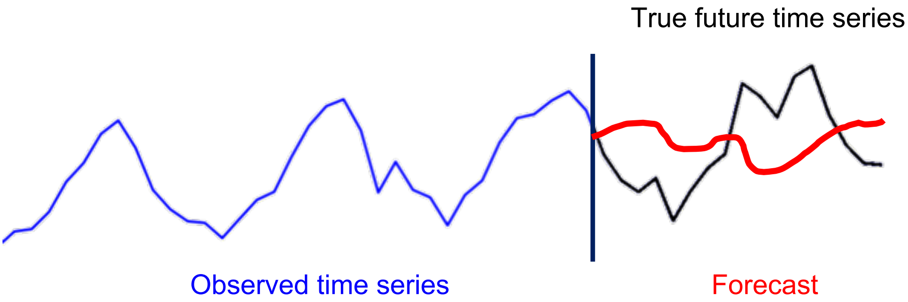
Two extremes (to be covered in Part 5):

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

Metrics to Evaluate Point Forecast



Metrics to Evaluate Point Forecast



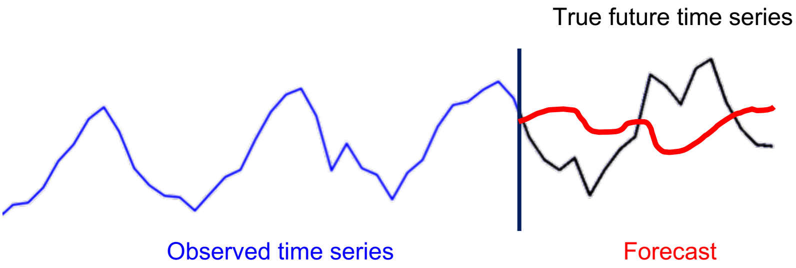
$$\text{Absolute Error}: \; e_t = |z_t - \widehat{z}_t|$$

- Mean Absolute Error (MAE): $\text{mean}(e_t)$ (over forecast horizon h)
- Mean Absolute Percentage Error (MAPE): $\frac{1}{h} \sum_t e_t / |z_t|$
- Root Mean Square Error (RMSE): $\sqrt{\text{mean}(e_t^2)}$

From Point Forecasts to Probabilistic Forecasts

To paraphrase George E. P. Box

All forecasts are wrong, but some are useful ...



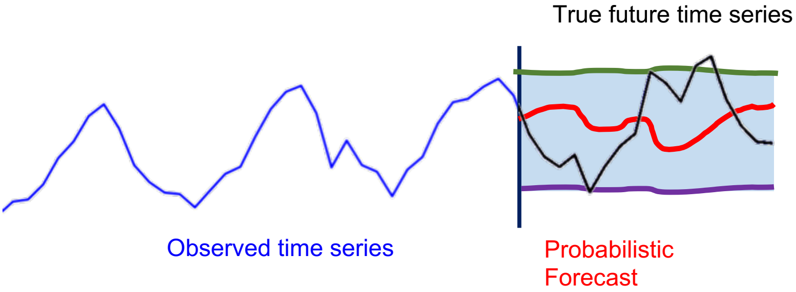
Point forecasts are typically insufficient for decision making.

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

From Point Forecasts to Probabilistic Forecasts

To paraphrase George E. P. Box

All forecasts are wrong, but some are useful ...



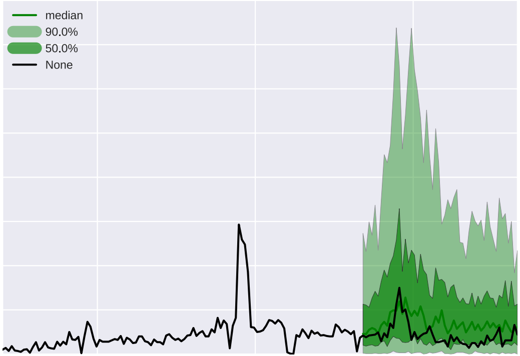
Point forecasts are typically insufficient for decision making.

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

From Point Forecasts to Probabilistic Forecasts

Probabilistic Forecast

What is the distribution of the future time series values $P(\widehat{z}_t)$?

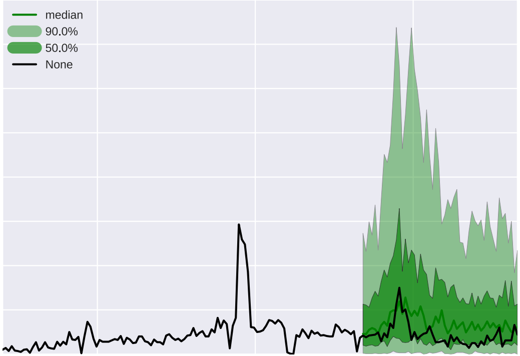


- P50 Forecast: 50% of the time $z_t \leq \widehat{z}_t$
- P90 Forecast: 90% of the time $z_t \leq \widehat{z}_t$

Evaluating Probabilistic Forecast

Probabilistic Forecast

P90 forecast means 90% of the time the true value $z_t \leq \hat{z}_t$ (or precisely $\hat{z}_t^{0.9}$)



Quantile Loss

$$e_t^q = \begin{cases} q \cdot (z_t - \hat{z}_t^q), & z_t \geqslant \hat{z}_t^q \\ (1-q) \cdot (\hat{z}_t^q - z_t), & \hat{z}_t^q \geqslant z_t \end{cases}$$

$\hat{z}_t^q$: Forecast at quantile q

Hit Rate / Calibration

Percentage of $z_t \leqslant \hat{z}_t^q$

Quantile Loss and Calibration: Two Sides of the Coin

Quantiles Loss

P90 forecast means 90% of the time the true value z_t is smaller than the forecast.

$$e_t = \begin{cases} 0.9 \cdot (z_t - \widehat{z}_t), & z_t \geqslant \widehat{z}_t \\ 0.1 \cdot (\widehat{z}_t - z_t), & \widehat{z}_t \geqslant z_t \end{cases}$$

Quantile loss is **proper scoring rule**, which means optimizing this metric leads to an accurate P90 forecast.

Quantile Loss and Calibration: Two Sides of the Coin

Calibration

P90 forecast means 90% of the time the true value z_t is smaller than the forecast.

$$(\text{Percentage of } z_t \leqslant \hat{z}_t) \approx 90\%$$

Calibration alone is not enough ...

True target time series	5, 5, 5, 5, 5, ...
P50 Forecast	0, 100, 0, 100, 0, 100, ...

Perfectly calibrated (calibration = 50%), but horrible forecast ...

Probabilistic Forecast

Goal

The goal of probabilistic forecasting is to maximize the sharpness of the predictive distribution subject to calibration. [Gneiting et al., 2007]

- Sharpness: the width of the predictive intervals
- The forecast distribution should be as close as possible to the true distribution

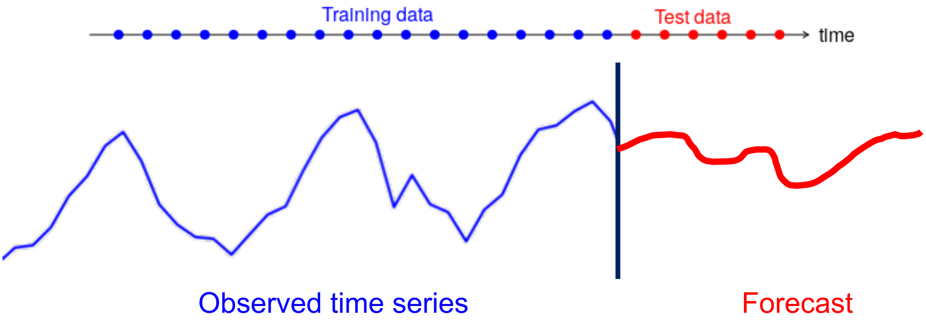
Continuous Ranked Probability Score (CRPS)

$$\text{CRPS} = \int_0^1 \text{QUANTILE LOSS}(q)\mathrm{d}q$$

Optimizing CRPS leads to sharp and calibrated forecast.

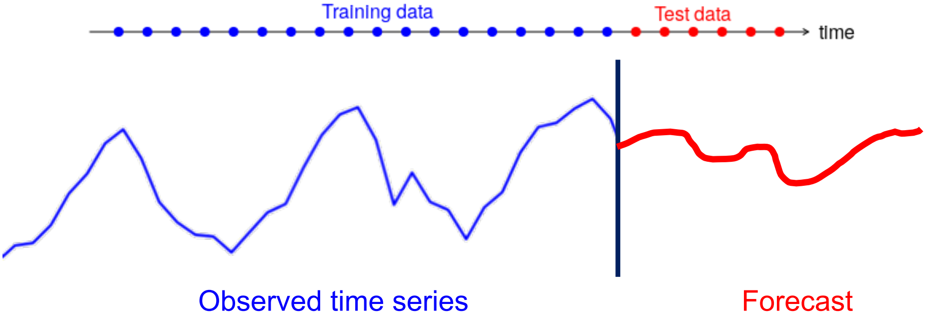
Evaluating Forecasts: Backtesting

Compute forecast accuracy on observed data for a fixed time series i:



Evaluating Forecasts: Backtesting

Compute forecast accuracy on observed data for a fixed time series i:



Accuracy now depends on the start of test data.

Evaluate forecast: better backtesting

Compute forecast accuracy (to be defined) on observed data for a fixed time series i:

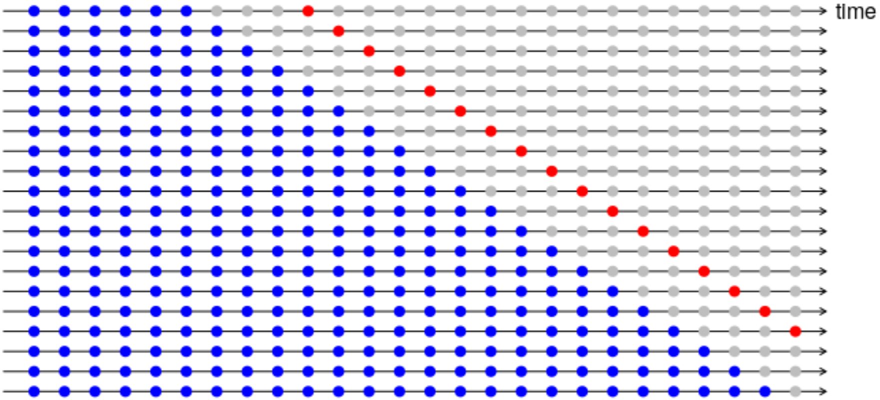


Figure: [Hyndman and Athanasopoulos, 2017]

Some Remarks on Accuracy

Alternative to MAPE, we can introduce scale-independent accuracy by scaling using

- the error of a benchmark/standard method (e.g., MRAE)
- the forecast accuracy of a benchmark method (e.g., RelMAE)
- in-sample random walk (e.g., MASE)
- evaluating spatio-temporal forecast with Optimal Transport [Roberts et al., 2017]

More in [Hyndman and Koehler, 2006; Kolassa and Schuetz, 2007]

Some Remarks on Accuracy

Alternative to MAPE, we can introduce scale-independent accuracy by scaling using

- the error of a benchmark/standard method (e.g., MRAE)
- the forecast accuracy of a benchmark method (e.g., RelMAE)
- in-sample random walk (e.g., MASE)
- evaluating spatio-temporal forecast with Optimal Transport [Roberts et al., 2017]

More in [Hyndman and Koehler, 2006; Kolassa and Schuetz, 2007]

Practical subtleties:

- how to evaluate given missing values (e.g., out-of-stock situations, errors in recording of events)?
- deal with $\cdot/0$ and $0/0$

Some Remarks on Accuracy

Potentially three different accuracy measures:

- loss-function for training the model
- forecast accuracy metric for backtesting
- forecast accuracy measure for reporting to stakeholders

Metric needs to be simple, intuitive. Insist on it being a proper score.

Some Remarks on Accuracy

Potentially three different accuracy measures:

- loss-function for training the model
- forecast accuracy metric for backtesting
- forecast accuracy measure for reporting to stakeholders

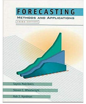
Metric needs to be simple, intuitive. Insist on it being a proper score.

Crucial to understand the down-stream consequences of the forecasts:

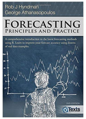
$$\text{best action} = \operatorname*{argmin}_a \mathrm{E}_{\mathrm{P}}[\text{cost}(a, z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h})]$$

More accurate forecasts may not lead to better downstream decisions.

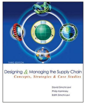
Selected References

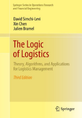


[Makridakis et al., 1998]. Classic introductory book.



[Hyndman and Athanasopoulos, 2017]. Recent introductory book.





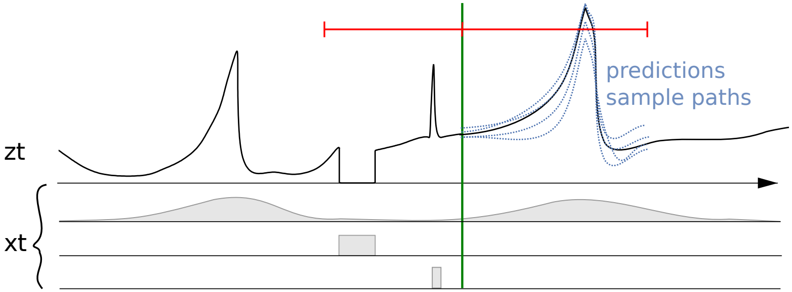
[Larson et al., 2001; Simchi-Levi et al., 2013]. Introduction to practical Supply Chain problems including forecasting.

FORESIGHT THE INTERNATIONAL JOURNAL OF APPLIED FORECASTING

General business forecasting journal:
https://foresight.forecasters.org/

Classical Methods for Forecasting: Old and New

Forecasting Problems: General Setup



- Predict the future behavior of a time series z_i for item $i \in I$ given its past

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

- Make optimal decisions

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

We drop the item index i whenever the context is clear.

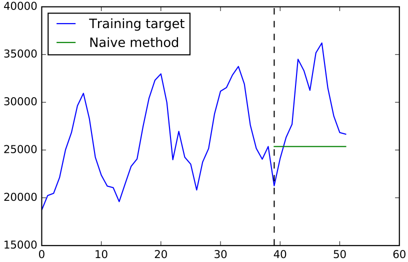
Simple Forecasting Methods

- Input: a sequence of observations $z_1, z_2, \ldots, z_T$
- Output: forecasts for all time steps in forecast horizon h: $T + 1, T + 2, \ldots, T + h$

Simple Forecasting Methods

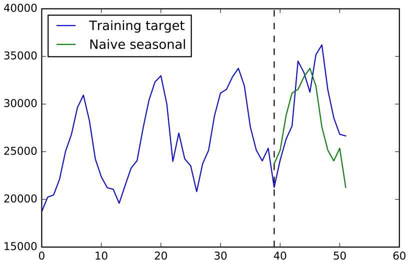
- Input: a sequence of observations $z_1, z_2, \ldots, z_T$
- Output: forecasts for all time steps in forecast horizon h: $T+1, T+2, \ldots, T+h$
- Naive method: future forecasts are equal to the last observed value.

$$z_{T+t} = z_T, \quad t = 1, 2, \ldots, h$$



Simple Forecasting Methods (contd.)

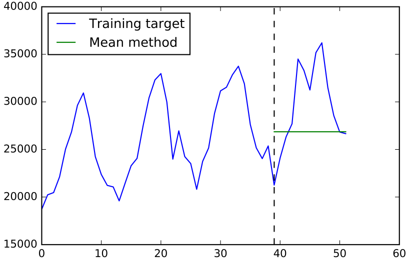
- Naive seasonal method: future forecasts are equal to the observed value from last season.



Simple Forecasting Methods (contd.)

- Mean method: future forecasts are equal to the average of all observed values.

$$z_{T+t} = \frac{1}{T}(z_1 + z_2 + \ldots + z_T), \quad t = 1, 2, \ldots, h$$

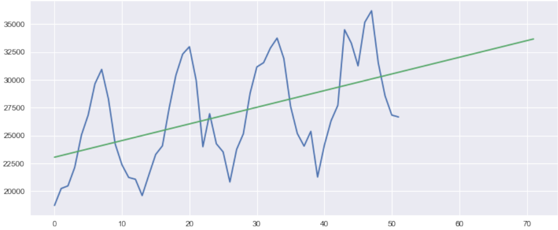


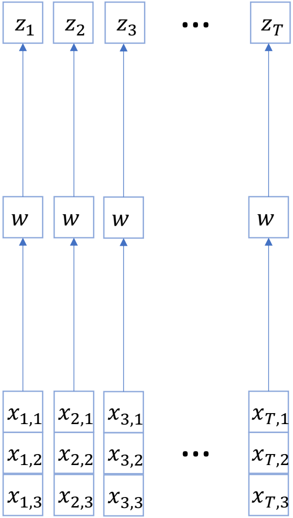
Forecasting with Linear Regression

- Assume our prediction $\widehat{z}_t$ is a weighted combination of *features* $x_{t,1}, \ldots, x_{t,D}$,

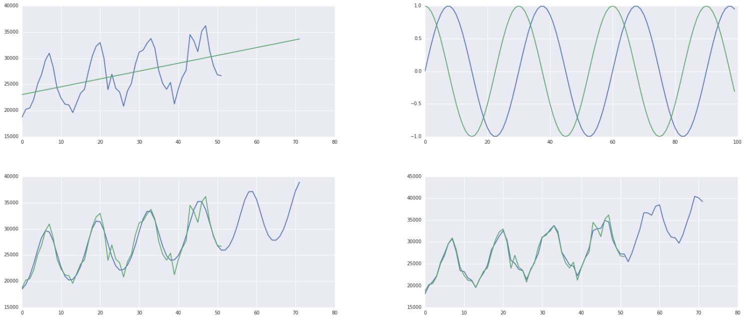
$$\widehat{z}_t = \sum_{d=1}^{D} w_d x_{t,d}$$

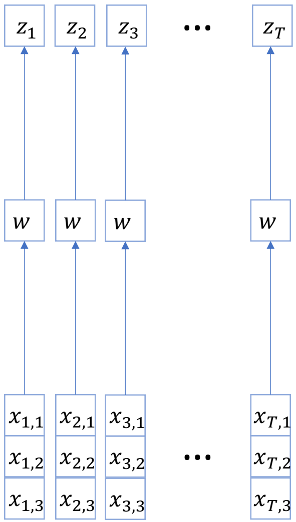
- The features $x_{t,d}$ are assumed to be given (= hand designed)





Forecasting with Linear Regression





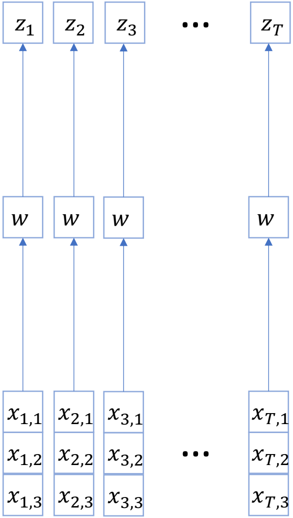
Forecasting with Linear Regression

- Goal: find weights $w_1, \ldots, w_D$ so that our prediction $\widehat{z}_t$ is close to the true z_t.
- Least-squares finds an optimal $w^\star$ by minimizing

$$w^\star = \underset{w}{\operatorname{argmin}} \sum_{t=1}^{T} (z_t - \widehat{z}_t)^2 = \sum_{t=1}^{T} \left(z_t - \sum_{d=1}^{D} w_d x_{t,d} \right)^2$$

- $w^\star$ can then be used to make forecasts:

$$z_t = \sum_{d=1}^{D} w_d^\star x_{t,d} \qquad t = T+1, \ldots, T+h$$



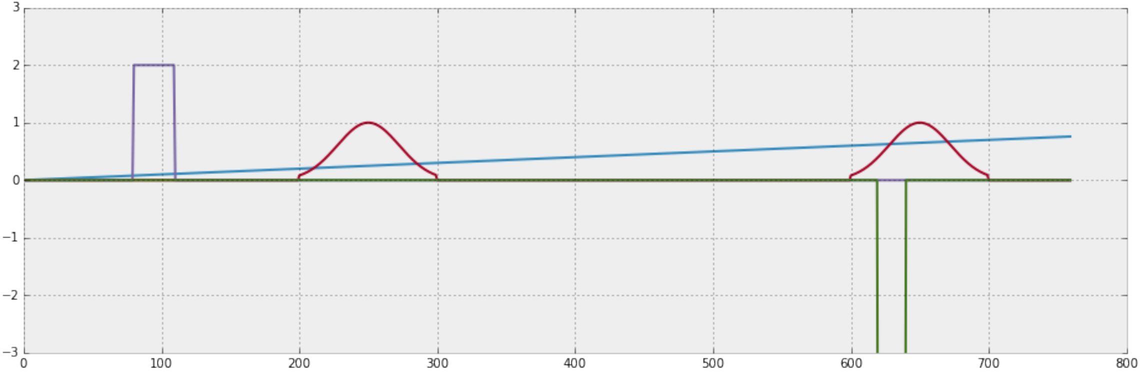
Features for Linear Regression

The features used in such a linear model are themselves time series $x_{1,d}, x_{2,d}, \ldots, x_{T+H,d}$.

