Deep Learning for Forecasting

ISF 2021: Deep Learning Workshop

Lorenzo Stella, Tim Januschowski

AWS AI Labs

https://github.com/awslabs/gluon-ts

June 28, 2021

Time Series @ AWS AI Labs

Based on joint work over the last 8 years with

Francois-Xavier Aubet	Konstantinos Benidis	Michael Bohlke-Schneider	Laurent Callot
Lifan Chen	*Valentin Flunkert*	Hilaf Hasson	Tim Januschowski
Peihong Jiang	*Andrey Kan*	Shubham Kapoor	Baris Kurt
Pedro Eduardo Mercado Lopez	Youngsuk Park	Mostafa Rahmani	Syama Rangapuram
Danielle Robinson	Lorenzo Stella	*Caner Turkmen*	Bernie Wang
Jasper Zschiegner	Richard Kurle	Jan Gasthaus	

What do we do?

AWS Services & Open Source: Amazon Forecast, Amazon DevOps Guru, Amazon Lookout for Metrics, Amazon Sagemaker DeepAR, GluonTS, ...

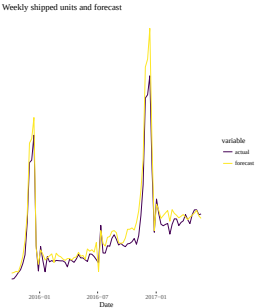
Forecasting at AWS AI Labs

We look after ...

- DeepAR in Amazon SageMaker
  https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html
- Amazon Forecast https://aws.amazon.com/forecast/
- GluonTS https://github.com/awslabs/gluon-ts
- Amazon internal applications like Labor Planning for Fulfillment Centers
  https://dl.acm.org/doi/abs/10.1145/3399579.3399869
- AWS services: Amazon Lookout for Metrics (anomaly detection for business metrics), Amazon DevOps Guru (anomaly detection for DevOps)
- and more!

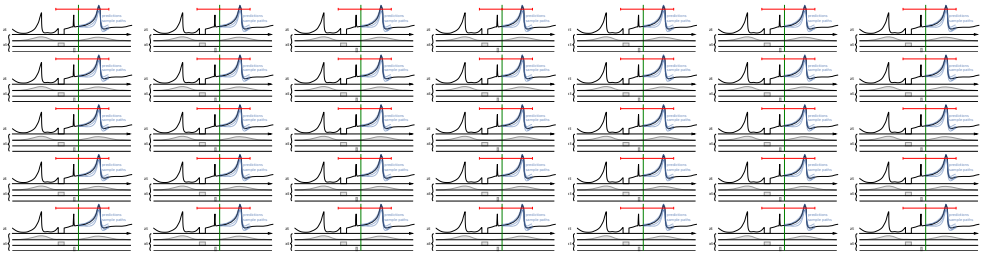
```
pip install --upgrade mxnet==1.8.0 torch==1.8.0
pip install git+https://github.com/awslabs/gluon-ts.git
```

Neural Networks: no silver bullet



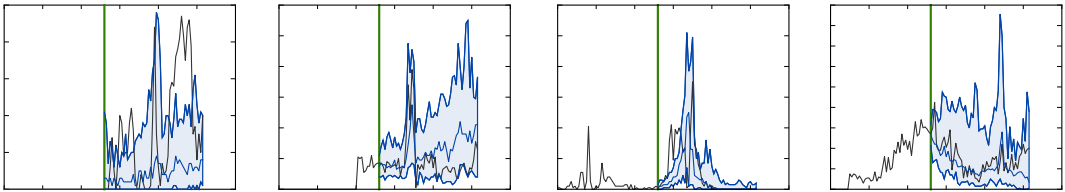
- Forecasting Problem: predict overall Amazon retail demand years into the future.
- Decision Problems: topology planning, market entry/segment analyses
- $\Rightarrow$ we call them *strategic* forecasting problems
- neural networks probably won't be the first/best/appropriate/... method: not enough "control", over fitting, hard to fit in domain knowledge, etc.

Contemporary business forecasting problems



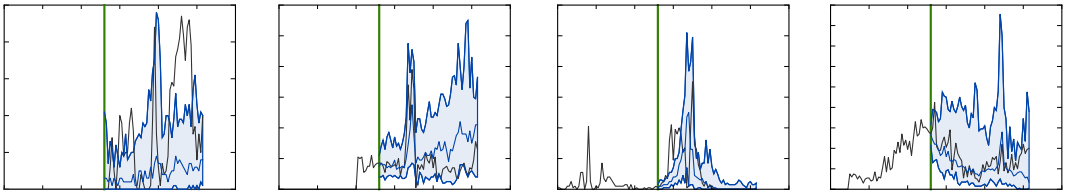
- Examples: demand for products of a retailer, workforce cohorts of a company in its locations, compute capacity needs per region and server type
- $\Rightarrow$ we call them *operational* forecasting problems
- neural networks probably a strong contender: patterns are complex but enough data is available to learn them, so no over fitting concerns; too many time series and edge cases to design specific methods for, etc.

Operational forecasting problems: characteristics & approaches



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

Why are neural networks good?

- easy to learn across multiple time series (*global* models)
- little manual feature engineering needed
- simplify (software) system

Structure of the Workshop (4h): Lecture & Practical Session

- Deep Learning (40 mins). 6pm
  - Gentle Intro: from linear methods to deep learning
  - Optimization challenges
  - Different architectures for different problem types
  - Probabilistic prediction
  - Q&A and/or break (5 mins)
- Deep Learning for Forecasting (40 mins). 6:45pm
  - Discriminative vs Generative Models
  - Deep dive into DeepAR
  - Categorization of state-of-the-art
  - Results
  - Q&A and/or break (5 mins)
- Advanced Topics: Multivariate Forecasting (20mins). 7:30pm.
- break (20 mins, 7:50pm)
- Practical session (1:40h, start at 8:10 pm)

Goals

- understand about the basics of deep learning
- if you know deep learning, understand how to get probabilistic predictions
- in-depth discussion of two simple neural networks architectures
- exposure to what's currently being worked on in the neural forecasting literature
- some hands-on experience [practical part & self study]

what we won't do here:

- broad survey of the state-of-the-art → https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/ or Benidis et al. [2020]
- discuss other machine learning techniques like gradient boosted trees

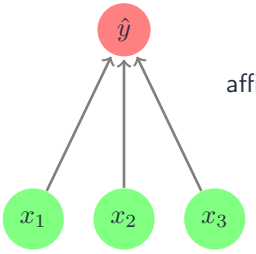
Part I: Deep Learning

From Linear Methods to Deep Learning

Linear methods: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n, \quad \hat{y} = \quad \mathbf{w}^T \boldsymbol{x} + b$

From Linear Methods to Deep Learning

Linear methods: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n, \quad \hat{y} = \quad \mathbf{w}^T\boldsymbol{x} + b$



affine function

$\mathbf{w}, b$

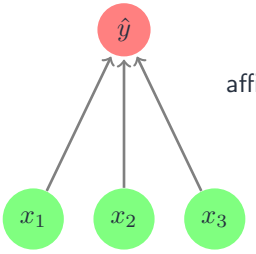
Output: Linear in both inputs $\boldsymbol{x}$ and weights w (modulo shift b)

$$\hat{y} = \quad \mathbf{w}^T\boldsymbol{x} + b$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

From Linear Methods to Deep Learning

Linear methods: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n, \quad \hat{y} = \quad \mathbf{w}^T\boldsymbol{x} + b$



affine function

$\mathbf{w}, b$

Output: Linear in both inputs $\boldsymbol{x}$ and weights w (modulo shift b)

$$\hat{y} = \quad \mathbf{w}^T\boldsymbol{x} + b$$

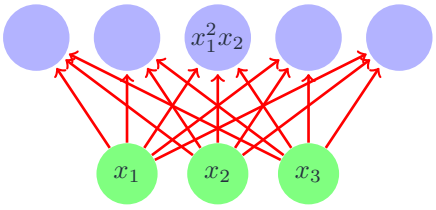
Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Estimation/Optimization: Analytical solution for the unknown $\mathbf{w}$ depending on the loss

Linear Methods with Non-Linear Features

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features

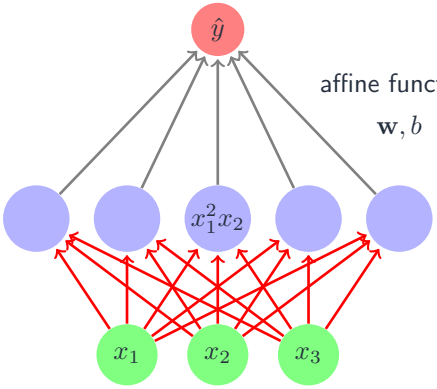


Manually engineered features:

$$\Phi(x) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

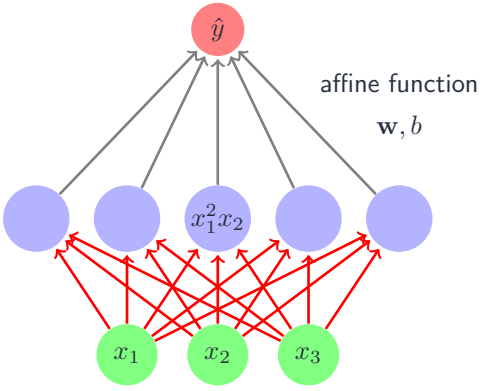
$$\hat{y} = \mathbf{w}^T \Phi(x) + b$$

Manually engineered features:

$$\Phi(x) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

$$\hat{y} = \mathbf{w}^T \Phi(x) + b$$

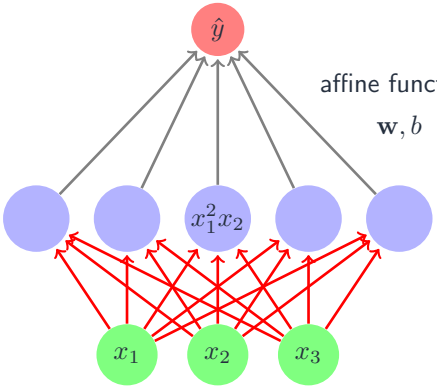
Manually engineered features:

$$\Phi(x) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Estimation/Optimization: still the same as linear models

Linear Methods with Non-Linear Features



Output: Linear model of non-linear features Φ

$$\hat{y} = \mathbf{w}^T \Phi(x) + b$$

Manually engineered features:

$$\Phi(x) = [x_1, x_2, x_1^2 x_2, x_1 x_3, x_3]$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

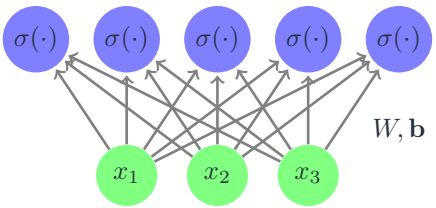
Estimation/Optimization: still the same as linear models

Kernel Methods: Non-linear feature map via a positive definite kernel

Why not learn the non-linear features?

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Why not learn the non-linear features?

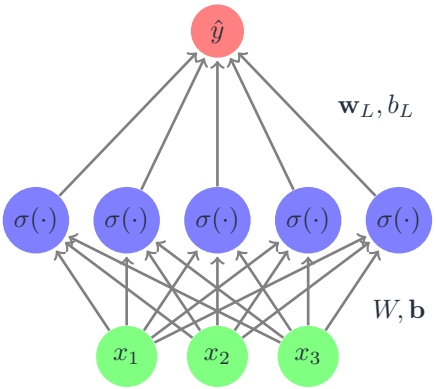


Learned features: Affine functions of inputs followed by a non-linear activation

$$\phi_j(\boldsymbol{x}; W_j, b_j) = \sigma(W_j^\top \boldsymbol{x} + b_j)$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Why not learn the non-linear features?



Output: Linear model of (learned) non-linear features Φ

$$\hat{y} = \mathbf{w}_L^T \Phi(x; W, \mathbf{b}) + b_L$$

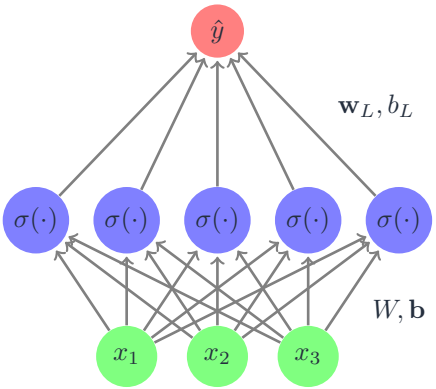
Learned features: Affine functions of inputs followed by a non-linear activation

$$\phi_j(\boldsymbol{x}; W_j, b_j) = \sigma(W_j^\top \boldsymbol{x} + b_j)$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Neural Networks (Non-Linear Methods)

Why not learn the non-linear features?



Estimation/Optimization: Non-convex!

Output: Linear model of (learned) non-linear features Φ

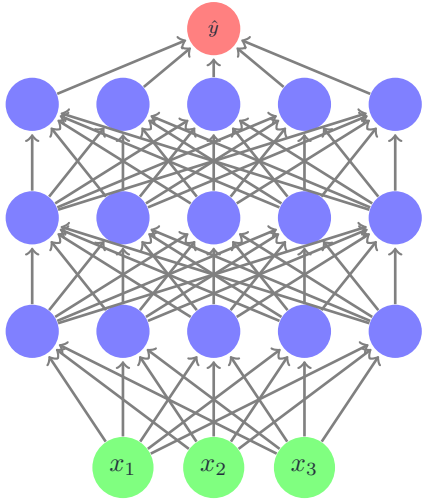
$$\hat{y} = \mathbf{w}_L^T \Phi(x; W, \mathbf{b}) + b_L$$

Learned features: Affine functions of inputs followed by a non-linear activation

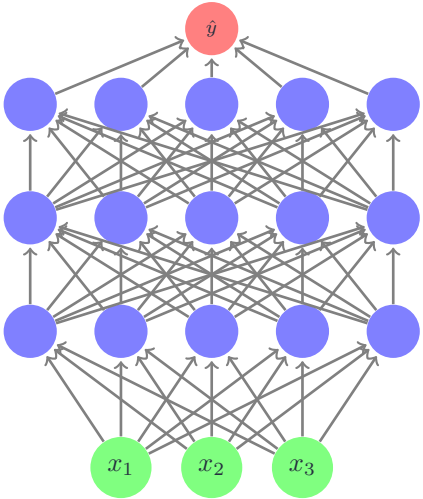
$$\phi_j(\boldsymbol{x}; W_j, b_j) = \sigma(W_j^\top \boldsymbol{x} + b_j)$$

Inputs: $\boldsymbol{x} = \{x_1, x_2, x_3\}$

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)

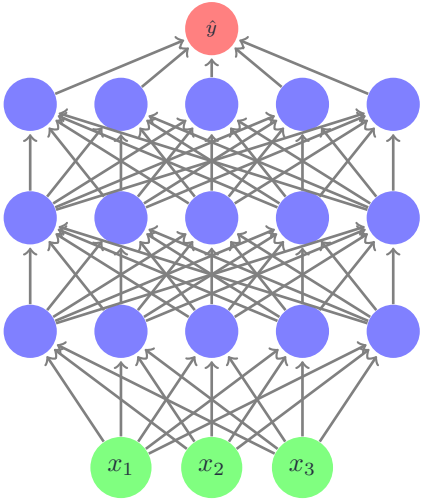


Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



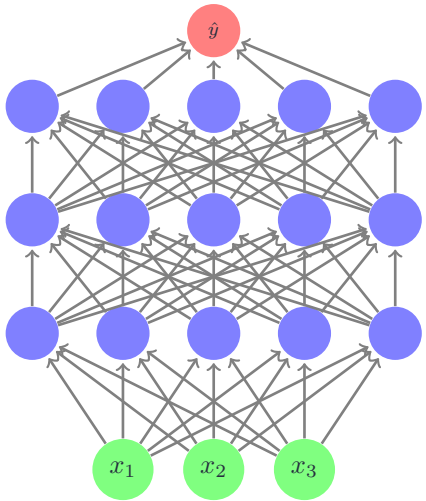
- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear activation function
- Last hidden layer provides "feature embeddings" useful for transfer learning