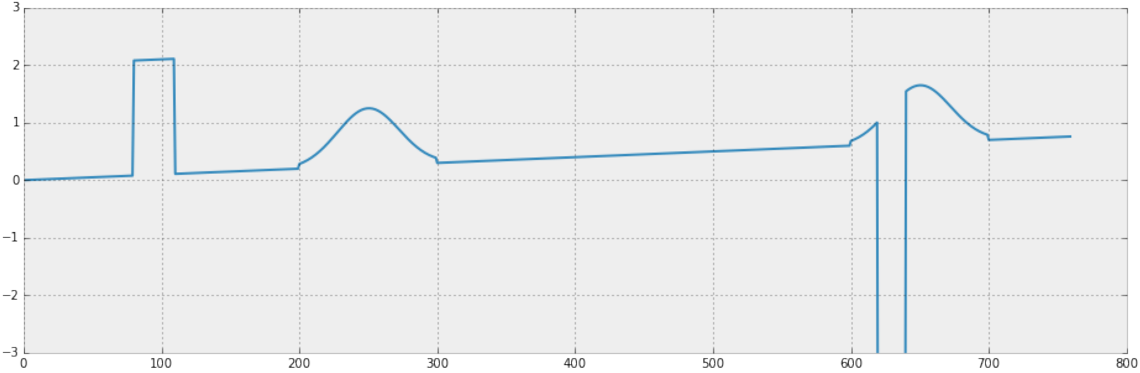
1. Trend features (linear, logarithmic, exponential, logistic, etc.)
2. Seasonal features: dummies (one-hot indicators), periodic features (e.g. Fourier, wavelet, etc.)
3. Lagged target values (e.g. use z_{t-1} and z_{t-2} as features to predict z_t)
4. Seasonal lagged target values (e.g. use z_{t-S} to predict z_t, with $S = 12$ for monthly data)
5. (Weighted) average features (e.g. $\text{mean}(z_{t-7:t-1})$)

Note that (4) and (5) are just special cases of (3). However, using features of type (4) or (5) designed using domain knowledge (e.g. seasonality) one can achieve the same effect with less parameters.

Examples



Examples

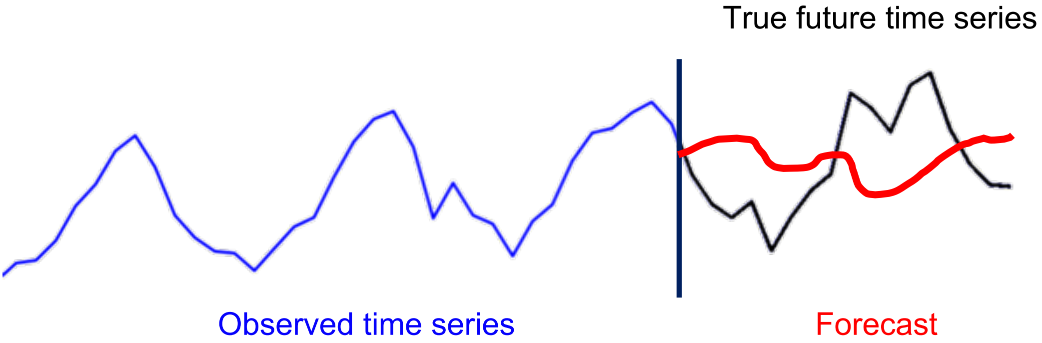


From Algorithm to Probabilistic Model

It is far better to foresee even without certainty than not to foresee at all. – Henri Poincaré

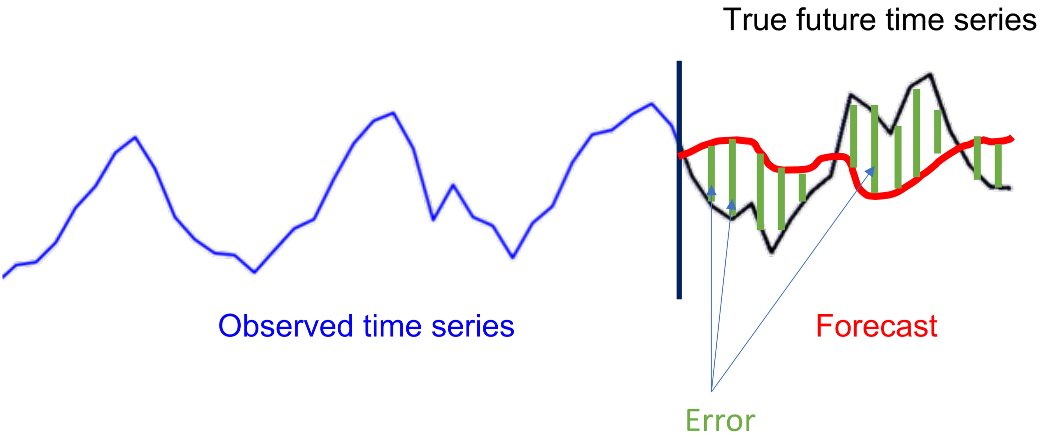
From Algorithm to Probabilistic Model

$$\min \sum_t e_t, \text{ with } e_t = (\widehat{z}_t - z_t)^2 \Longleftrightarrow \max \prod_t P(e_t), \text{ with } e_t \sim \mathcal{N}(e_t|0, \sigma^2)$$



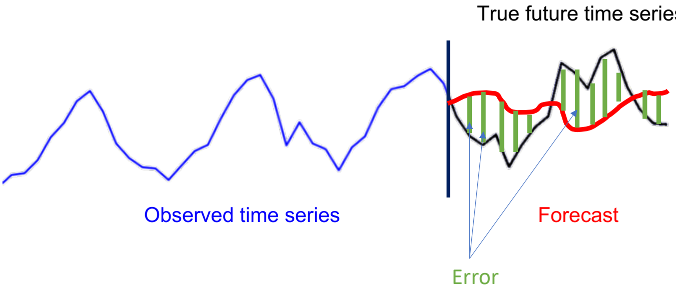
From Algorithm to Probabilistic Model

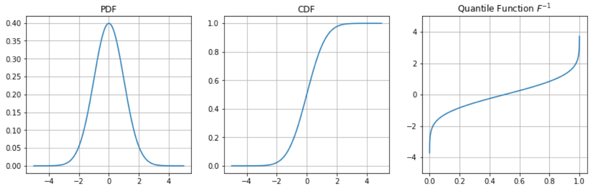
$$\min \sum_t e_t, \text{ with } e_t = (\widehat{z}_t - z_t)^2 \Longleftrightarrow \max \prod_t P(e_t), \text{ with } e_t \sim \mathcal{N}(e_t|0, \sigma^2)$$



From Algorithm to Probabilistic Model

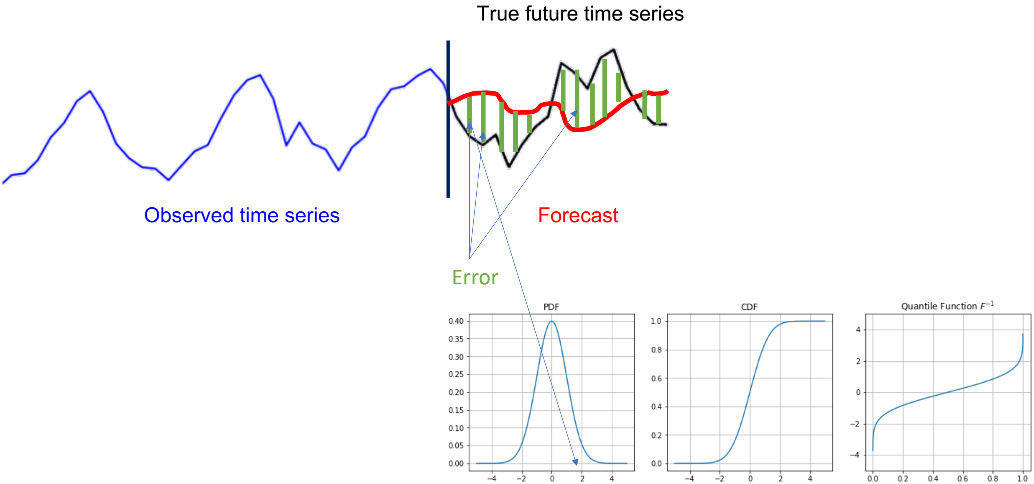
$$\min \sum_t e_t, \text{ with } e_t = (\widehat{z}_t - z_t)^2 \iff \max \prod_t P(e_t), \text{ with } e_t \sim \mathcal{N}(e_t|0, \sigma^2)$$





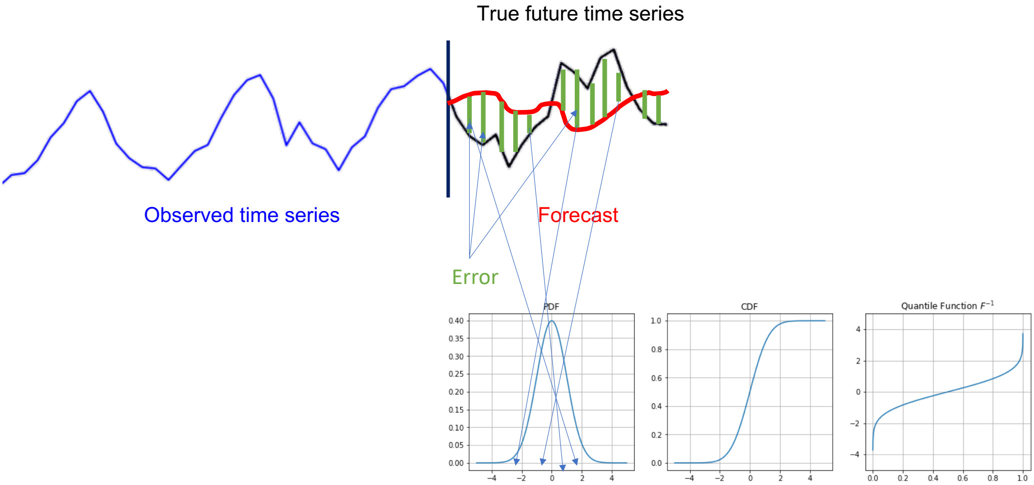
From Algorithm to Probabilistic Model

$$\min \sum_t e_t, \text{ with } e_t = (\widehat{z}_t - z_t)^2 \iff \max \prod_t P(e_t), \text{ with } e_t \sim \mathcal{N}(e_t|0,\sigma^2)$$



From Algorithm to Probabilistic Model

$$\min \sum_t e_t, \text{ with } e_t = (\widehat{z}_t - z_t)^2 \iff \max \prod_t P(e_t), \text{ with } e_t \sim \mathcal{N}(e_t|0, \sigma^2)$$



Linear Regression as a Probabilistic Model

- Assume the following *model* for the data, introducing a *latent* (unobserved) variable y_t

$$y_t = \sum_{d=1}^{D} w_d x_{t,d}$$

$$z_t = y_t + \epsilon_t \qquad\qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- Equivalently, $P(z_t|y_t) = \mathcal{N}(z_t|y_t, \sigma^2)$, so that

$$P(z_t|y_t) = \mathcal{N}(z_t|y_t, \sigma^2) = \mathcal{N}(e_t|0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{e_t^2}{2\sigma^2}\right\}$$

- In terms of the features $x_{t,d}$ we have

$$P(z_t|x_t) = \mathcal{N}\left(z_t \middle| \sum_d w_d x_{t,d}, \sigma^2\right)$$

Linear Regression as a Probabilistic Model

- Finding the least square solution $w^\star \implies$ maximum likelihood estimation (MLE)

$$
\begin{aligned}
w^\star &= \underset{w}{\operatorname{argmax}} \prod_{t=1}^{T} P(z_t|x_t) = \underset{w}{\operatorname{argmax}} \sum_{t=1}^{T} \log P(z_t|x_t) \\
&= \underset{w}{\operatorname{argmax}} \sum_{t=1}^{T} \left(-\log\sqrt{2\pi\sigma^2} - \left(z_t - \sum_d w_d x_{t,d} \right)^2 / (2\sigma^2) \right) \\
&= \underset{w}{\operatorname{argmax}} \sum_{t=1}^{T} - \underbrace{\left(z_t - \sum_d w_d x_{t,d} \right)^2}_{e_t^2} \\
&= \underset{w}{\operatorname{argmin}} \sum_{t=1}^{T} e_t^2
\end{aligned}
$$

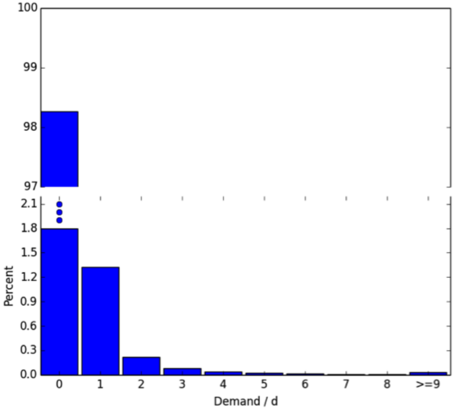
- We can also find the optimal σ^2 in the same way

Generalized Linear Models (GLM)

- The Gaussian noise *assumption* $P(z_t|y_t) = \mathcal{N}(z_t|y_t, \sigma^2)$ is a modelling *choice*!
- Other choices for $P(z_t|y_t)$ (observation model, also called *likelihood* in Bayesian models) are possible
- Common choices are:
  - Poisson, Negative-Binomial (count data)
  - Beta (data in $(0, 1)$ interval)
  - Bernoulli (binary data)
  - Student-t (heavy-tailed real data)

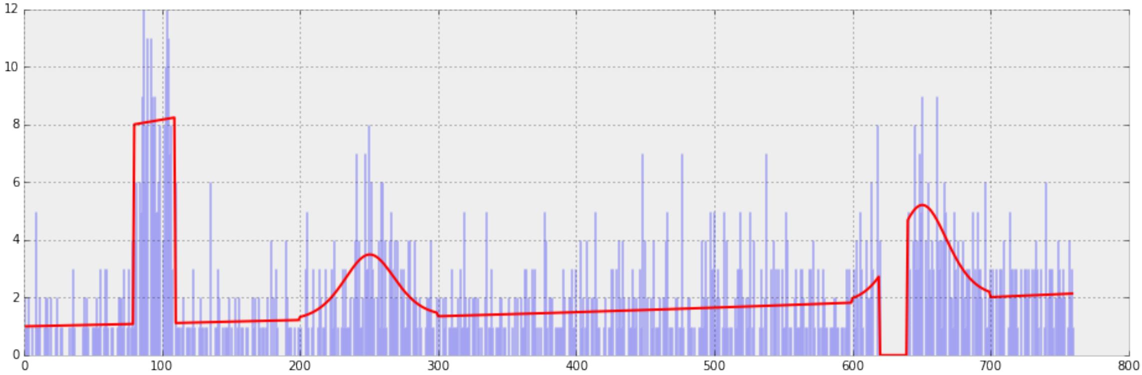
Latent Linear Models: Likelihood $P(z_t|y_t)$

- In daily real-world sales data, 98% of the values are 0



- Could use (not a good fit): Gaussian, Gamma, Poisson, Negative Binomial (NB)
- Better choices: Zero-inflated Poisson/NB, multi-stage likelihood [Seeger et al., 2016]

Examples



Features for Linear Regression

The features used in such a linear model are themselves time series $x_{1,d}, x_{2,d}, \ldots, x_{T+H,d}$.

1. Trend features (linear, logarithmic, exponential, logistic, etc.)
2. Seasonal features: dummies (one-hot indicators), periodic features (e.g. Fourier, wavelet, etc.)
3. Lagged target values (e.g. use z_{t-1} and z_{t-2} as features to predict z_t)
4. Seasonal lagged target values (e.g. use z_{t-S} to predict z_t, with $S = 12$ for monthly data)
5. (Weighted) average features (e.g. $\text{mean}(z_{t-7:t-1})$)

Note that (4) and (5) are just special cases of (3). However, using features of type (4) or (5) designed using domain knowledge (e.g. seasonality) one can achieve the same effect with less parameters.

Fewer parameters $\Longrightarrow$ less training data needed! Less prone to overfitting!

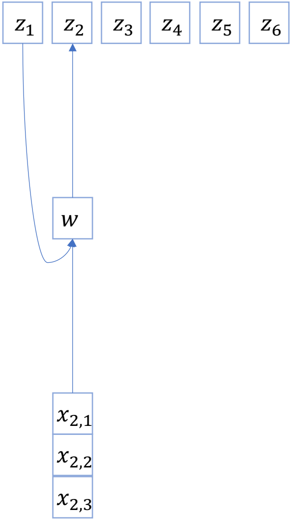
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



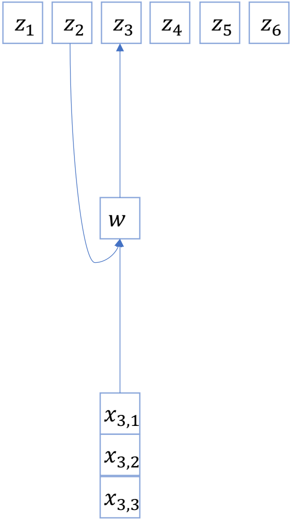
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



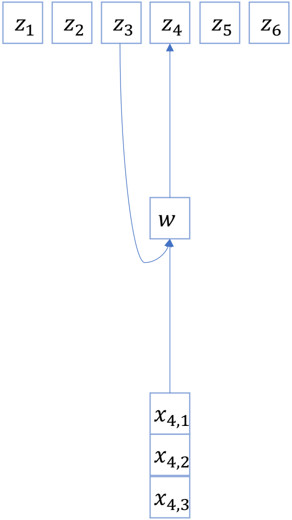
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



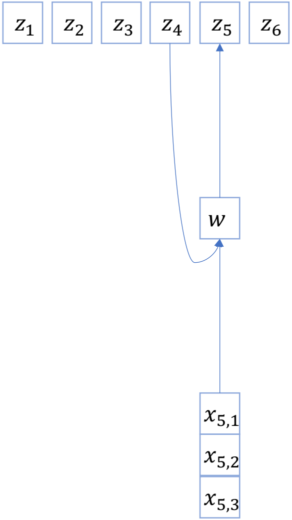
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



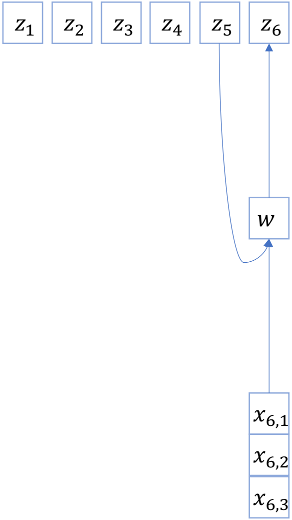
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



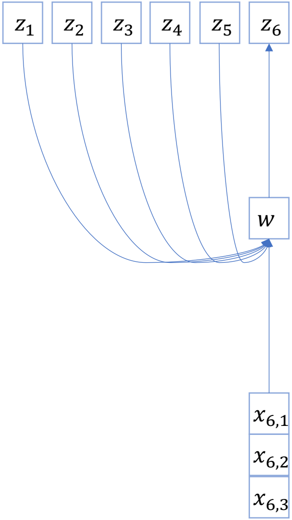
Lagged Target Values / Autoregressive Model

Idea: Use lagged values z_{t-l} (for $l = 1, 2, 3, \ldots, p$) as features to predict z_t

- In a linear model we have $z_t = \sum_d w_d x_{t,d} + b + \epsilon_t$
- By using lagged targets as features, i.e. $x_{t,d} = z_{t-d}$ for $d = 1, 2, \ldots, p$ we get

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

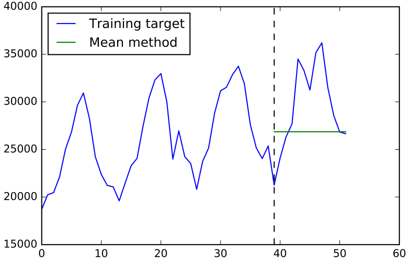
- Each value is modelled as a weighed average of previous values, plus noise
- The noise is usually assume to be iid. Gaussian, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$
- This is the so-called **Autoregressive Model (AR)** of order p!