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



- Each neuron in a hidden layer computes an affine function of the previous layer, followed by a non-linear activation function
- Last hidden layer provides "feature embeddings" useful for transfer learning
- Flexible general function estimators
- More (and larger) hidden layers $\rightarrow$ more complex functions

Feed-Forward Neural Networks/Multi-layer Perceptrons (MLPs)



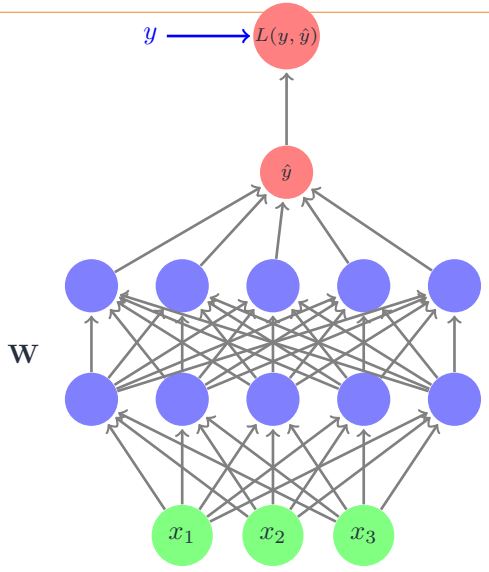
Pros

- Can learn complex input-output relationships
- Less manual feature engineering

Cons

- More data needed for training
- Careful tuning (e.g., regularization, learning rate)
- Sensitive to scaling of inputs

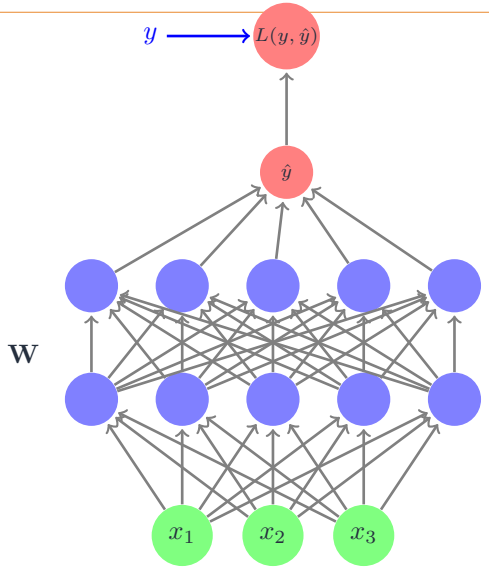
Training Neural Networks



- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\boldsymbol{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\boldsymbol{x}_i; \mathbf{W})$$

Training Neural Networks

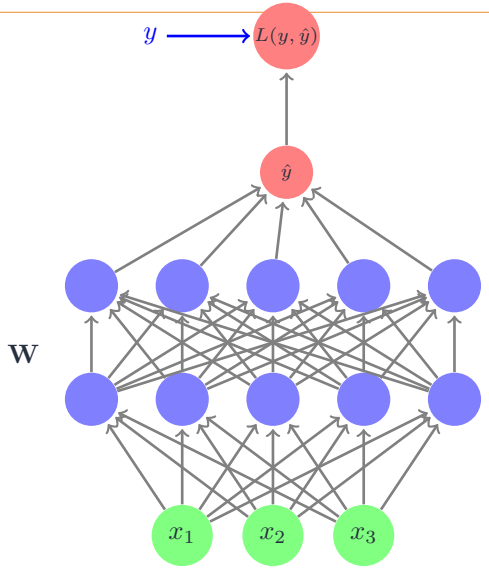


- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\boldsymbol{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\boldsymbol{x}_i; \mathbf{W})$$

- No analytical (closed-form) solution

Training Neural Networks



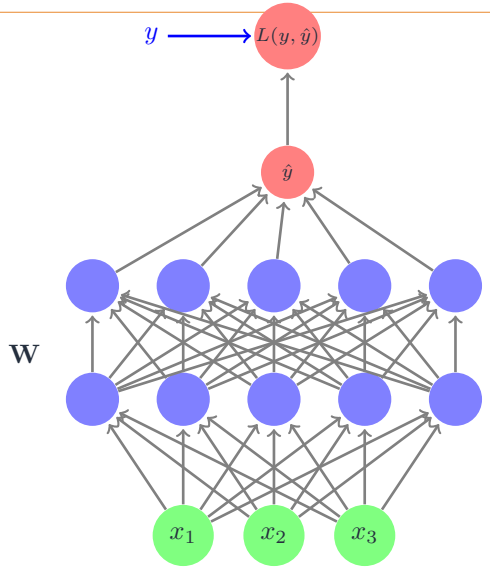
- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\boldsymbol{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\boldsymbol{x}_i; \mathbf{W})$$

- No analytical (closed-form) solution
- Numerical optimization, e.g., (stochastic) gradient descent

$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \nabla \sum_i L\left(y_i, f(\boldsymbol{x}_i; \mathbf{W}^{t-1})\right)$$

Training Neural Networks



- **Loss function:** Minimize some notion of error $L(y, \hat{y})$ on a training set $D = \{\boldsymbol{x}_i, y_i\}$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_i L(y_i, \hat{y}_i), \quad \hat{y}_i = f(\boldsymbol{x}_i; \mathbf{W})$$

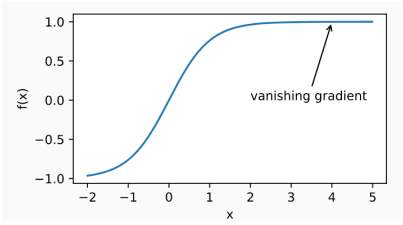
- No analytical (closed-form) solution
- Numerical optimization, e.g., (stochastic) gradient descent

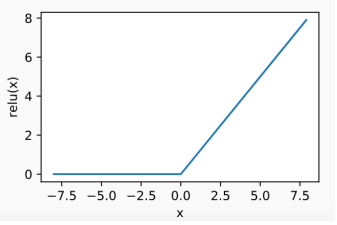
$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \nabla \sum_i L\left(y_i, f(\boldsymbol{x}_i; \mathbf{W}^{t-1})\right)$$

- **Backpropagation:** computes gradients efficiently (automatic differentiation via chain rule)

Optimization Challenges in Deep Learning

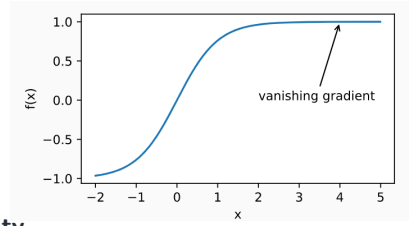
Vanishing gradient: Choice of non-linearities and architectures (RNN)

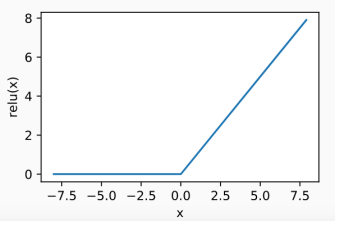




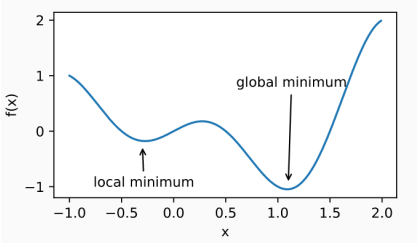
Optimization Challenges in Deep Learning