Simple Forecasting Methods: Mean method

- Mean method: future forecasts are equal to the average of all observed values.

$$z_{T+t} = \frac{1}{T}(z_1 + z_2 + \ldots + z_T), \quad t = 1, 2, \ldots, h$$



Alternative to linear regression

Prediction is the weighted average of all observations

$$\widehat{z}_2 = \alpha z_1 + (1-\alpha)\widehat{z}_1$$
$$\widehat{z}_3 = \alpha z_2 + \alpha(1-\alpha)z_1 + (1-\alpha)\widehat{z}_1$$
$$\ldots$$

Alternative to linear regression

Prediction is the weighted average of all observations

$$\widehat{z}_2 = \alpha z_1 + (1-\alpha)\widehat{z}_1$$
$$\widehat{z}_3 = \alpha z_2 + \alpha(1-\alpha)z_1 + (1-\alpha)\widehat{z}_1$$
$$\ldots$$

$\widehat{z}_1$ and α are adjustable parameters

Alternative to linear regression

Prediction is the weighted average of all observations

$$\widehat{z}_2 = \alpha z_1 + (1-\alpha)\widehat{z}_1$$
$$\widehat{z}_3 = \alpha z_2 + \alpha(1-\alpha)z_1 + (1-\alpha)\widehat{z}_1$$
$$\ldots$$

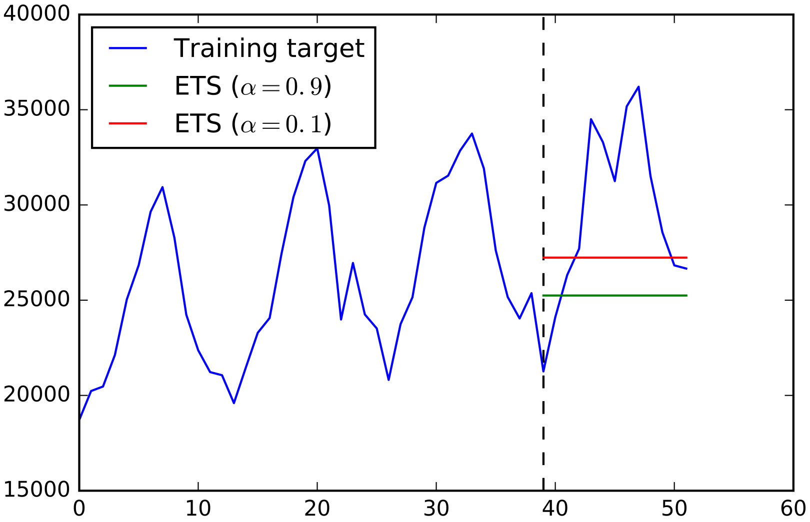
$\widehat{z}_1$ and α are adjustable parameters

- ETS (Simple ExponenTial Smoothing): weighted average of all observations:

$$\widehat{z}_{T+h} = \alpha z_T + \alpha(1-\alpha)z_{T-1} + \alpha(1-\alpha)^2 z_{T-2} + \cdots + (1-\alpha)^T \widehat{z}_1$$

  α is a smoothing parameter and $\widehat{z}$ is the prediction for $t = 1$.

ETS (contd.)



Training target
ETS ($\alpha = 0.9$)
ETS ($\alpha = 0.1$)
40000
35000
30000
25000
20000
15000
0
10
20
30
40
50
60

Simple Exponential Smoothing

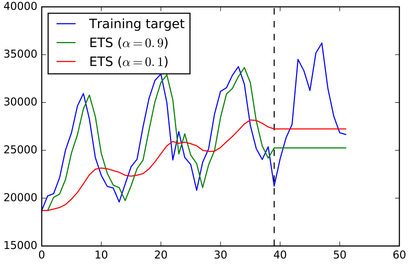
- Forecast at time t is adjusted to previous error

$$\widehat{z}_{t+1} = \underbrace{\widehat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \widehat{z}_t)}_{\text{error in previous forecast}},$$

Simple Exponential Smoothing

- Forecast at time t is adjusted to previous error

$$\widehat{z}_{t+1} = \underbrace{\widehat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \widehat{z}_t)}_{\text{error in previous forecast}},$$

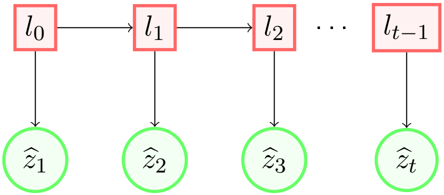


Simple Exponential Smoothing (contd.)

$$\hat{z}_{t+1} = \underbrace{\hat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \hat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation	$\hat{z}_t = \ell_{t-1}$
Error Correction	$\ell_t = \ell_{t-1} + \alpha(z_t - \hat{z}_t),$



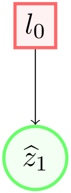
- Adjustable parameters: α and ℓ_0

Simple Exponential Smoothing (contd.)

$$\hat{z}_{t+1} = \underbrace{\hat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \hat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation	$\hat{z}_t = \ell_{t-1}$
Error Correction	$\ell_t = \ell_{t-1} + \alpha(z_t - \hat{z}_t),$



- Adjustable parameters: α and ℓ_0

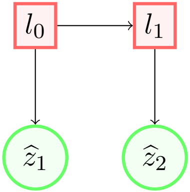
Simple Exponential Smoothing (contd.)

$$\widehat{z}_{t+1} = \underbrace{\widehat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \widehat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation $\qquad \widehat{z}_t = \ell_{t-1}$

Error Correction $\qquad \ell_t = \ell_{t-1} + \alpha(z_t - \widehat{z}_t),$



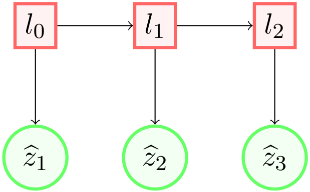
- Adjustable parameters: α and ℓ_0

Simple Exponential Smoothing (contd.)

$$\widehat{z}_{t+1} = \underbrace{\widehat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \widehat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation	$\widehat{z}_t = \ell_{t-1}$
Error Correction	$\ell_t = \ell_{t-1} + \alpha(z_t - \widehat{z}_t),$



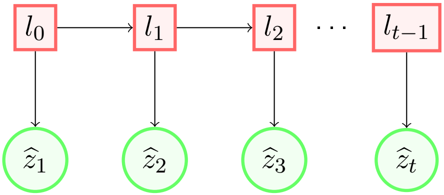
- Adjustable parameters: α and ℓ_0

Simple Exponential Smoothing (contd.)

$$\hat{z}_{t+1} = \underbrace{\hat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \hat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation	$\hat{z}_t = \ell_{t-1}$
Error Correction	$\ell_t = \ell_{t-1} + \alpha(z_t - \hat{z}_t),$

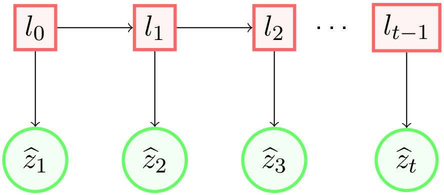


Simple Exponential Smoothing (contd.)

$$\widehat{z}_{t+1} = \underbrace{\widehat{z}_t}_{\text{previous forecast}} + \alpha \underbrace{(z_t - \widehat{z}_t)}_{\text{error in previous forecast}}$$

- Latent variable ℓ_t representing the "level" of the time series t

Forecast equation	$\widehat{z}_t = \ell_{t-1}$
Error Correction	$\ell_t = \ell_{t-1} + \alpha(z_t - \widehat{z}_t),$



- Adjustable parameters: α and ℓ_0

General Exponential Smoothing

- $\boldsymbol{l}_t$ is now an array representing unobserved patterns
  - Level, Trend, various seasonal effects (repeating patterns)

General Exponential Smoothing

- $\boldsymbol{l}_t$ is now an array representing unobserved patterns
  - Level, Trend, various seasonal effects (repeating patterns)

Forecast equation	$\widehat{z}_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$
Error Correction	$\boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t (z_t - \widehat{z}_t),$

General Exponential Smoothing

- $\boldsymbol{l}_t$ is now an array representing unobserved patterns
  - Level, Trend, various seasonal effects (repeating patterns)

Forecast equation $\quad \widehat{z}_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$

Error Correction $\quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t (z_t - \widehat{z}_t),$

- $\boldsymbol{a}_t$ $\boldsymbol{F}_t$ depend on the pattern being modeled

- $\boldsymbol{g}_t$ and $\boldsymbol{l}_0$ will be learned from data

Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$

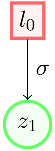
Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$



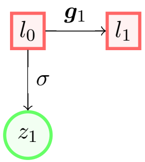
Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \text{diag}(\sigma_0^2)).$$

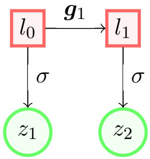


Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$
$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$

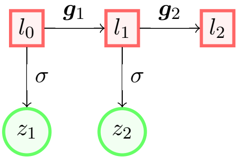


Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

Measurements	$z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$
State transition	$\boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$



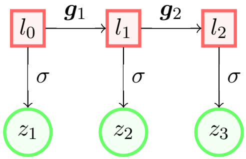
Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$



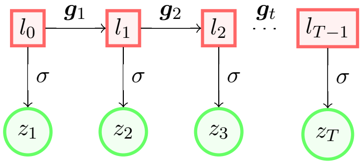
Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \text{diag}(\sigma_0^2)).$$



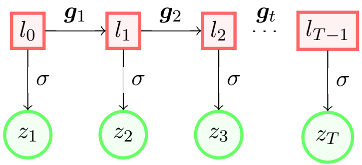
Linear State Space Model

- Statistical model: data generating process
  - produce an entire probability distribution for a future time period
  - compute prediction intervals with a given level of confidence

SSM:

$$\text{Measurements} \quad z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

$$\text{State transition} \quad \boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$



- Parameters: $\boldsymbol{l}_0$, $\boldsymbol{g}_t$ and σ

Linear State Space Models

- Linear dynamical system: $\boldsymbol{l}_t = \boldsymbol{F}\boldsymbol{l}_{t-1} + \boldsymbol{g}\varepsilon_t \qquad \varepsilon_t \sim N(0, 1)$
- Encompasses ARIMA and variants of ETS
- Combine linear model and dynamical system to yield

$$z_t \sim P(z_t|y_t)$$

$$y_t = \boldsymbol{x}_t^T \boldsymbol{w} + \boldsymbol{a}^T \boldsymbol{l}_{t-1}.$$

$$\boldsymbol{l}_t = \boldsymbol{F}\boldsymbol{l}_{t-1} + \boldsymbol{g}\varepsilon_t, .$$

State Space Model

Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

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

State Space Model

Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

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

Probabilistic model for data (likelihood): $z_t \sim P(z_t | u_t + b_t)$

State Space Model

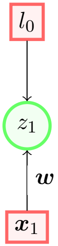
Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

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

Probabilistic model for data (likelihood): $z_t \sim P(z_t | u_t + b_t)$



Additional parameter: $\boldsymbol{w}$

State Space Model

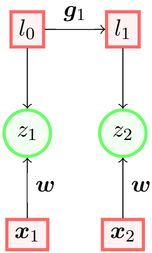
Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

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

Probabilistic model for data (likelihood): $z_t \sim P(z_t | u_t + b_t)$



Additional parameter: $\boldsymbol{w}$

State Space Model

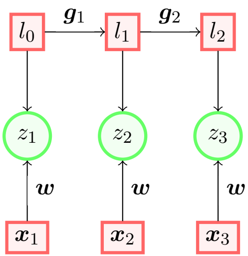
Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

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

Probabilistic model for data (likelihood): $z_t \sim P(z_t | u_t + b_t)$



Additional parameter: $\boldsymbol{w}$

State Space Model

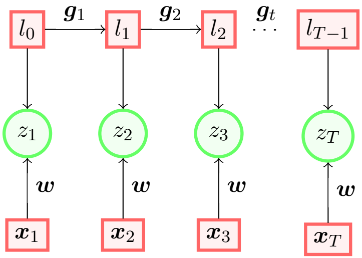
Linear State Space Model part:

$$u_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1}$$

Feature-based part:

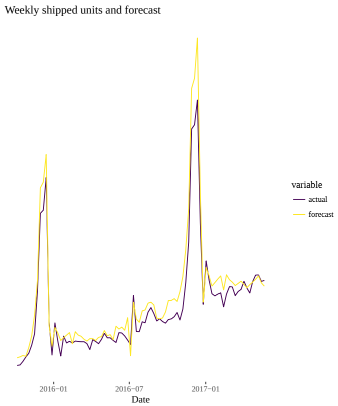
$$b_t = \boldsymbol{w}^T \boldsymbol{x}_t$$

Probabilistic model for data (likelihood): $z_t \sim P(z_t | u_t + b_t)$



Additional parameter: $\boldsymbol{w}$

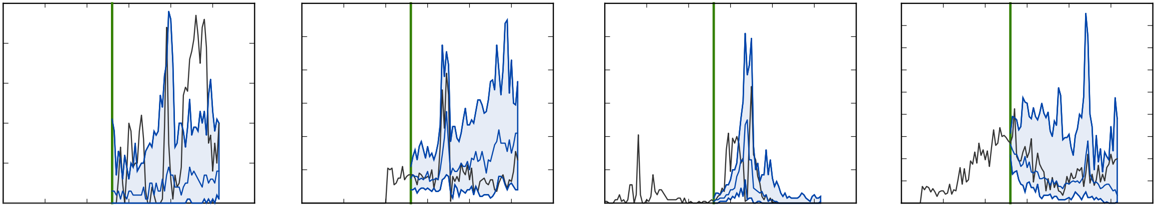
Classical methods are good for strategic forecasting problems



Predict overall Amazon retail demand years into the future.

Time series have enough history, are regular and exhibit clear patterns.

Classical models struggle with operational forecasting problems



Predict the demand for a each product available at Amazon

Time series are irregular, only combined do they have enough history and exhibit clear patterns.

The Classical Approach(es): Pros and Cons

Image (c) User:Colin / Wikimedia Commons / CC BY-SA 3.0

PROS

- De-facto standard; widely used
- Decomposition $\rightarrow$ decoupling
- White box: explicitly model-based
- Embarassingly parallel

CONS

- Requires lots manual work by experts $\Rightarrow$ hard to tune & maintain
- Cannot learn patterns across time series $\Rightarrow$ pipelines of models must be used
- Cannot handle cold-starts
- Model-based: all effects need to be explicitly modelled

Selected References

- General introduction: [Hyndman and Athanasopoulos, 2017]
- Classical textbooks: [Box et al., 2015; Brockwell and Davis, 2013]
- Exponential Smoothing and State Space Models: [Hyndman et al., 2008; Durbin and Koopman, 2012; Harvey, 1990]
- Bayesian Structure Time Series Models: [Scott and Varian, 2014; Taylor and Letham, 2018]
- Forecasting with exponential smoothing and intermittent time series: [Snyder et al., 2012]
- Approximate Bayesian inference for combination of state space and GLMs: [Seeger et al., 2016, 2017]
- Hierarchical forecasting: [Ben Taieb et al., 2017; Athanasopoulos et al., 2017; Wickramasuriya et al., 2018]
- Automatic time series forecasting: [Hyndman et al., 2007]
- ... ...

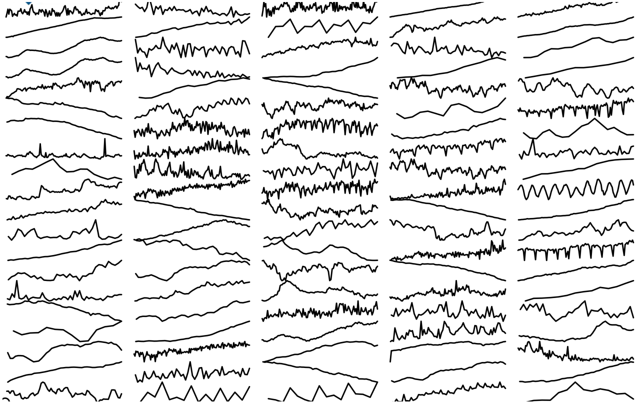
Modern Methods for Forecasting: Old and New

From Local to Global ...

- Old: local, one (preferable parsimonious!) model per time series

From Local to Global ...

- New: global, one large and complex/expressive model for all time series

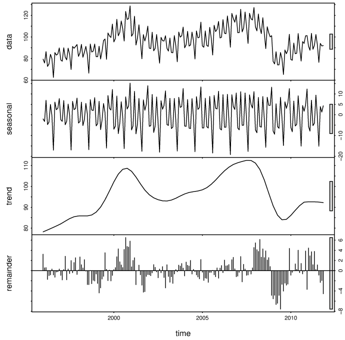


M3 Time series data visualization from [Hyndman, 2015]

Time Series Decomposition: Recap

Underlying patterns in the time series data

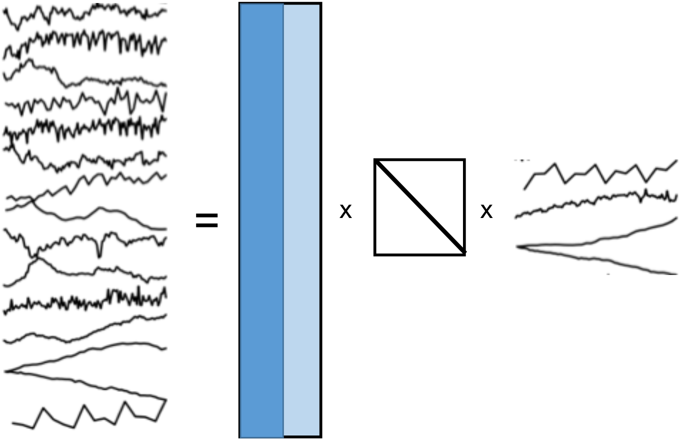
- Level
- Trend
- Seasonal variations (repeating patterns)



Questions

- How can we find common patterns across time series?
- How can we do forecast based on the learned (latent) patterns?