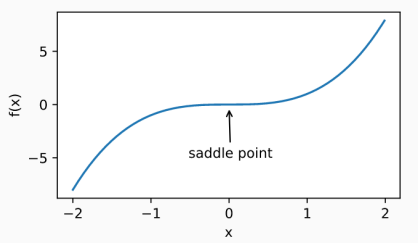
Vanishing gradient: Choice of non-linearities and architectures (RNN)





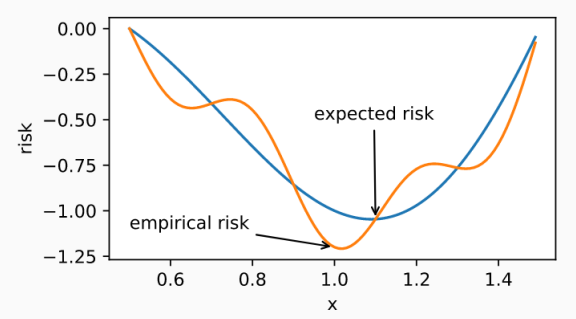
Non-convexity





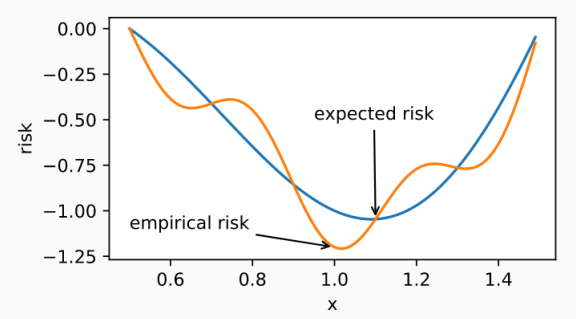
What do you want to optimize?

Goal of machine learning: Minimize generalization error / expected risk
Practice: Minimize a surrogate: empirical risk on a *training set*



What do you want to optimize?

Goal of machine learning: Minimize generalization error / expected risk
Practice: Minimize a surrogate: empirical risk on a *training set*



Regularization: Explicit penalty in a convex model vs Approximate solution in a non-convex model

Stochastic Gradient Descent

Idea: Use a randomly sampled training example to update the weights

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \nabla L \left(y_i, f(\boldsymbol{x}_i; \mathbf{W}^t) \right)$$

Motivation:

- Reduce computational cost: error in estimation proportional to $\frac{1}{\sqrt{n}}$
- Better generalization performance
- Avoiding local minima

Stochastic Gradient Descent

Idea: Use a randomly sampled training example to update the weights

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \nabla L\left(y_i, f(\boldsymbol{x}_i; \mathbf{W}^t)\right)$$

Motivation:

- Reduce computational cost: error in estimation proportional to $\frac{1}{\sqrt{n}}$
- Better generalization performance
- Avoiding local minima

Practice:

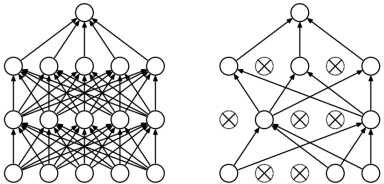
- Mini-batch stochastic gradient descent

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta_t \frac{1}{|\mathbf{B^t}|} \sum_{i \in \mathbf{B^t}} \nabla L\left(y_i, f(\boldsymbol{x}_i; \mathbf{W}^t)\right)$$

- How to choose size of the batch?
- Popular algorithm: Adam

Further practical questions

Regularization via Dropout



(a) Standard Neural Net (b) After applying dropout.

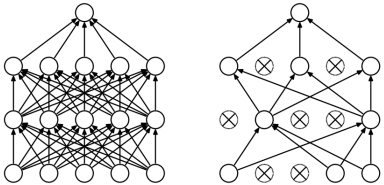
Randomly drop neurons (and their connections) during training [Srivastava et al., 2014]

Initialization of weights, how to adapt the step size, how to tune parameters, . . . Many more!

Big question: which neural architecture is best?

Further practical questions

Regularization via Dropout



(a) Standard Neural Net (b) After applying dropout.

Randomly drop neurons (and their connections) during training [Srivastava et al., 2014]

Initialization of weights, how to adapt the step size, how to tune parameters, . . . Many more!

Big question: which neural architecture is best? Next: Convolutional and recurrent neural networks as two examples.

Convolutional Neural Networks

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

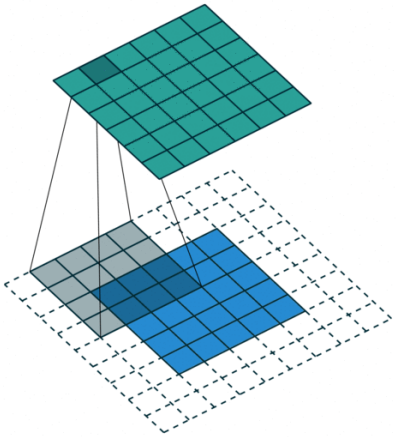


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

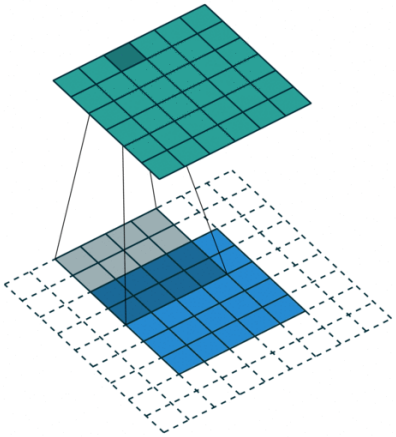


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

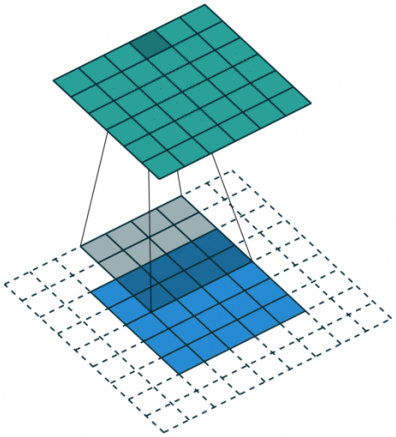


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

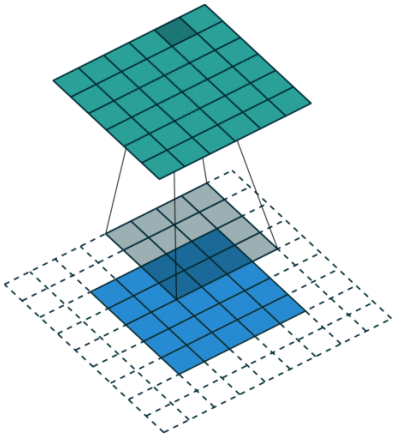


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

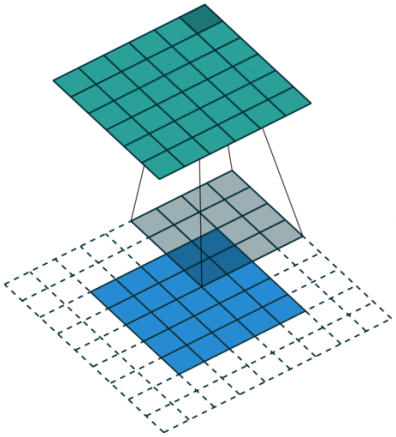


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

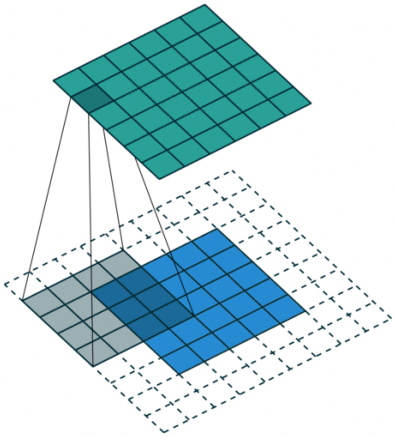


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

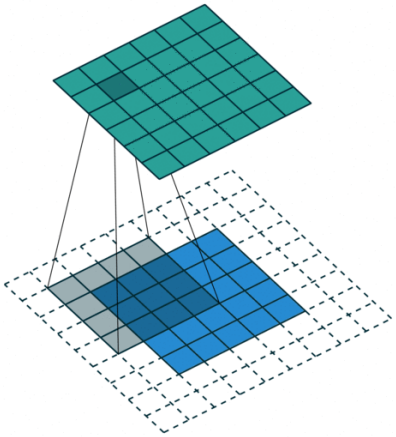


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

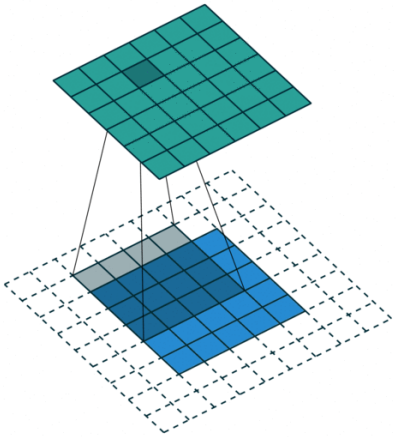


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks [Le Cun, 1989]