Matrix Decomposition

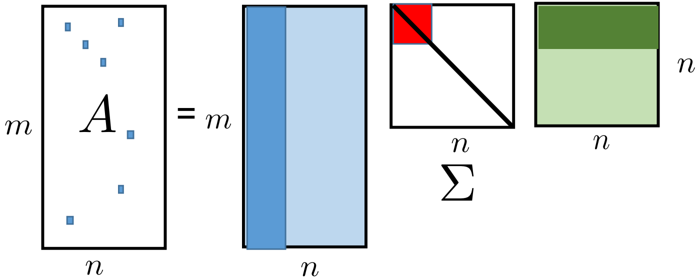


M3 Time series data visualization from [Hyndman, 2015]

Each time series is a linear combination of the hidden (time series) components.

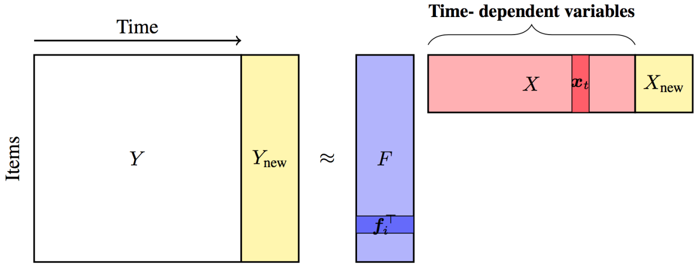
Matrix Decomposition

$$A \approx A_k = U_k \Sigma_k V_k'$$



- Perform k-SVD on the time series matrix $Z \approx U \Sigma V'$
- Use classical time series model such as ARIMA or ETS to forecast k hidden components
- Linearly combine the forecasts of the hidden components yields the individual forecasts

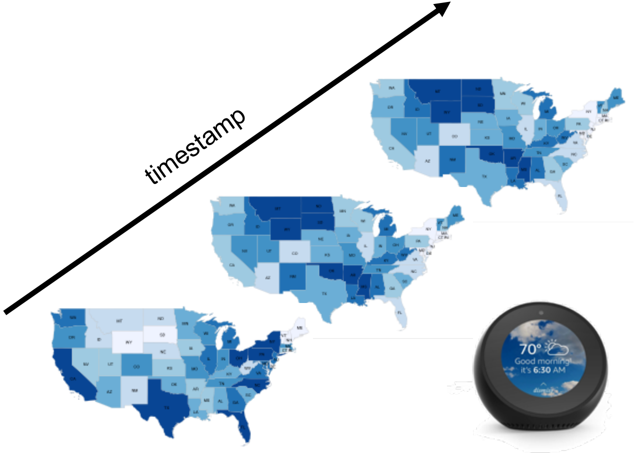
Temporal Regularized Matrix Factorization [Yu et al., 2016]



$$\min_{F,X} \sum_{(i,t)\in\Omega} (Y_{it} - \boldsymbol{f}_i^T \boldsymbol{x}_t)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{R}_x(X)$$

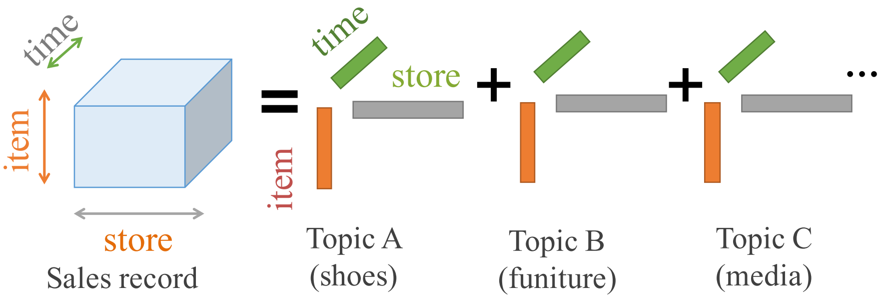
- Temporally regularization to encourage smoothness through time
- Minimizing L_2 loss without statistical assumption
- Only producing point forecasts and can not handle missing data

Tensor Decomposition: A Sample Problem



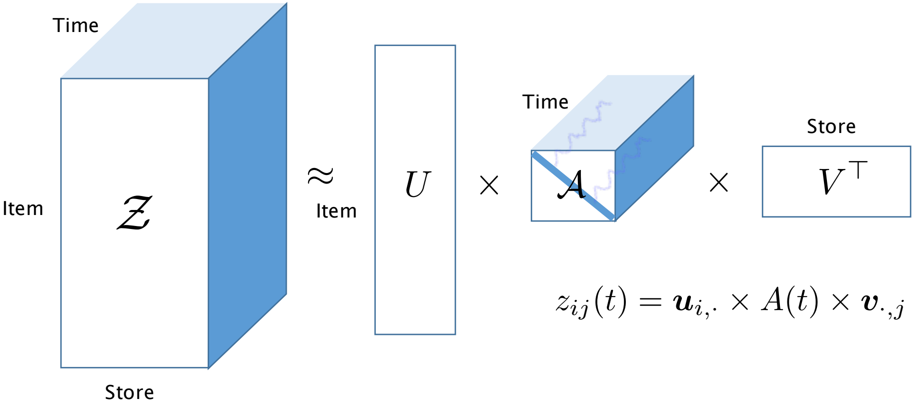
Forecast the sales for items in different locations at different time.

Tensor Decomposition: PARAFAC



Generalization of matrix decomposition for multi-view data.

Tensor Decomposition: PARAFAC



$$z_{ij}(t) = \boldsymbol{u}_{i,\cdot} \times A(t) \times \boldsymbol{v}_{\cdot,j}$$

Generalization of matrix decomposition for multi-view data.

Selected References

- Tensor decomposition and applications [Kolda and Bader, 2009]
- Latent space model for road networks to predict time-varying traffic [Deng et al., 2016]
- Autoregressive tensor factorization for spatio-temporal predictions [Takeuchi et al., 2017]
- High-Order Temporal Correlation Model Learning for Time-Series Prediction [Jing et al., 2018]
- Parcube: Sparse parallelizable tensor decompositions [Papalexakis et al., 2012]
- Fast mining and forecasting of complex time-stamped events [Matsubara et al., 2012]
- Tensorcast: Forecasting with context using coupled tensors [de Araujo et al., 2017]
- FUNNEL: automatic mining of spatially coevolving epidemics [Matsubara et al., 2014]
- ... ...

Neural Network Forecasting: Old and New – Timeline

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
Editor: Esfandiar Maasoumi
Volume 24 Number 3 2005

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

- 1969 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

Neural Network Forecasting: Old and New

"Consensus" in the Forecasting Community: NNs don't work!

This supports the general consensus in forecasting, that neural networks (and other highly non-linear and nonparametric methods) are not well suited to time series forecasting due to the relatively short nature of most time series. The longest series in this competition was only 126 observations long. That is simply not enough data to fit a good neural network model.

– Rob Hyndman on M-challenges, 2018

Neural Network Forecasting: Old and New

"Consensus" in the Forecasting Community: NNs don't work!

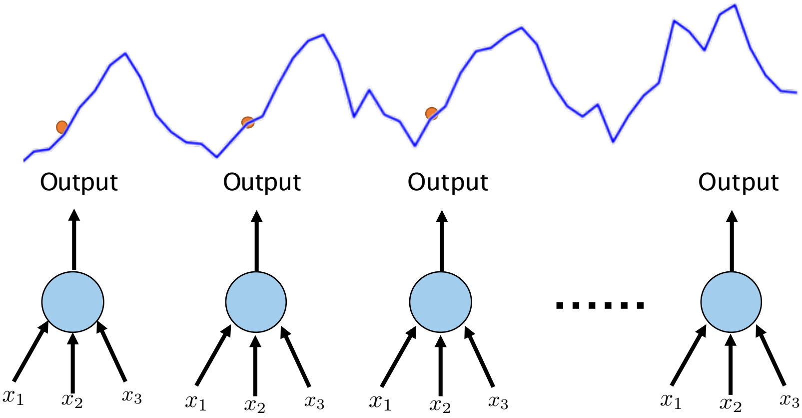
This supports the general consensus in forecasting, that neural networks (and other highly non-linear and nonparametric methods) are not well suited to time series forecasting due to the relatively short nature of most time series. The longest series in this competition was only 126 observations long. That is simply not enough data to fit a good neural network model.

– Rob Hyndman on M-challenges, 2018

Our View

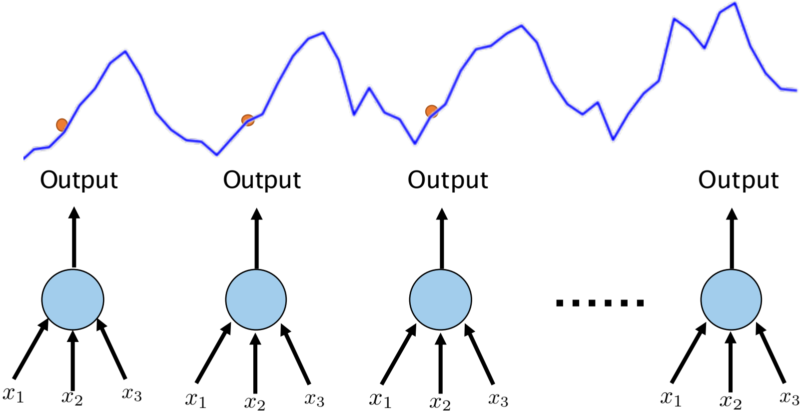
Neural networks are **great** for learning complex patterns from *many* time series in operational forecasting problems!

From Linear Regression to Feed-Forward Neural Networks



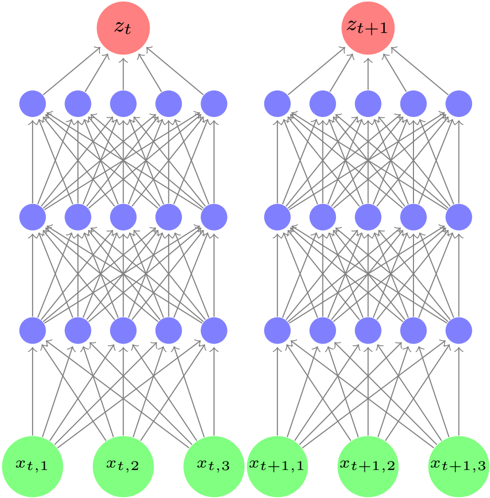
$$z_t = \boldsymbol{w}^T \boldsymbol{x}_t$$

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))

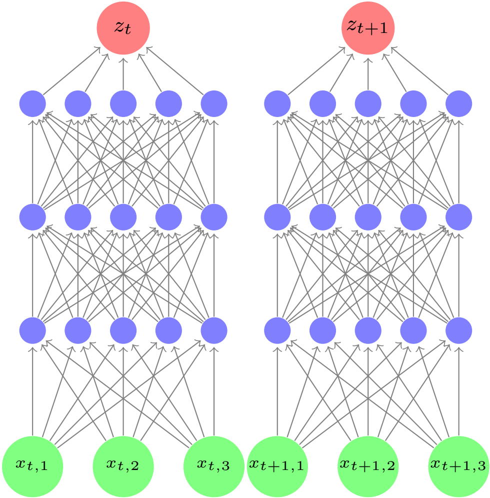


- Linear model + non-linear hidden layers
- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear *activation* function,

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

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

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



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

⇒ Less manual feature engineering

- 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*,

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

- 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, e.g., negative log likelihood.

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"

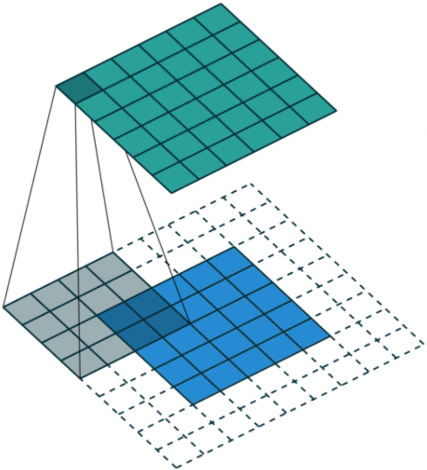
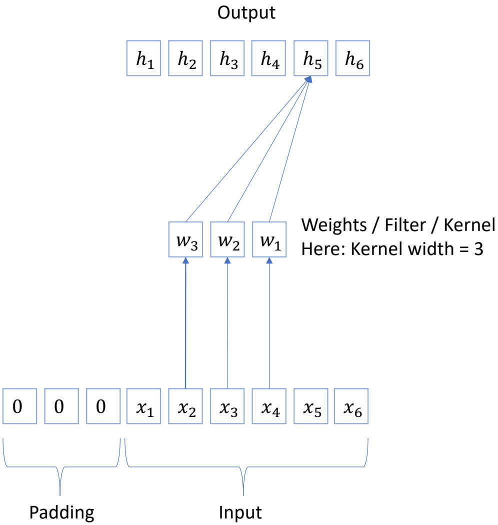


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers



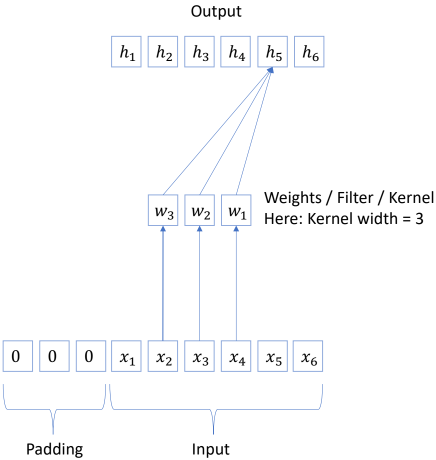
- The output h_j of a neuron j in a convolution layer is a discrete convolution of the inputs $\mathbf{x}$ with the layer's weigths/filter $\mathbf{w}$.
- For a one-dimensional convolution with a kernel with width D we have

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

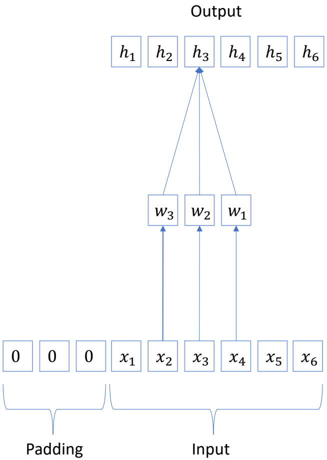
- Padding is used shift the input relative to the output and change the behavior around the edges (causal vs. non-causal)

Causal vs. Non-Causal Convolution

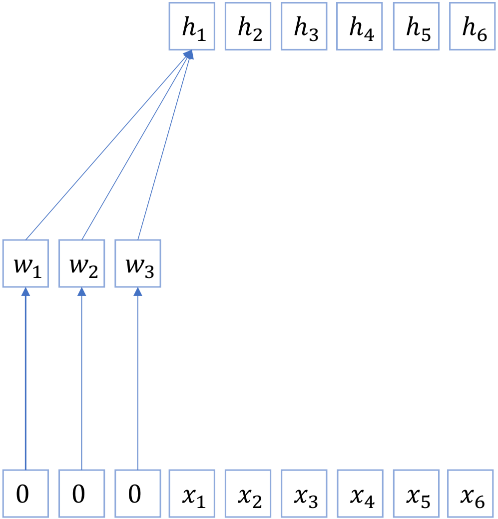
Causal Convolution



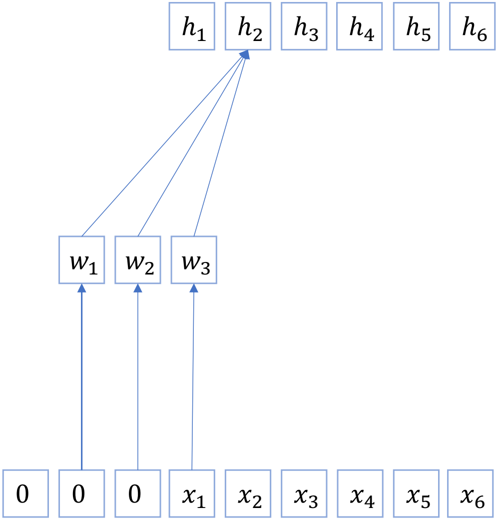
Non-Causal Convolution



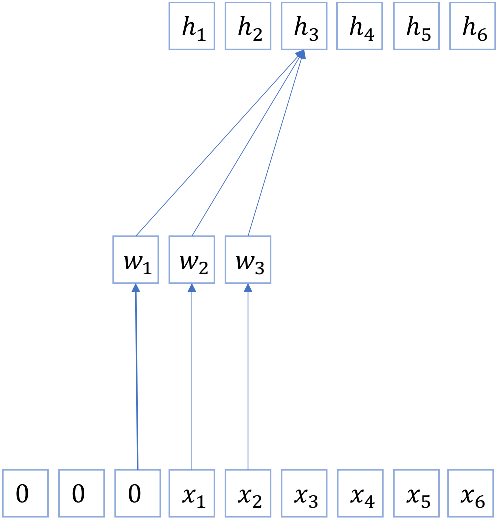
1D Causal Convolution



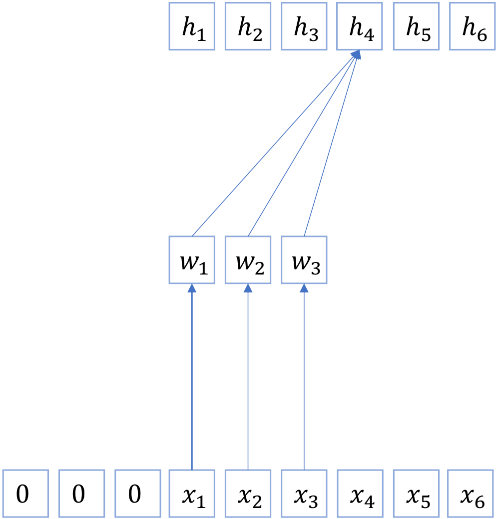
1D Causal Convolution



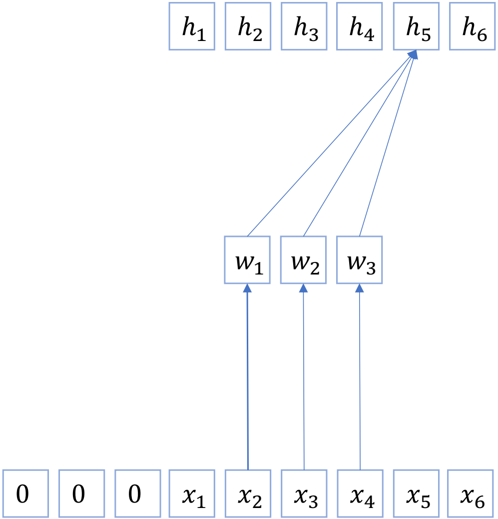
1D Causal Convolution



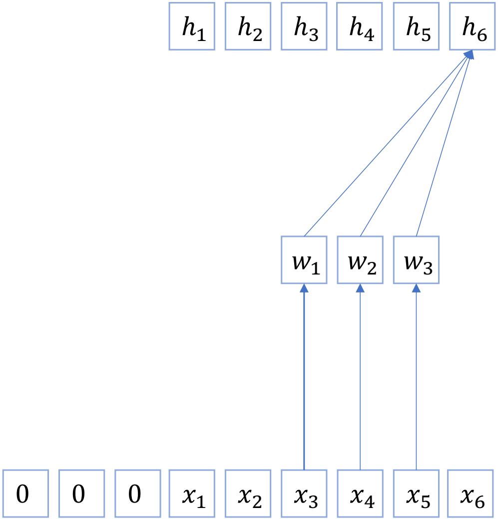
1D Causal Convolution



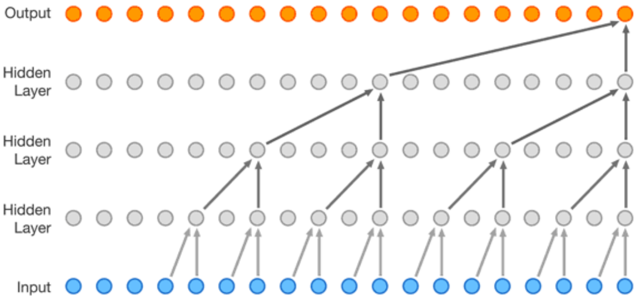
1D Causal Convolution



1D Causal Convolution



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



- Dilation increases receptive field
- Forecast is generated in an autoregressive fashion
- Can be used as encoder or decoder in sequence-to-sequence (next section)
- More complex structures including gating and residual links [Van Den Oord et al., 2016]