- Convolutional Neural Networks (CNNs) = NNs that use *convolutional layers*
- CNNs with 2D convolutions are extremely successful in computer vision applications

$$\Longrightarrow \text{encode spatial invariance}$$

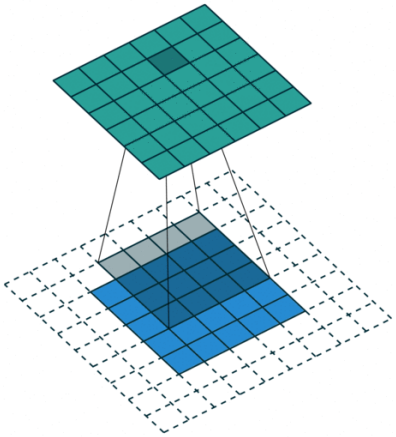
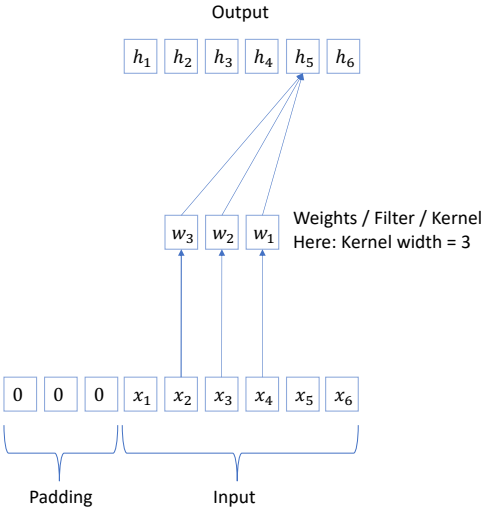


Figure credit: Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning; https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers

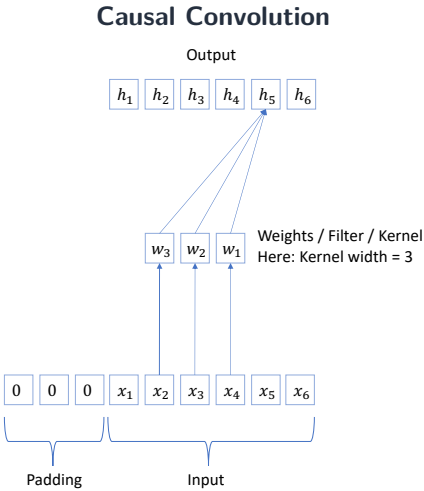


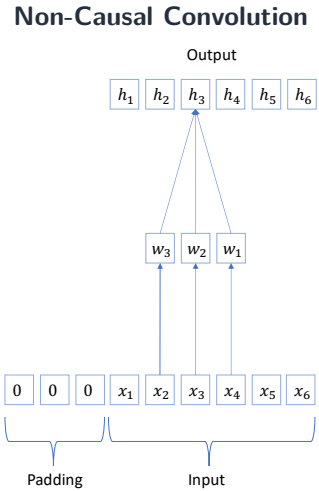
- The output h_j of a neuron j in a convolution layer is a discrete convolution of the inputs $\mathbf{x}$ with the layer's weigths/filter $\mathbf{w}$.
- For a one-dimensional convolution with a kernel with width D we have

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

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

Causal vs. Non-Causal Convolution





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

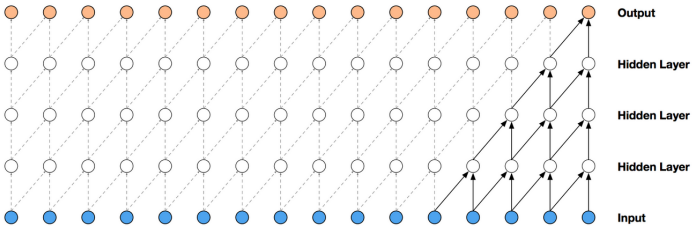


Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

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

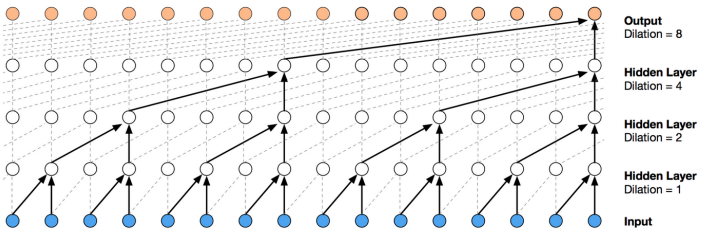
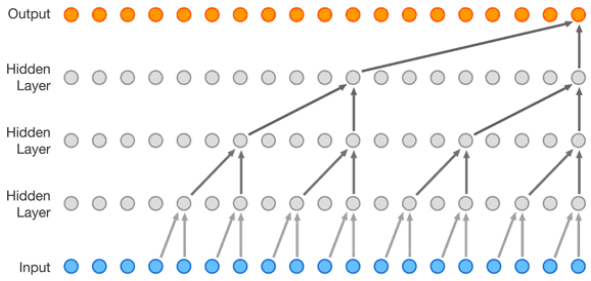


Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

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]

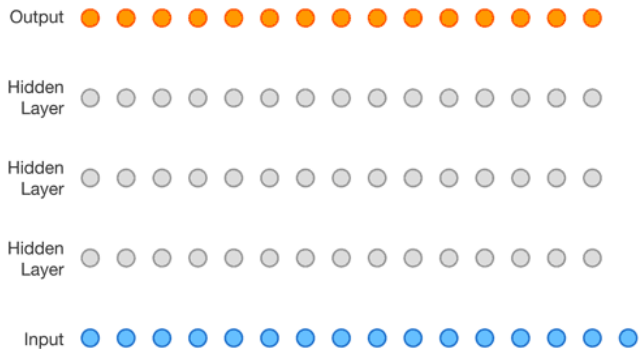
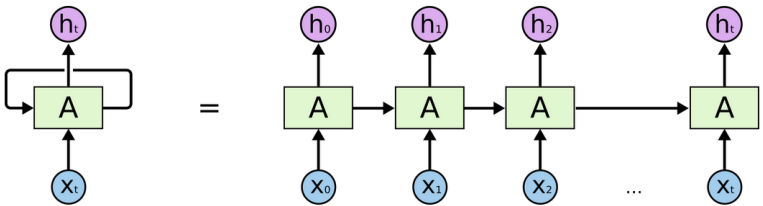


Figure credit: WaveNet: A Generative Model for Raw Audio; https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Convolutional Neural Networks