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

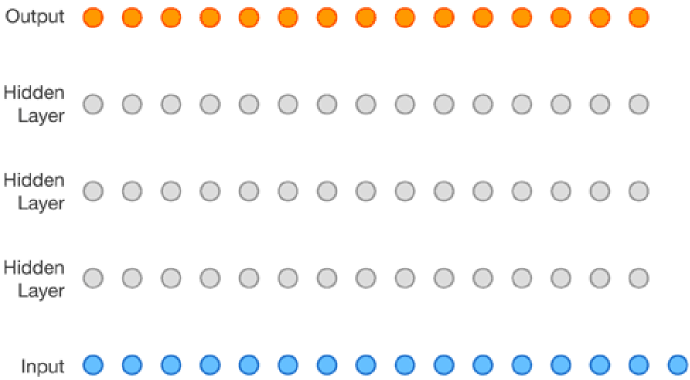
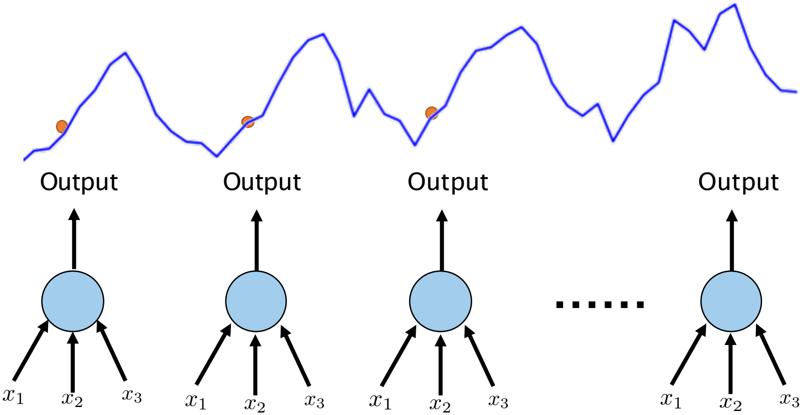


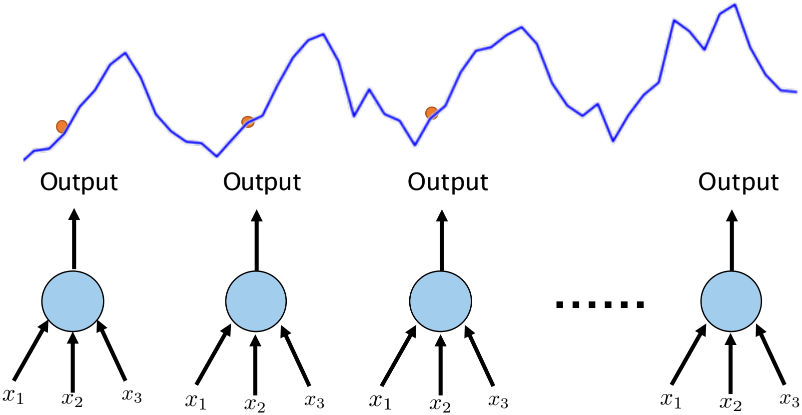
Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Recap: MLP for Forecasting



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

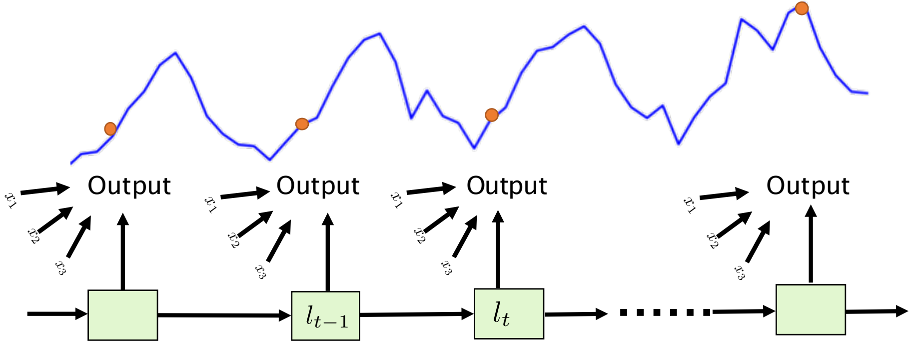
Recap: MLP for Forecasting



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

How about the sequential relationship?

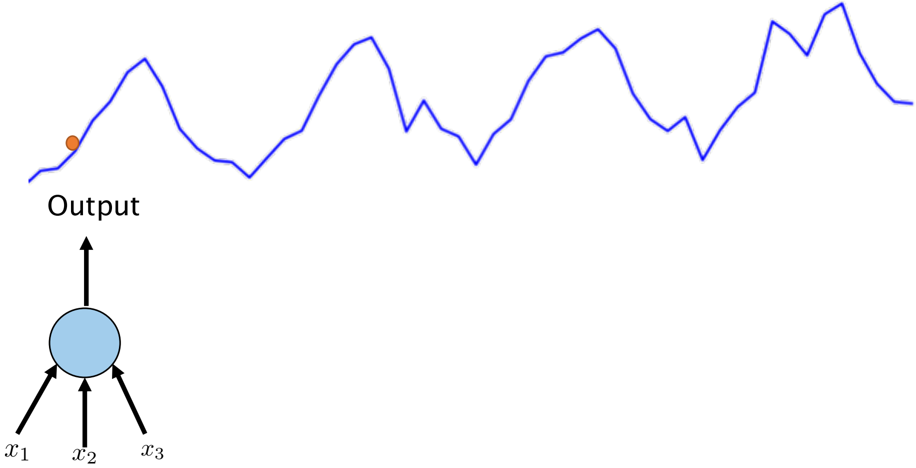
Recap: State-Space Models for Forecasting



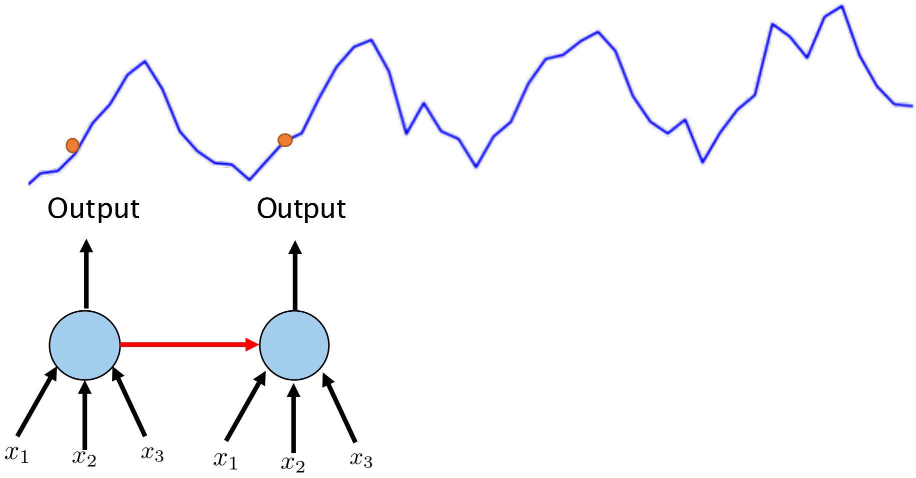
$$\boldsymbol{h}_t = \boldsymbol{l}_{t-1} + \alpha \cdot \epsilon_t$$
$$z_t = \boldsymbol{l}_t + \boldsymbol{w}^T \boldsymbol{x}_t + \epsilon_t$$

Can we do the same with NNs?

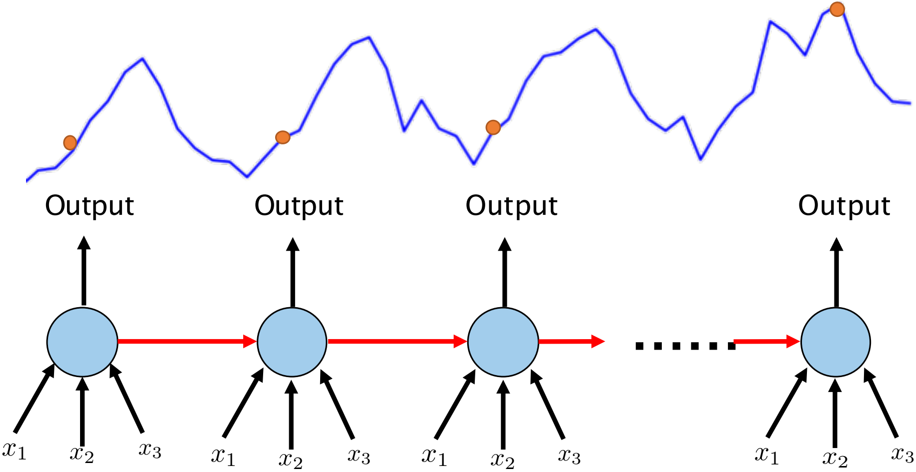
From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN



From Feed-forward NN to Recurrent NN

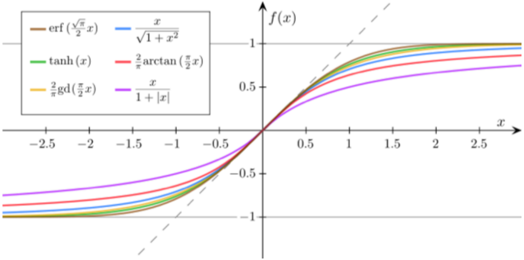


From Latent State (Exponential Smoothing) to Recurrent NN

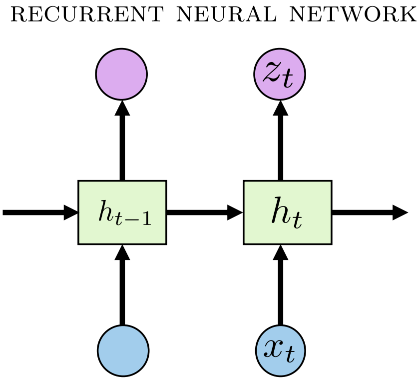
Current hidden state h_t combines

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

and goes into



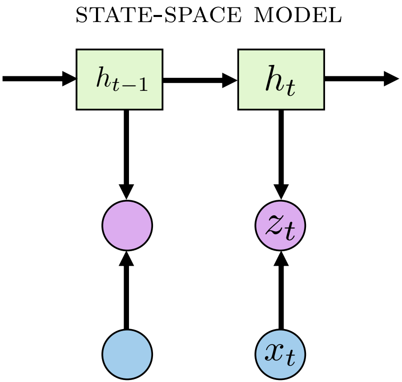
Source: Wikipedia

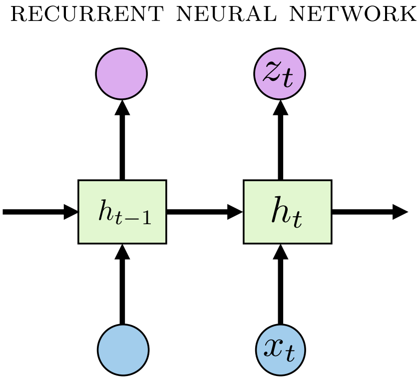


$$\boldsymbol{h}_t = \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$
$$z_t = \sigma(\theta h_t)$$

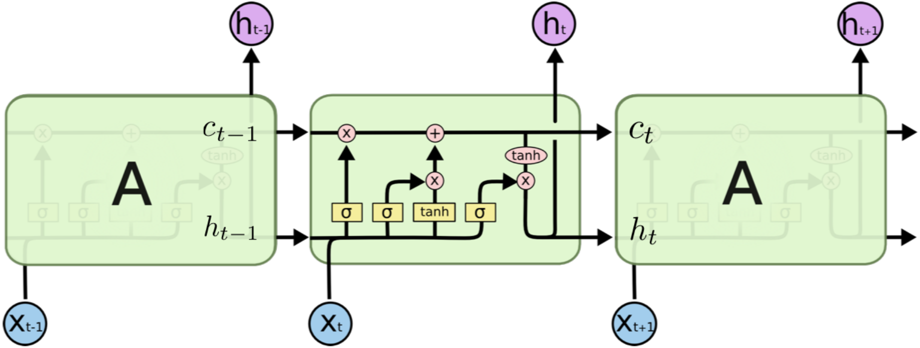
Central idea: Exponential Smoothing

today = yesterday's information + new knowledge





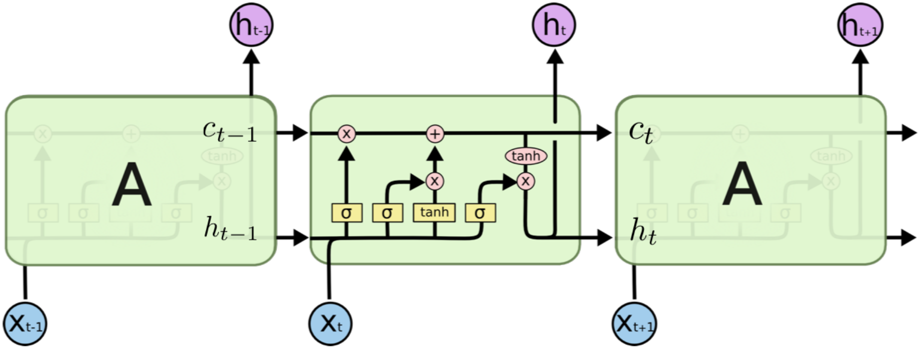
Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]



HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

WHAT AND WHY?

Long Short-Term Memory (LSTM): What?



HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

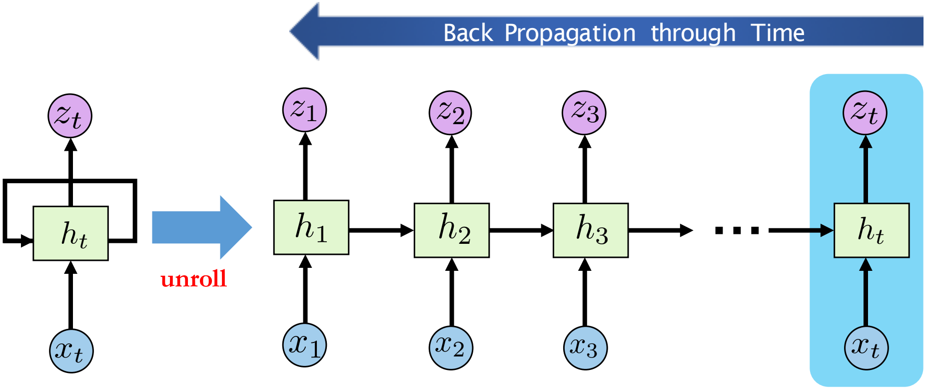
$$C_t = \alpha_t \cdot C_{t-1} + \beta_t \times \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$

current state = forgot gate × old stuff + input gate × new stuff.

The same exponential smoothing idea!

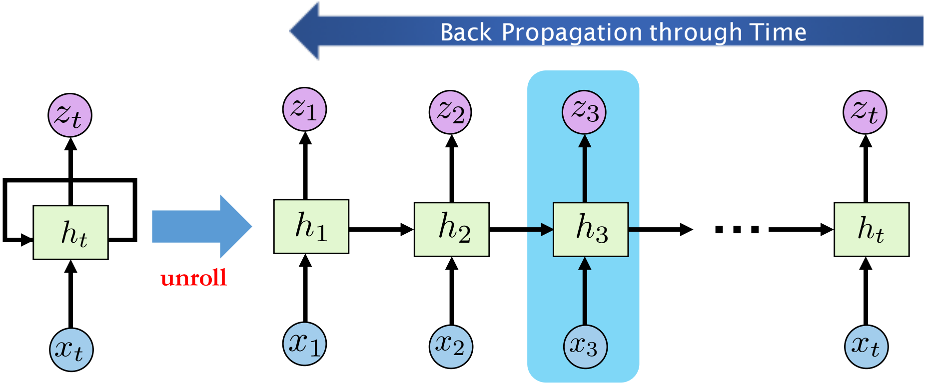
Long Short-Term Memory (LSTM): Why?

What do you wish to happen



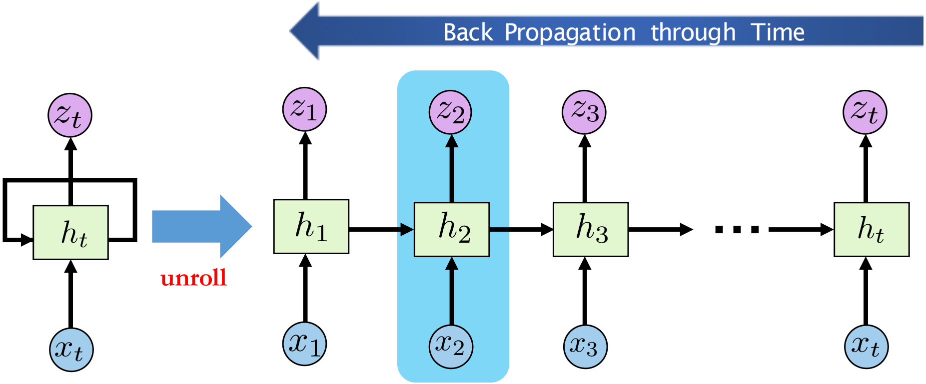
Long Short-Term Memory (LSTM): Why?

What do you wish to happen



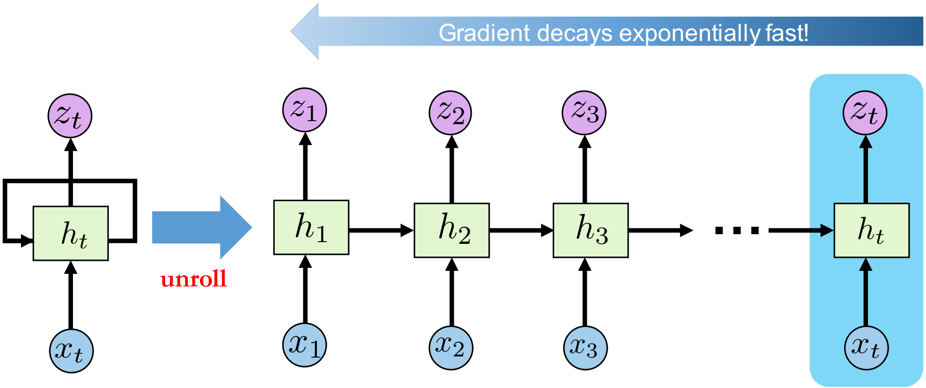
Long Short-Term Memory (LSTM): Why?

What do you wish to happen



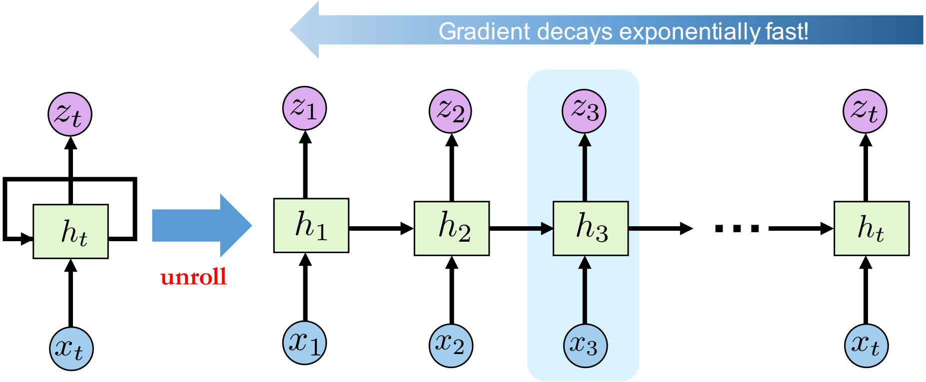
Long Short-Term Memory (LSTM): Why?

WHAT REALLY HAPPENS



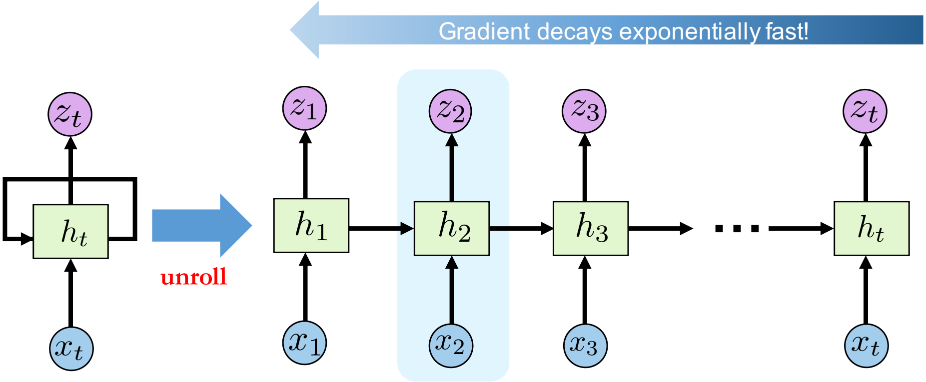
Long Short-Term Memory (LSTM): Why?

What really happens

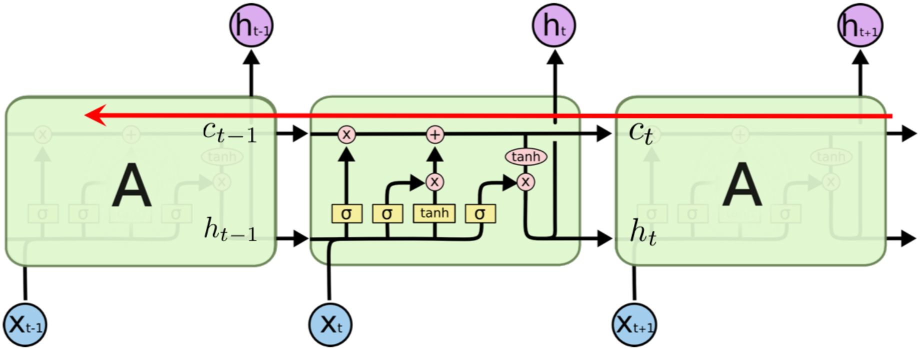


Long Short-Term Memory (LSTM): Why?

What really happens



Long Short-Term Memory (LSTM): Here is Why!

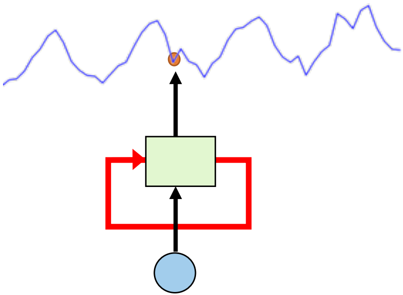


HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

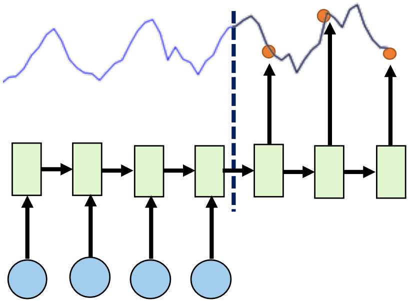
LSTM $\frac{\partial c_t}{\partial c_{t-1}} = \text{diag}(f_t)$ $c_t = f \odot c_{t-1} + i \odot g$

Canonical RNN $\frac{\partial h_t}{\partial h_{t-1}} = \theta_0$ $h_t = \sigma(\theta_0 h_{t-1} + \theta_1 x_1)$

Back to Forecasting



Canonical (One-to-One)

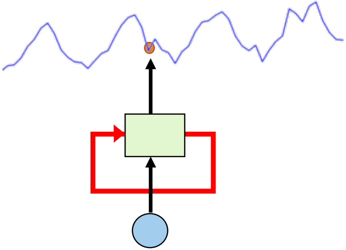


Seq2Seq (Many-to-Many)

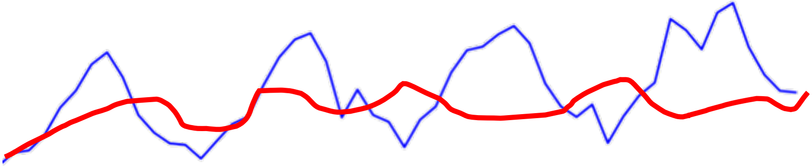
- Canonical (One-to-One) RNN: DeepAR [Flunkert et al., 2017], AR-MDN [Mukherjee et al., 2018], Deep LSTM [Yu et al., 2017a], ...
- Sequence-to-Sequence (Many-to-Many) models: Diffusion Convolutional RNNs [Li et al., 2018], MQ-RNN [Wen et al., 2017], ...

Canonical RNN Structure (One-to-One)

$$f_t : x_t \mapsto z_t$$



Canonical (One-to-One)

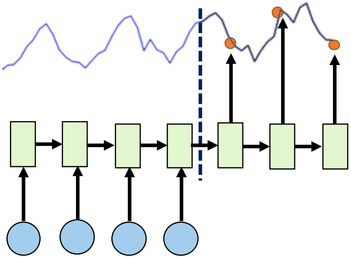


Training sequence

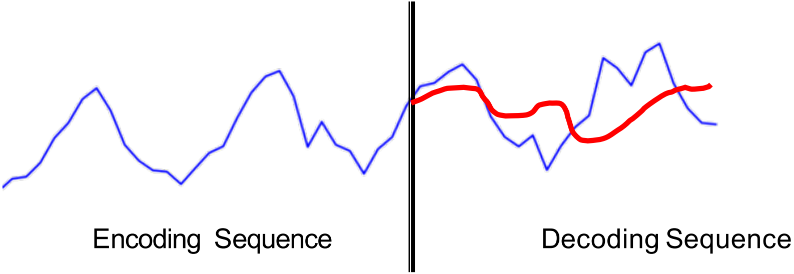
How well does the prediction reconstruct the the observed time series?

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

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



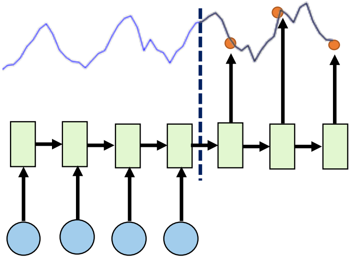
Seq2Seq (Many-to-Many)



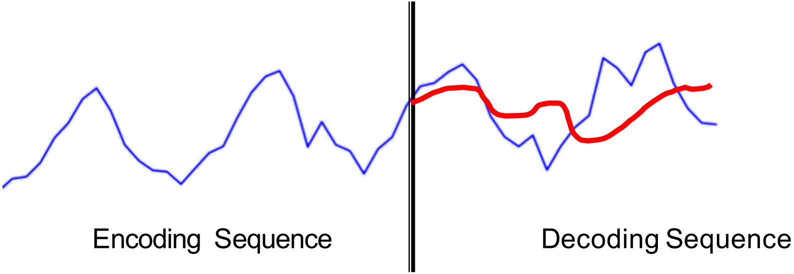
How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence?

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

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

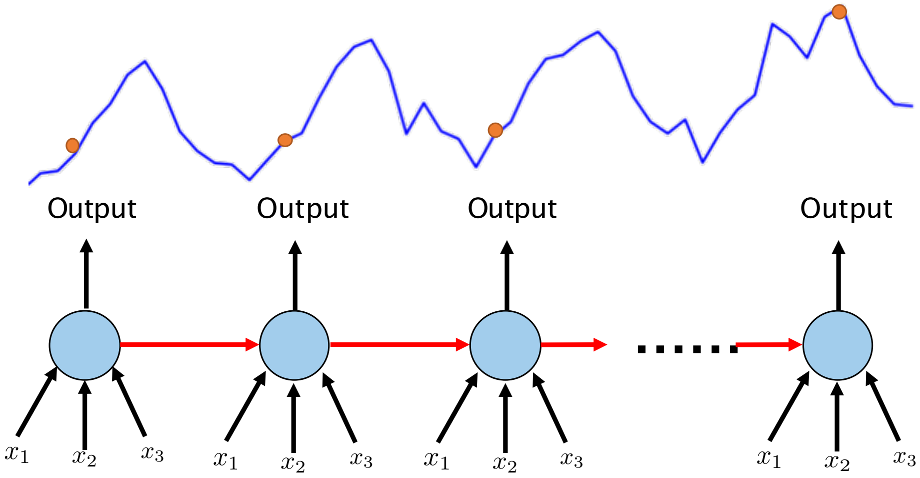


Seq2Seq (Many-to-Many)

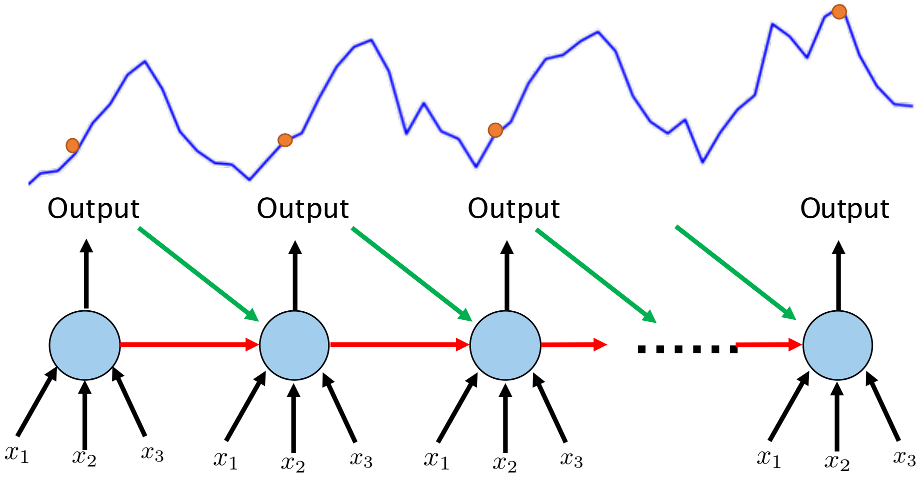


How well does the prediction reconstruct the decoding sequence conditioned on the encoding sequence? Conceptually close to multivariate regression

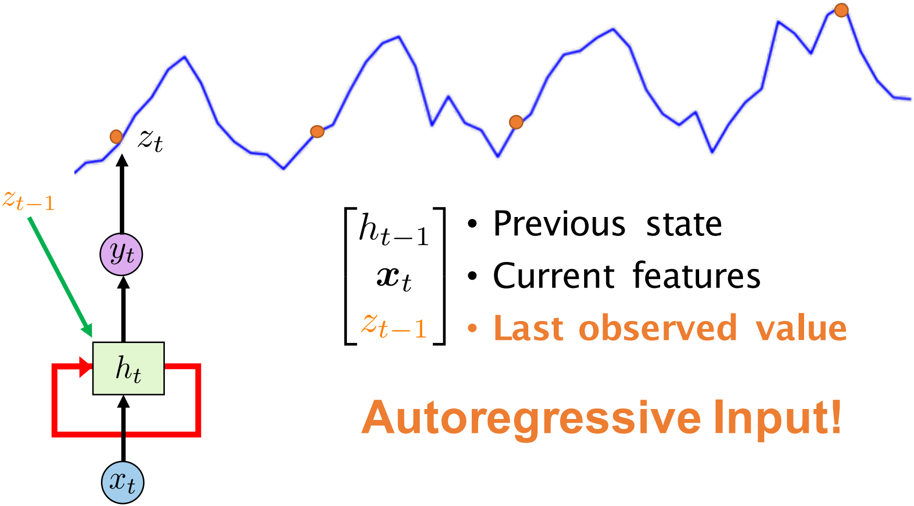
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



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



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

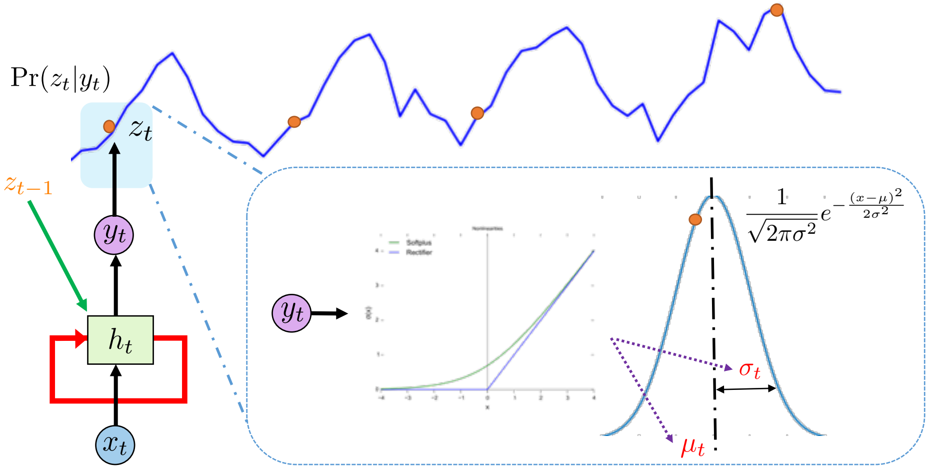


$$\begin{bmatrix} h_{t-1} \\ \boldsymbol{x}_t \\ z_{t-1} \end{bmatrix}$$

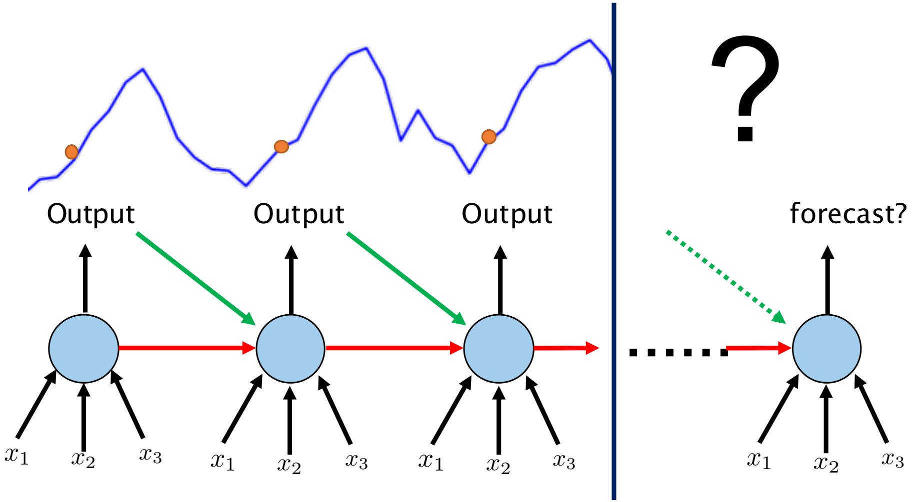
- Previous state
- Current features
- **Last observed value**

Autoregressive Input!

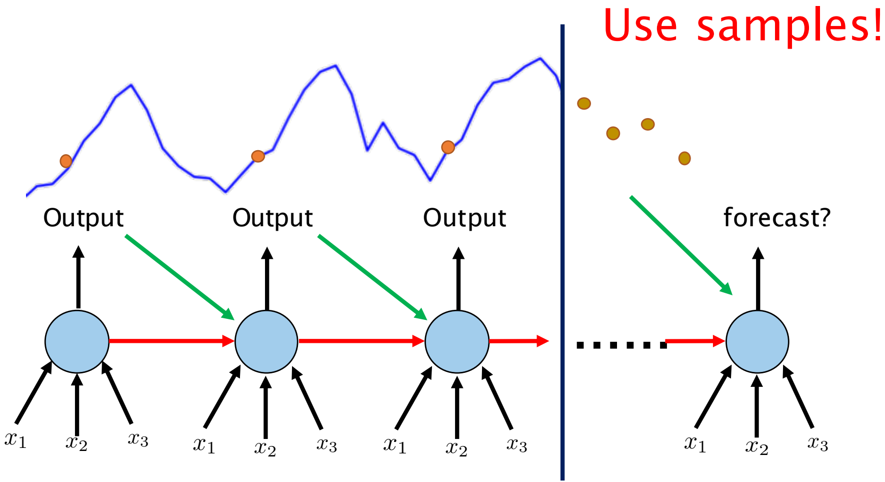
Canonical RNN Structure: DeepAR [Flunkert et al., 2017]



Canonical RNN Structure: How do we do forecast?

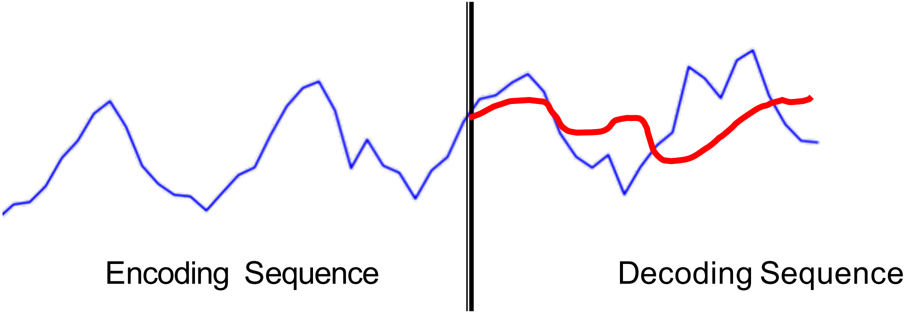


Canonical RNN Structure: How do we do forecast?