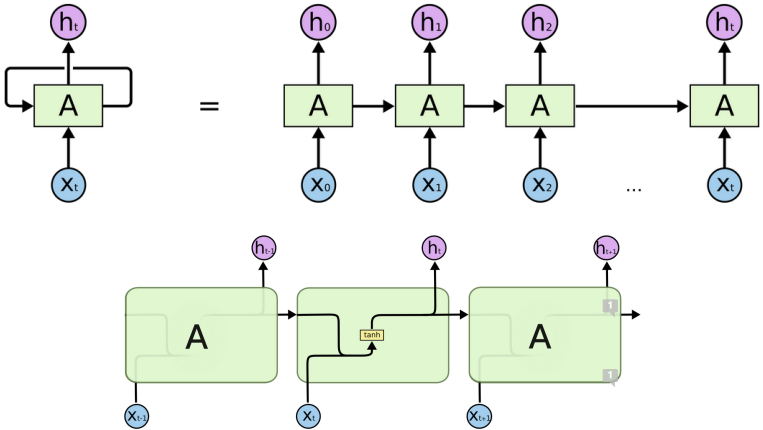
Recurrent Neural Networks (RNN)

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/



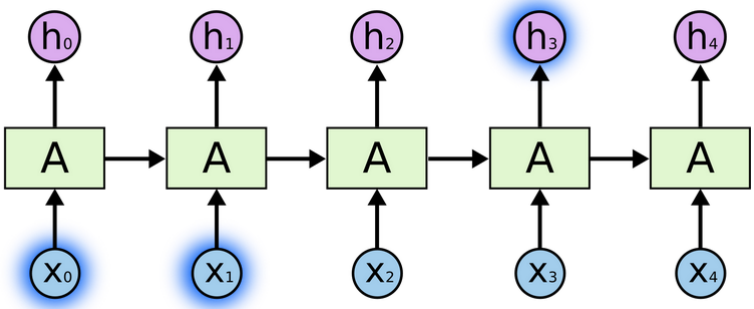
Recurrent Neural Networks (RNN)

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/



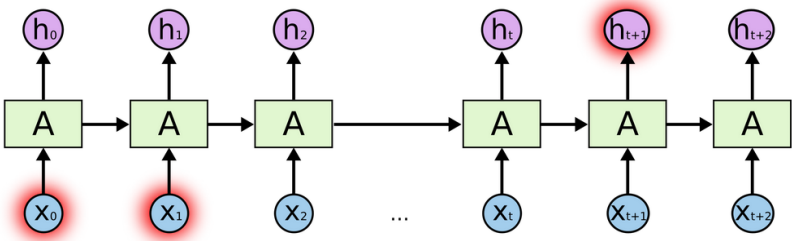
Problem of Long Term Dependencies

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

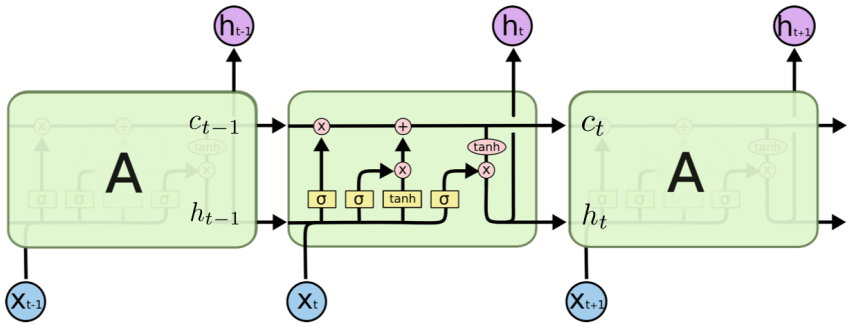


Problem of Long Term Dependencies

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

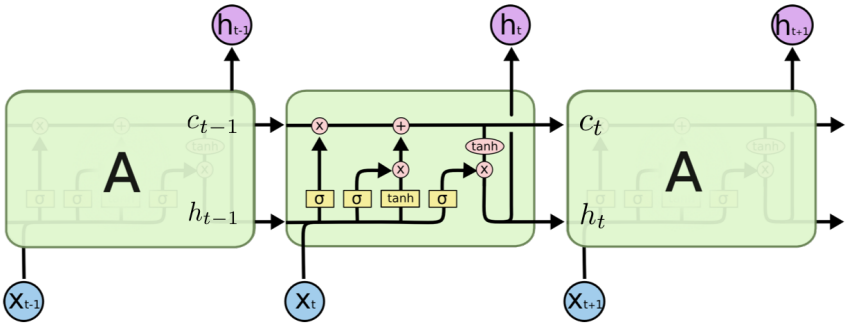


Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]



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

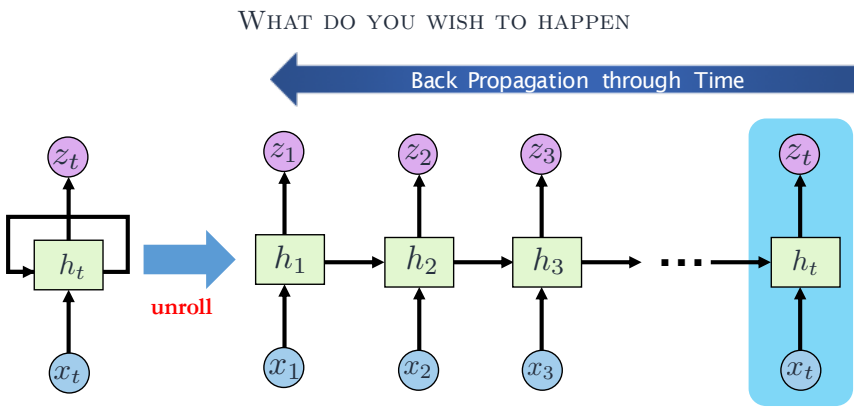
Long Short-Term Memory (LSTM): What?



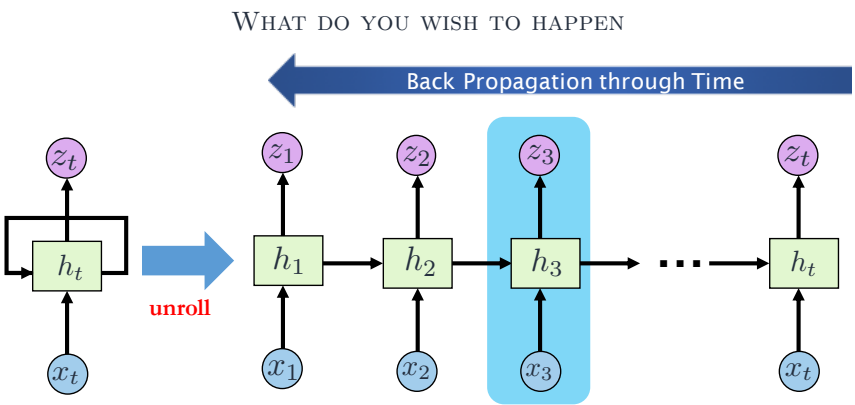
HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