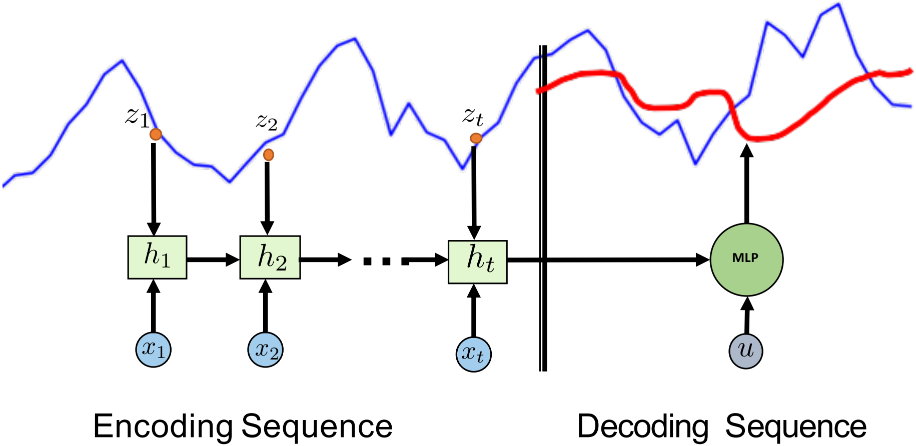


Sequence to Sequence (Seq2Seq) Structure: Many-to-Many

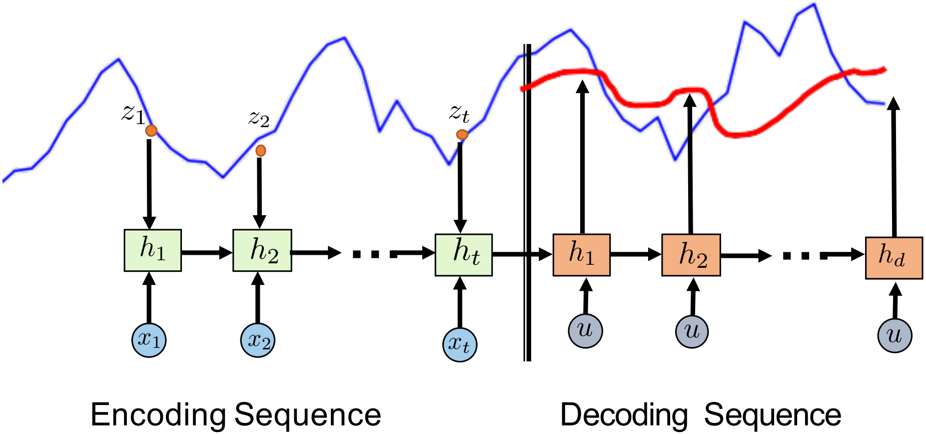
$$f_{encoder} : \{z_1, \cdots, z_{T_e}\} \mapsto \boldsymbol{h}_{T_e}$$
$$f_{decoder} : \boldsymbol{h}_{T_e} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



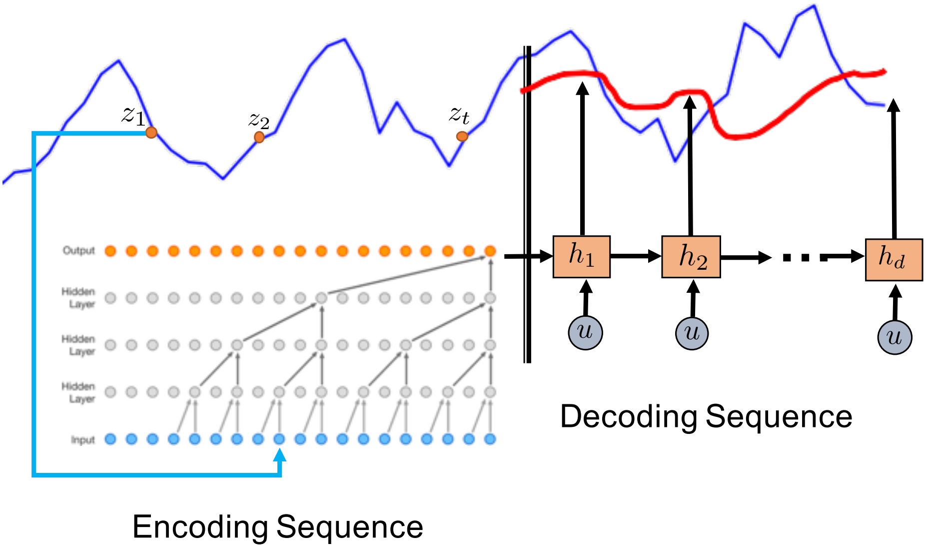
Seq2Seq: RNN-MLP [Wen et al., 2017]



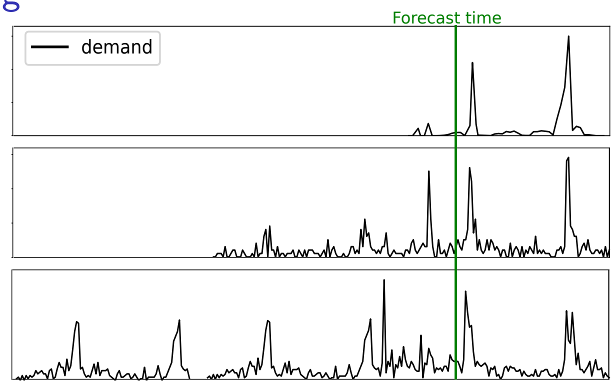
Seq2Seq: RNN-RNN



Seq2Seq: Causal CNN-RNN

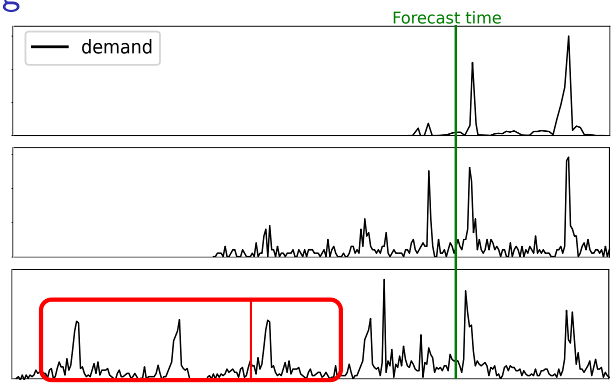


Seq2Seq: Training



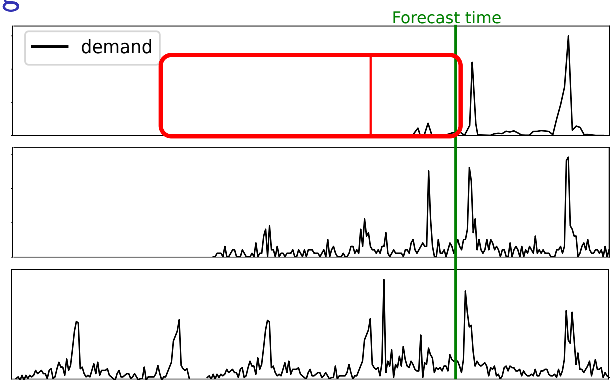
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s

Seq2Seq: Training



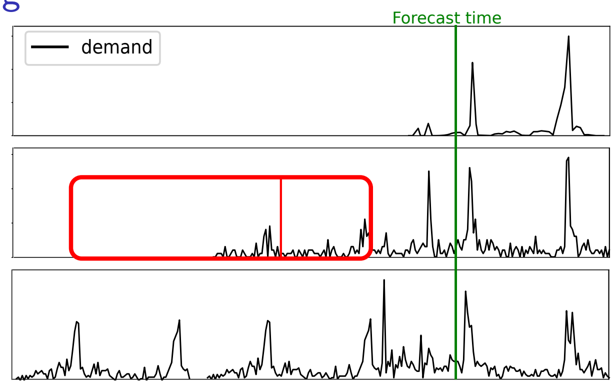
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



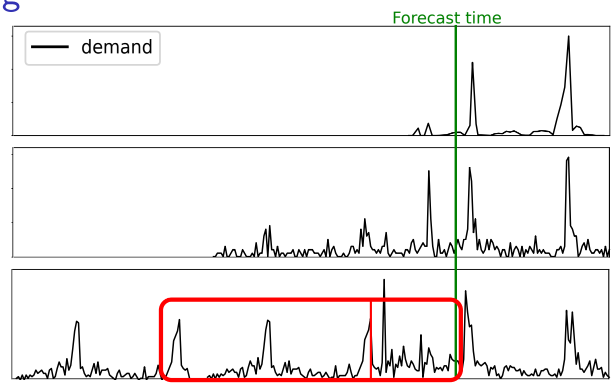
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training



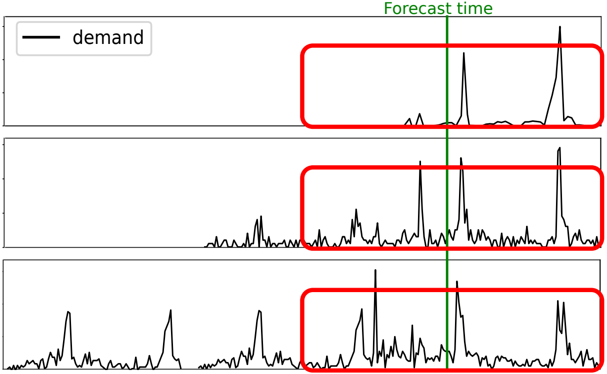
- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Training

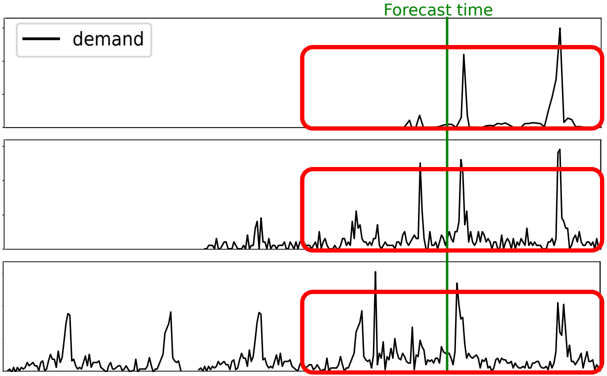


- Input:
  - time series (targets) z_t's: encoding and decoding
  - input features: encoding $\boldsymbol{x}_t$'s and decoding $\boldsymbol{u}$'s
- Slicing windows across item and time (temporal)
- Trained by minimizing certain metrics (negative loglikelihood, L_1/L_2 loss, quantile loss, etc.)

Seq2Seq: Prediction

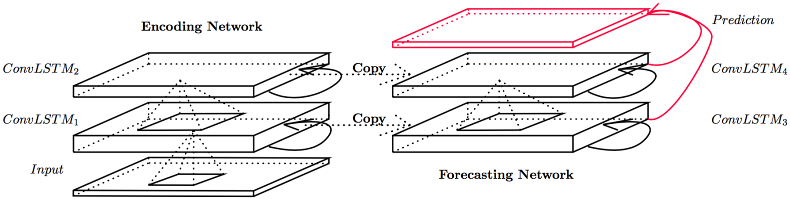


Seq2Seq: Prediction



- target z_t is unobserved after the forecast time

Seq2Seq: Convolutional LSTM [Xingjian et al., 2015]

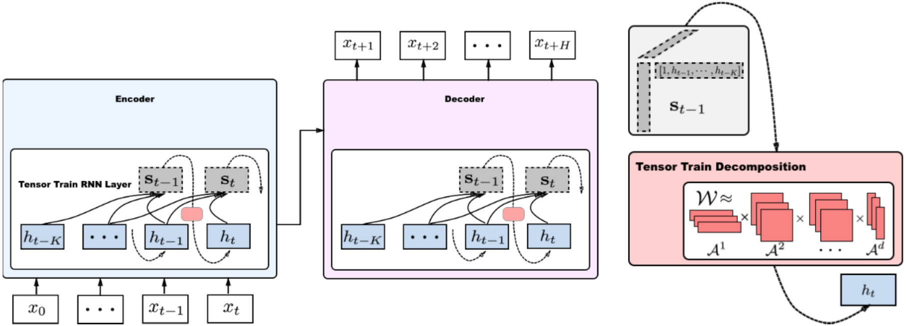


LSTM $c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c)$

ConvLSTM $\mathcal{C}_t = f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c)$

- Input is time series of images (tensor data)
- Convolution happens in the spatial domain

Seq2Seq: Tensor-Train RNNs [Yu et al., 2017b]

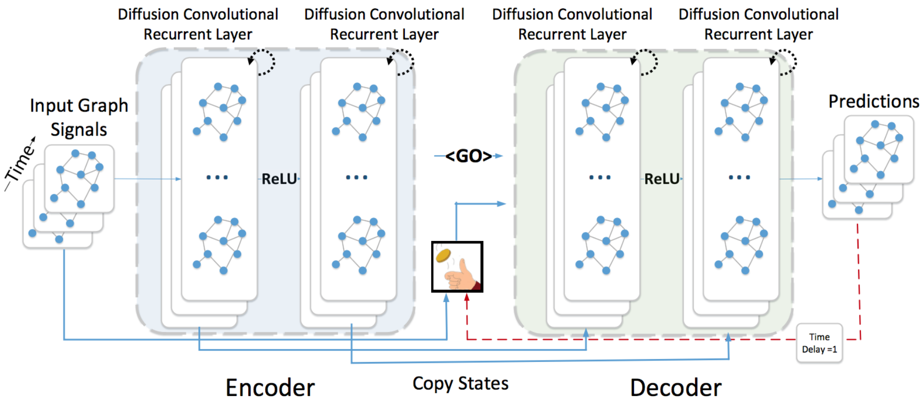


- State transition is L-th order Markov transition to capture higher-order dynamics

$$h_t = f(x_t, h_{t-1}, \cdots, h_{t-L}),$$

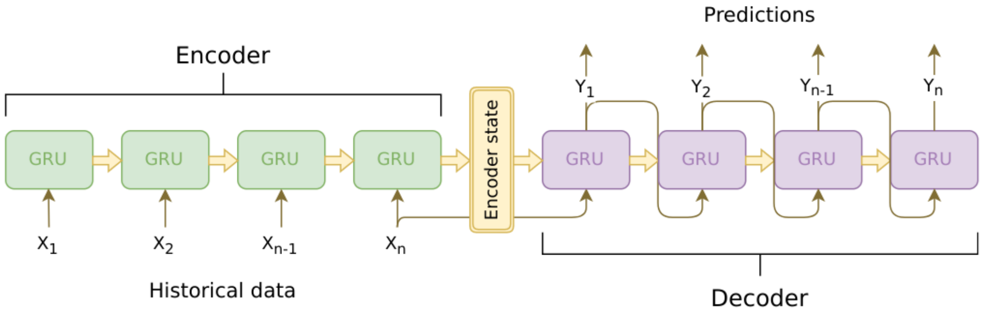
- Tensor train decomposition to approximate the weight tensor
- Polynomial interactions between the hidden states h_t and x_t

Seq2Seq: Diffusion Convolutional RNNs [Li et al., 2018]



- Pair-wise spatial correlations between traffic sensors as a directed graph
- Diffusion convolution on the graphs

Seq2Seq: Gated Recurrent Units (GRU) [Suilin, 2017]

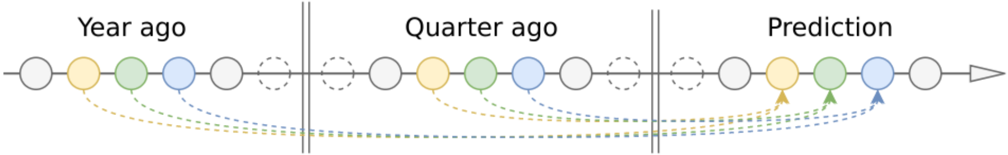


https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

- Kaggle Wikipedia winning solution: GRU and GRU decoder

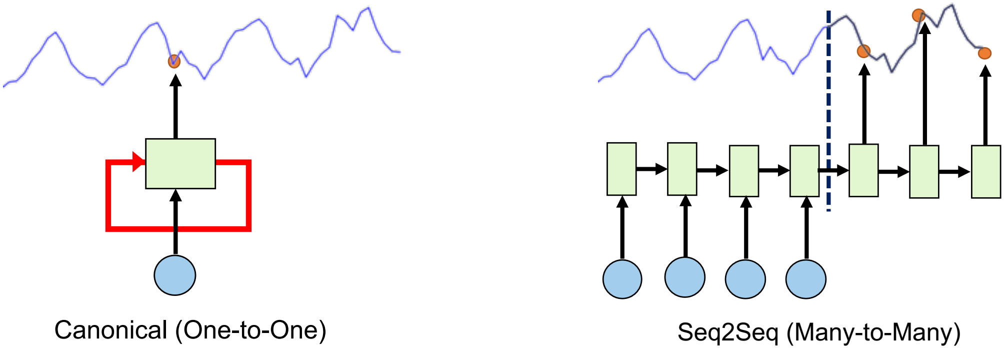
What about ATTENTION?

Lag is All You Need!



https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

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

So ... Neural Networks, huh?

When to use what for which types of time series?

- MLPs are robust baseline methods, but requires heavy feature engineering
- RNNs are, the *de facto* standard model for sequence modeling. Sometimes stability problems in training.
- Recent research [Miller and Hardt, 2018; Bai et al., 2018] advocates that CNNs are as accurate but much more efficient
- But people argue that dilated RNNs [Chang et al., 2017] are just as efficient ...

So ... Neural Networks, huh?

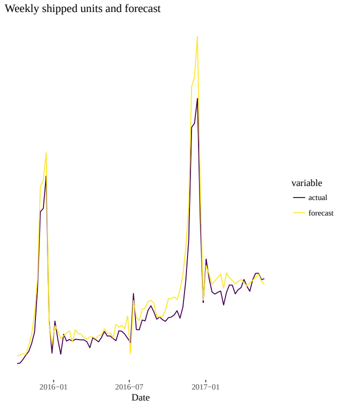
When to use what for which types of time series?

- MLPs are robust baseline methods, but requires heavy feature engineering
- RNNs are, the *de facto* standard model for sequence modeling. Sometimes stability problems in training.
- Recent research [Miller and Hardt, 2018; Bai et al., 2018] advocates that CNNs are as accurate but much more efficient
- But people argue that dilated RNNs [Chang et al., 2017] are just as efficient ...

We do not know yet, but we will!

http://quantumfuture.net/quantum_future/homepage.htm

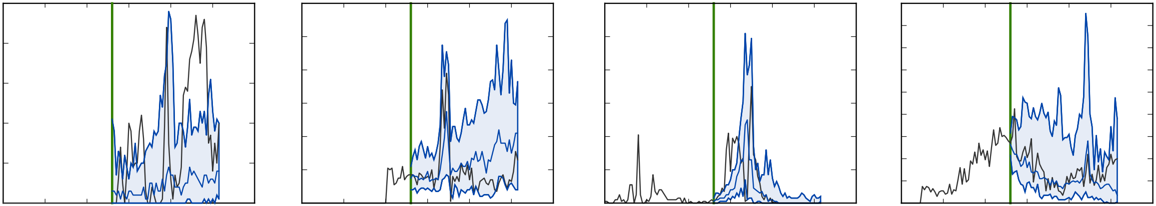
Modern methods struggle with strategic forecasting problems



Predict overall Amazon retail demand years into the future.

Not enough data may be available for training, assumptions on long-term behaviour should be handled properly. Use a classical, local model

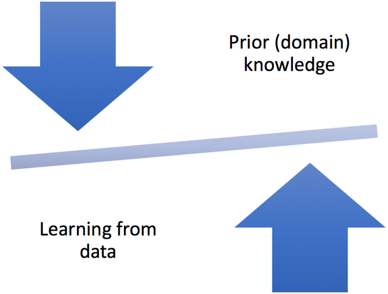
Modern models handle operational forecasting problems well



Predict the demand for a each product available at Amazon

Time series are irregular, only combined to they have enough history and exhibit clear patterns.

Finding the right balance: data vs model driven

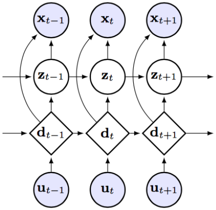
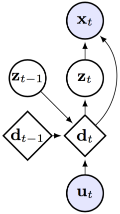
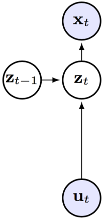
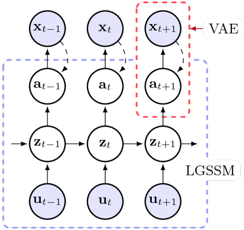


Goals:

- increase data efficiency: data efficiency improves sample complexity
- improve interpretability: interpretability facilitates better decision-making
- enforce structure: structure enables fast computation

Combining Probabilistic Graphical Models (PGM) and DNN

Efficient inference of PGM with the flexibility and expressibility of DNN

SRNN [Fraccaro et al., 2016]	VRNN/SSL/LSTM-LDA [Chung et al., 2015; Zaheer et al., 2017; Zheng et al., 2017]	DMM [Krishnan et al., 2017, 2015]	KVAE/DVBF [Fraccaro et al., 2017; Karl et al., 2017]
			

Further avenues: hybrid global-local models.

Selected References

- NN Forecasting in the early days: [Tang et al., 1991; Azoff, 1994; Gately, 1995; Zhang et al., 1998]
- Convolution structure for sequence modeling: [Van Den Oord et al., 2016; Bai et al., 2018]
- Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting [Ghaderi et al., 2017]
- Autoregressive Convolutional Neural Networks for Asynchronous Time Series [Bińkowski et al., 2017]
- Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals [Alberg and Lipton, 2017]
- Time-series extreme event forecasting with neural networks at Uber [Laptev et al., 2017]
- An overview and comparative analysis of recurrent neural networks for short term load forecasting [Bianchi et al., 2017]
- ... ...

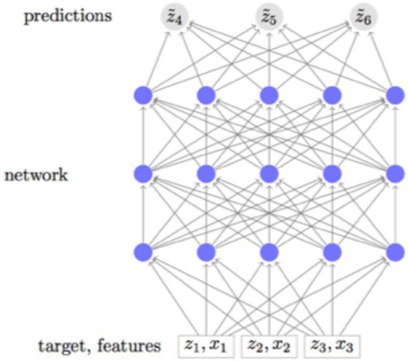
Building Forecasting Systems: Old and New

Pecularities of forecasting systems

- Time plays a role: important for backtesting & evaluation
- main primitive type are time series (one dimension more than usual)
- difference between data at train and inference time
- long feedback cycles (e.g., compare with recommender systems)
- complex interaction with downstream decision problems
- traditionally batch system, moving towards on-demand/real-time systems
- B2B not B2C scenario
- users are typically Business Intelligence officers, analysts, data scientists or business functions

Forecasting Systems: Two Extremes

vs

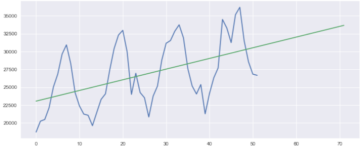


A complex pipeline of simple model vs. a complex model in a simple pipeline.

Example: m4 forecasting competition. Won by neural network approach, follow-up by ensemble methods. [Makridakis et al., 2018]

Building Forecasting Systems with Classical Models

Building Forecasting Systems with Classical Models



Building Forecasting Systems with Classical Models

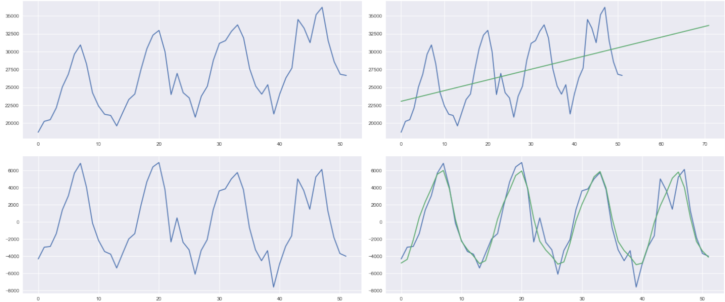
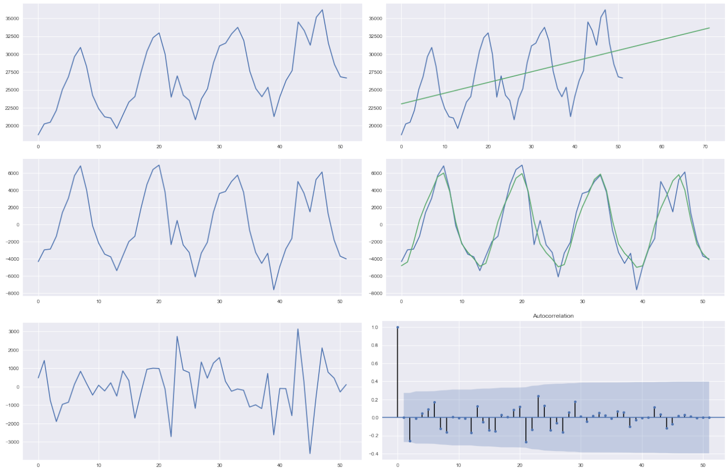


Image (c) User:Colin / Wikimedia Commons / CC BY-SA 3.0

Building Forecasting Systems with Classical Models

Image (c) User:Colin / Wikimedia Commons / CC BY-SA 3.0



Forecasting Systems with classical models

Image (c) User:Colin / Wikimedia Commons / CC BY-SA 3.0

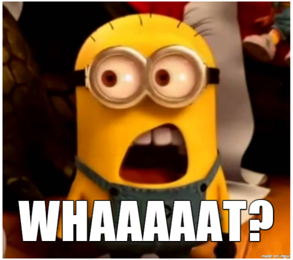
PROS

- models are canonical and relatively easy to understand
- Decomposition $\rightarrow$ decoupling
- White box: explicitly model-based
- Embarassingly parallel

CONS

- Requires lots manual work by experts $\Rightarrow$ hard to tune & maintain
- Cannot learn patterns across time series $\Rightarrow$ pipelines of models must be used
- Cannot handle cold-starts
- Model-based: all effects need to be explicitly modelled

Notable Non-Advantages



interpretability even though each model may be interpretable, the pipeline is not.

running time even though each models runs quickly, entire pipeline does not $\rightarrow$ on-demand forecasting hard to realize.

simple infrastructure forecasting pipeline require complex model combination mechanisms and feature preprocessing.

further things maintainability, tunability, ...

Forecasting System Architecture

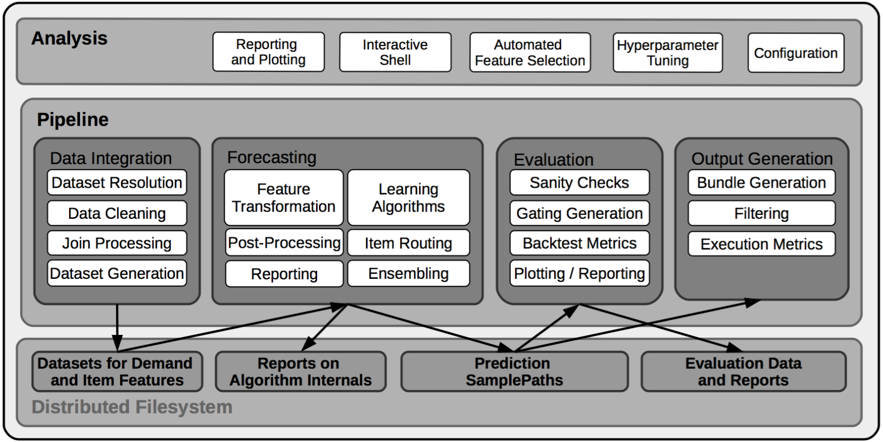
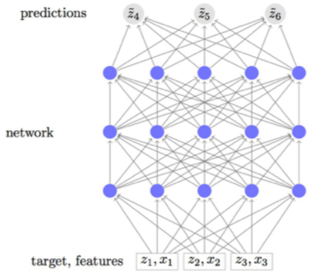


Figure from *Probabilistic Demand Forecasting at Scale* [Böse et al., 2017]

Neural Forecasting Approaches



PROS

- litte feature engineering needed
- learns across time series
- quick at inference
- default settings lead to surprinsingly good results
- state-of-the art performance in competitions

CONS

- little control over predictions
- potentially high-variance in training
- costly to train
- model serving infrastructure needed

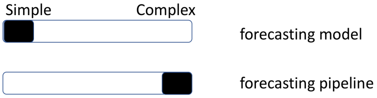
Recent public competitions won by neural forecasting approaches: m4 competiton, wikipedia Kaggle competition.

Third Prinicple

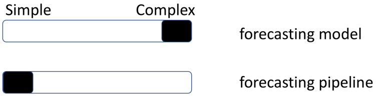
Conservation law

Forecasting systems are complex.

Classical forecasting system



Neural forecasting system



Naturally, combinations of both extremes are possible.

Forecasting system are ML systems

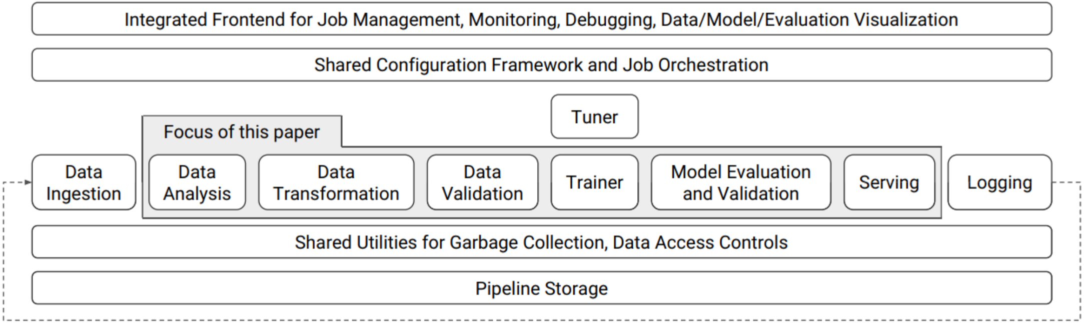


Figure from [Polyzotis et al., 2017]

ML systems: uncomprehensive laundry list of components

- ETL & Data Provenance management
- Data Cleaning, Imputation & monitoring (both for training and inference data)
- feature transformation component
- model training & experiment tracking
- model ensembling
- hyper-parameter optimization & Auto-ML/Meta-Learning
- model serving & management
- model monitoring
- live testing: bandits & A/B tests
- reporting & plotting & notebook
- configuration & orchestration
- ...

Take away: many challenges. See more in [Modi et al., 2017].

Selected References

- TFX: [Modi et al., 2017; Polyzotis et al., 2017]
- Spark-based ML: [Boehm et al., 2016; Meng et al., 2016; Sparks et al., 2017]
- Declarative ML: [Schelter et al., 2016]
- Data verification: [Schelter et al., 2018]
- Missing data: [Biessmann et al., 2018]
- Model serving: [Crankshaw et al., 2017, 2015]
- Experiment and Meta-Data Tracking: [Schelter et al., 2017]

Selected References: Forecasting competitions

M4 competition: [Makridakis et al., 2018] (and predecessors)

- Winning entry: https://eng.uber.com/m4-forecasting-competition/

Kaggle competitions on forecasting:

- Rossmann store sales: https://www.kaggle.com/c/rossmann-store-sales
- Wikipedia traffic forecast: https://www.kaggle.com/c/web-traffic-time-series-forecasting

Getting Started with Forecasting

Open-source forecasting packages: Classical methods

Rob Hyndman's R package [Hyndman et al., 2007] is among the most popular packages. Contains implementations for many classic methods. Very robust, very hard to beat. You have to like R.

Facebook's Prophet package [Taylor and Letham, 2018] uses Stan [Carpenter et al., 2017] behind the scences. Very flexible but the inference is slow.

Open-source deep forecasting packages

MXNet/Gluon [Chen et al., 2015] contains a number of notebooks (linked from our website) to get started, e.g., https://gluon.mxnet.io/chapter12_time-series/lds-scratch.html

Forecasting Software: AWS SageMaker

Amazon Web Services @awscloud

Follow

You can now use the DeepAR algorithm for more accurate time series forecasting in Amazon SageMaker! amzn.to/2GBKZSI

12:55 PM - 28 Mar 2018

```
style = {'description_width': 'initial'}
interact_manual(
    plot_interact,
    customer_id=IntSlider(min=0, max=369, value=181, style=style),
    forecast_day=IntSlider(min=0, max=100, value=21, style=style),
    confidence=IntSlider(min=51, max=99, value=80, step=5, style=style),
    show_samples=Checkbox(value=False),
    continuous_update=False
)
```

customer_id 181

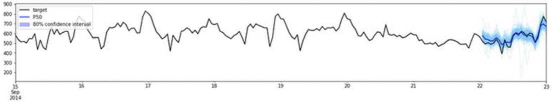
forecast_day 21

confidence 80

show_samples

Run Interact

```
calling served model to generate predictions for customer 181 starting at 2014-09-22 00:00:00
done in 1105 ms
```



Visit our booth for the DeepAR Demo!

THANK YOU FOR ATTENDING!

Website: https://lovvge.github.io/Forecasting-Tutorial-VLDB-2018/

aws

Carnegie Mellon

VLDB2018

References

Alberg, J. and Lipton, Z. C. (2017). Improving factor-based quantitative investing by forecasting company fundamentals. *arXiv preprint arXiv:1711.04837*.

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., and Petropoulos, F. (2017). Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1):60–74.

Azoff, E. M. (1994). *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc.

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Ben Taieb, S., Taylor, J. W., and Hyndman, R. J. (2017). Coherent probabilistic forecasts for hierarchical time series. In *Proceedings of the 34th International Conference on Machine Learning*.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R. (2017). An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*.

Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., and Lange, D. (2018). Deep learning for missing value imputation in tables with non-numerical data. *CIKM*.

Bińkowski, M., Marti, G., and Donnat, P. (2017). Autoregressive convolutional neural networks for asynchronous time series. *arXiv preprint arXiv:1703.04122*.

Boehm, M., Dusenberry, M. W., Eriksson, D., Evfimievski, A. V., Manshadi, F. M., Pansare, N., Reinwald, B., Reiss, F. R., Sen, P., Surve, A. C., and Tatikonda, S. (2016). Systemml: Declarative machine learning on spark. *Proc. VLDB Endow.*, 9(13).

References (cont.)

Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M., and Wang, Y. (2017). Probabilistic demand forecasting at scale. *PVLDB*, 10(12):1694–1705.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Brockwell, P. J. and Davis, R. A. (2013). *Time series: theory and methods*. Springer Science & Business Media.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. (2017). Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 77–87.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Crankshaw, D., Bailis, P., Gonzalez, J. E., Li, H., Zhang, Z., Franklin, M. J., Ghodsi, A., and Jordan, M. I. (2015). The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*.

References (cont.)

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. (2017). Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 613–627.

de Araujo, M. R., Ribeiro, P. M. P., and Faloutsos, C. (2017). Tensorcast: Forecasting with context using coupled tensors. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 71–80. IEEE.

Deng, D., Shahabi, C., Demiryurek, U., Zhu, L., Yu, R., and Liu, Y. (2016). Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1525–1534. ACM.

Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*, volume 38. OUP Oxford.

Flunkert, V., Salinas, D., and Gasthaus, J. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110*.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3604–3613.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pages 2199–2207.

Gately, E. (1995). *Neural networks for financial forecasting*. John Wiley & Sons, Inc.

Ghaderi, A., Sanandaji, B. M., and Ghaderi, F. (2017). Deep forecast: deep learning-based spatio-temporal forecasting. *arXiv preprint arXiv:1707.08110*.

References (cont.)

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268.

Harvey, A. C. (1990). *Forecasting, structural time series models and the Kalman filter*. Cambridge university press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hyndman, R. (2015). Exploring the feature space of large collections of time series. *WorkÂŋshop on FronÂŋtiers in FuncÂŋtional Data AnalyÂŋsis, Banff, Canada*.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. (2008). *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

Hyndman, R. J. and Athanasopoulos, G. (2017). Forecasting: Principles and practice. *www. otexts. org/fpp.*, 987507109.

Hyndman, R. J., Khandakar, Y., et al. (2007). *Automatic time series for forecasting: the forecast package for R*. Number 6/07. Monash University, Department of Econometrics and Business Statistics.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688.

Jing, P., Su, Y., Jin, X., and Zhang, C. (2018). High-order temporal correlation model learning for time-series prediction. *IEEE Transactions on Cybernetics*, (99):1–13.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. (2017). Deep variational bayes filters: Unsupervised learning of state space models from raw data. *ICLR*.

References (cont.)

Kolassa, S. and Schuetz, W. (2007). Advantages of the mad/mean ratio over the mape. *Foresight. Applied Forecasting Journal*.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *AAAI*, pages 2101–2109.

Laptev, N., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. In *ICML Time Series Workshop*, number 34, pages 1–5.

Larson, P. D., Simchi-Levi, D., Kaminsky, P., and Simchi-Levi, E. (2001). Designing and managing the supply chain: Concepts, strategies, and case studies. *Journal of Business Logistics*, 22(1):259–261.

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

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*.

Makridakis, S. G., Wheelwright, S. C., and Hyndman, R. J. (1998). Forecasting: Methods and applications.

Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T., and Yoshikawa, M. (2012). Fast mining and forecasting of complex time-stamped events. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 271–279. ACM.

References (cont.)

Matsubara, Y., Sakurai, Y., van Panhuis, W. G., and Faloutsos, C. (2014). Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, pages 105–114. ACM.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016). Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(34):1–7.

Miller, J. and Hardt, M. (2018). When recurrent models don't need to be recurrent. *arXiv preprint arXiv:1805.10369*.

Modi, A. N., Koo, C. Y., Foo, C. Y., Mewald, C., Baylor, D. M., Breck, E., Cheng, H.-T., Wilkiewicz, J., Koc, L., Lew, L., Zinkevich, M. A., Wicke, M., Ispir, M., Polyzotis, N., Fiedel, N., Haykal, S. E., Whang, S., Roy, S., Ramesh, S., Jain, V., Zhang, X., and Haque, Z. (2017). Tfx: A tensorflow-based production-scale machine learning platform. In *KDD 2017*.

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*.

Papalexakis, E. E., Faloutsos, C., and Sidiropoulos, N. D. (2012). Parcube: Sparse parallelizable tensor decompositions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer.

Polyzotis, A., Zinkevich, M. A., Whang, S., and Roy, S. (2017). Data management challenges in production machine learning. pages 1723–1726.

Roberts, L., Razoumov, L., Su, L., and Wang, Y. (2017). Gini regularized optimal transport with an application to spatio-temporal forecasting. *NIPS Workshop on Optimal Transport*.

References (cont.)

Schelter, S., Boese, J.-H., Kirschnick, J., Klein, T., and Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In *NIPS Workshop ML Systems*.

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., and Grafberger, A. (2018). Automating large-scale data quality verification. *PVLDB*, 11.

Schelter, S., Palumbo, A., Quinn, S., Marthi, S., and Musselman, A. (2016). Samsara: Declarative machine learning on distributed dataflow systems. In *NIPS Workshop MLSystems*.

Scott, S. L. and Varian, H. R. (2014). Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1-2):4–23.

Seeger, M., Rangapuram, S., Wang, Y., Salinas, D., Gasthaus, J., Januschowski, T., and Flunkert, V. (2017). Approximate bayesian inference in linear state space models for intermittent demand forecasting at scale. *arXiv preprint arXiv:1709.07638*.

Seeger, M. W., Salinas, D., and Flunkert, V. (2016). Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pages 4646–4654.

Simchi-Levi, D., Chen, X., and Bramel, J. (2013). *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*.

Snyder, R. D., Ord, J. K., and Beaumont, A. (2012). Forecasting the intermittent demand for slow-moving inventories: A modelling approach. *International Journal of Forecasting*, 28(2):485–496.

Sparks, E. R., Venkataraman, S., Kaftan, T., Franklin, M. J., and Recht, B. (2017). Keystoneml: Optimizing pipelines for large-scale advanced analytics. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*.

References (cont.)

Takeuchi, K., Kashima, H., and Ueda, N. (2017). Autoregressive tensor factorization for spatio-temporal predictions. In *ICDM*, pages 1105–1110. IEEE.

Tang, Z., De Almeida, C., and Fishwick, P. A. (1991). Time series forecasting using neural networks vs. box-jenkins methodology. *Simulation*, 57(5):303–310.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

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. *arXiv preprint arXiv:1711.11053*.

Wickramasuriya, S. L., Athanasopoulos, G., and Hyndman, R. J. (2018). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, pages 1–45.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810.

Yu, H.-F., Rao, N., and Dhillon, I. S. (2016). Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS*, pages 847–855.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., and Liu, Y. (2017a). 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.

References (cont.)

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. (2017b). Long-term forecasting using tensor-train rnns. *arXiv preprint arXiv:1711.00073*.

Zaheer, M., Ahmed, A., and Smola, A. J. (2017). Latent lstm allocation: Joint clustering and non-linear dynamic modeling of sequence data. In *International Conference on Machine Learning*, pages 3967–3976.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62.

Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E. P., and Smola, A. J. (2017). State space lstm models with particle mcmc inference. *arXiv preprint arXiv:1711.11179*.