current state = forgot gate $\times$ old stuff + input gate $\times$ new stuff

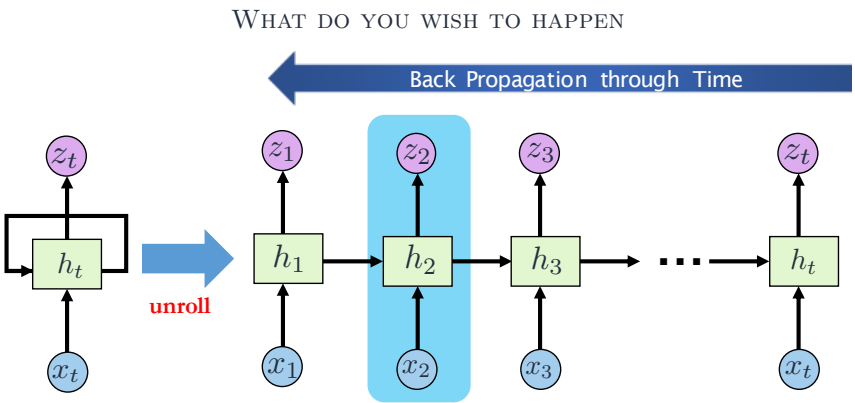
RNN: Vanishing Gradient Problem



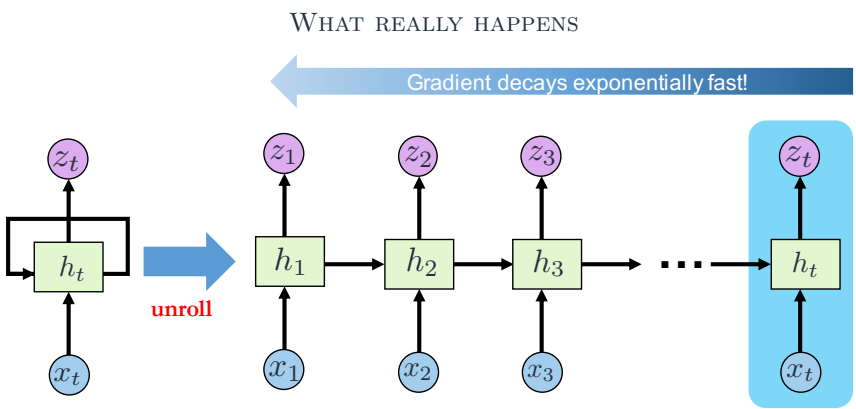
RNN: Vanishing Gradient Problem



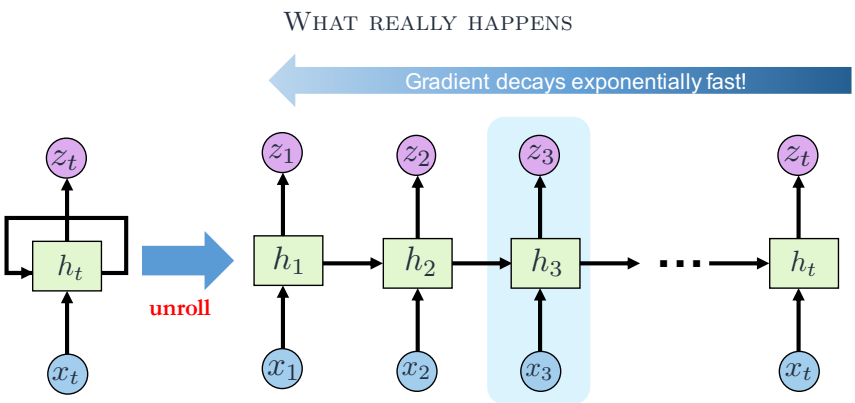
RNN: Vanishing Gradient Problem



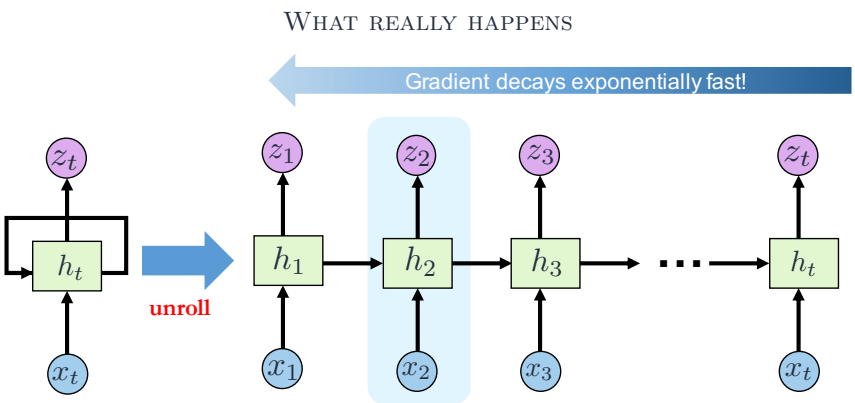
RNN: Vanishing Gradient Problem



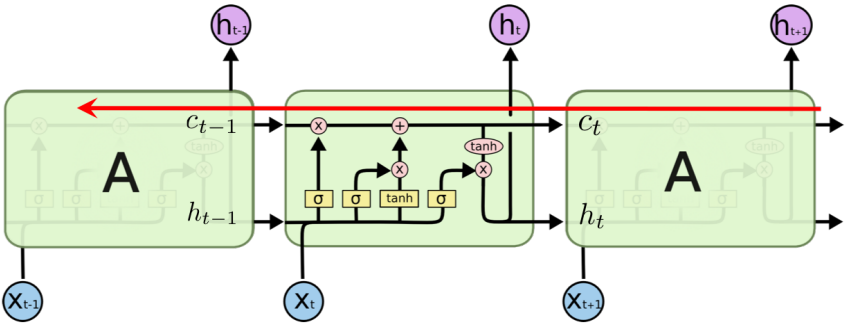
RNN: Vanishing Gradient Problem



RNN: Vanishing Gradient Problem



LSTMs Prevent Vanishing Gradient!



There is at least one path where the gradient does not vanish!

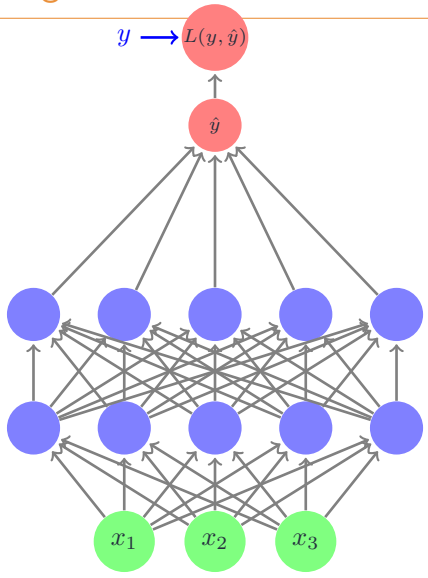
Probabilistic Predictions

Probabilistic Prediction with Deep Learning

Point prediction: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$

$$\hat{y} = f(\boldsymbol{x}; \theta)$$

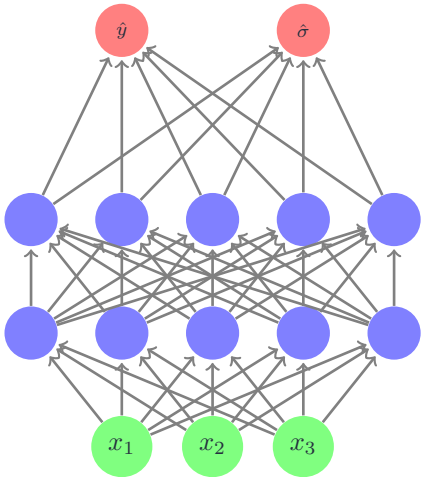
$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_i L(y_i, \hat{y}_i)$$



Probabilistic Prediction with Deep Learning

Probabilistic prediction: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$

$$y = f(\boldsymbol{x}; \theta) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_x^2),$$
$$p(y|\boldsymbol{x}; \theta) = \mathcal{N}(y| \underbrace{f(\boldsymbol{x}; \theta)}_{\text{mean}}, \underbrace{\sigma_x^2}_{\text{variance}}),$$



Probabilistic Prediction with Deep Learning

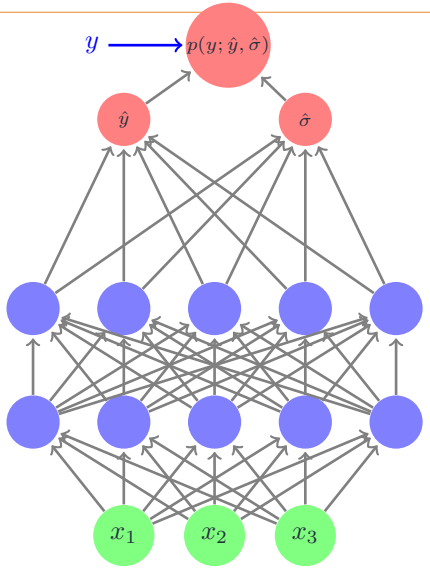
Probabilistic prediction: $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$

$$y = f(\boldsymbol{x}; \theta) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_x^2),$$
$$p(y|\boldsymbol{x}; \theta) = \mathcal{N}(y| \underbrace{f(\boldsymbol{x}; \theta)}_{\text{mean}}, \underbrace{\sigma_x^2}_{\text{variance}}),$$

Maximum likelihood principle:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_i \log p(y_i|\boldsymbol{x}_i; \theta)$$

(Auto differentiation if the likelihood is differentiable)



Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y: $p(y; \theta)$ with parameters θ
  1. Regression: Gaussian, Poisson, Zero-inflated distributions etc.
  2. Classification: Bernoulli, Categorical
  3. Structured prediction: e.g., for sequences: State space model, Auto-regressive model etc.

Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y: $p(y; \theta)$ with parameters θ
  1. Regression: Gaussian, Poisson, Zero-inflated distributions etc.
  2. Classification: Bernoulli, Categorical
  3. Structured prediction: e.g., for sequences: State space model, Auto-regressive model etc.

- Make θ (can have more than one parameter) the output of the neural network:

$$\theta = \mathrm{NN}(\boldsymbol{x}; W)$$

  Here the type of the neural network NN is chosen according to the input features $\boldsymbol{x}$

Recipe for Probabilistic Prediction with Deep Learning

- Choose appropriate probabilistic model for the targets y: $p(y; \theta)$ with parameters θ
  1. Regression: Gaussian, Poisson, Zero-inflated distributions etc.
  2. Classification: Bernoulli, Categorical
  3. Structured prediction: e.g., for sequences: State space model, Auto-regressive model etc.

- Make θ (can have more than one parameter) the output of the neural network:

$$\theta = \mathrm{NN}(\boldsymbol{x}; W)$$

  Here the type of the neural network NN is chosen according to the input features $\boldsymbol{x}$

- **Training:** Learn the neural network weights W by maximizing the log-likelihood of θ given the targets y and features $\boldsymbol{x}$

$$W^* = \underset{W}{\mathrm{argmax}} \sum_i \log p(y_i | \boldsymbol{x}_i; \underbrace{\mathrm{NN}(\boldsymbol{x}_i; W)}_{\theta})$$

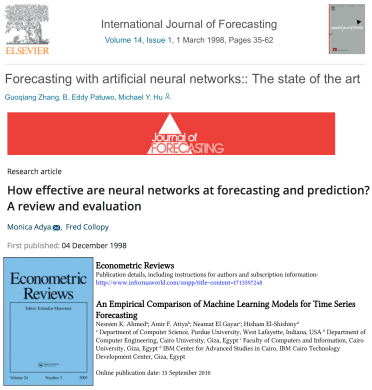
One other, straight-forward alternative: estimate specific quantiles directly (use quantile loss/pinball loss)

Q & A and/or Coffee Break (back at 6:45pm)

GluonTS github repository: https://github.com/awslabs/gluon-ts

Part II: Deep Learning for Forecasting

Neural Network Forecasting: Old and New – Timeline

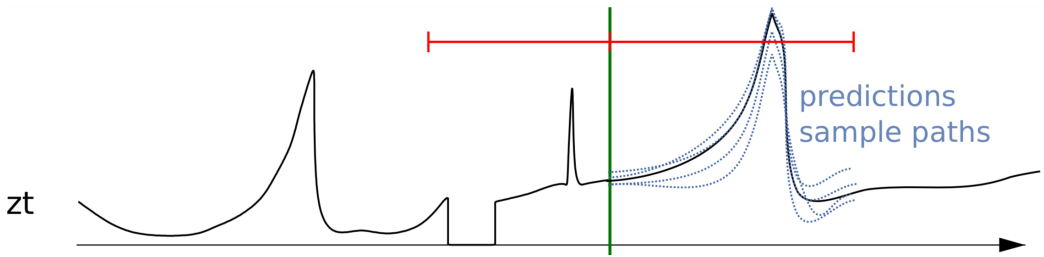


- 1969 Weather forecasting with adaptive linear neurons (Hu)
- 1986 Backpropagation (Rumelhart et al.)
- 1988 NNs using backpropagation applied to forecasting; positive results (Werbos)
- 199x Many authors applying mostly feed-forward models to various forecasting problem (single time series, *local* models)
- 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)
- 2017 Transformer models (BERT) take over NLP
- 2018 M4 competition: combination of NNs and classical techniques wins
- 2020 M5 competition: "ML" techniques win. NBEATS in 2nd place, DeepAR in 3rd place

General Setup

Predict the future behaviour of a time series given its past

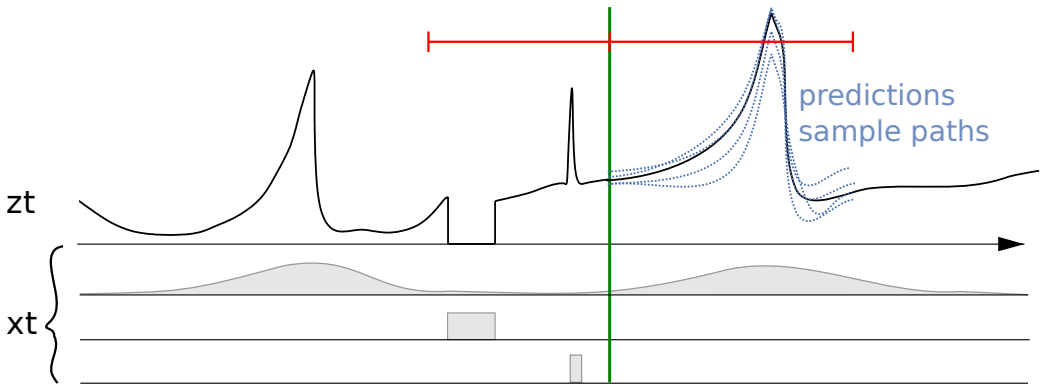
$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$



General Setup

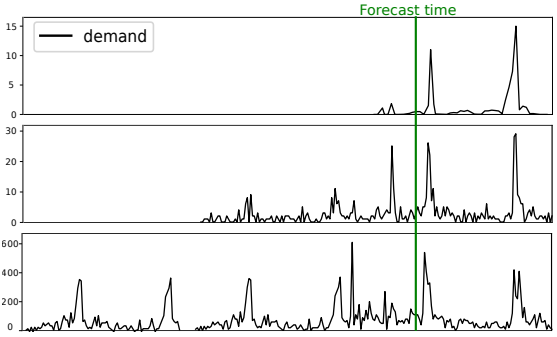
Predict the future behaviour of a time series given its past

$$z_1, z_2, \ldots, z_T \Longrightarrow P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$



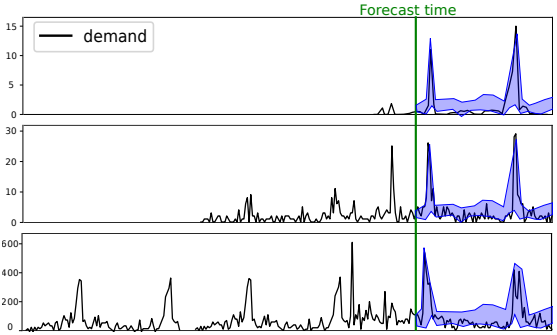
Deep Learning for Forecasting

- Not a single item (time series) but a large collection of related time series
- Exploit information across related time series by sharing parameters *globally*



Deep Learning for Forecasting

- Not a single item (time series) but a large collection of related time series
- Exploit information across related time series by sharing parameters *globally*
- Probabilistic prediction



Neural Networks: use as *global* models

- confusing terminology!
- let Z be a panel of time series, so $Z = \{\mathbf{z}_i = \{z_1, z_2, \ldots, z_t\} \in \mathbb{R}^t\}_{i=1,\ldots,n}$ all of which we want to forecast. X are co-variates/features.
- *local model*: $P(z_{i,t+1}|z_{i,1}, \ldots, z_{i,t}; X_i, \theta_i)$. Important is i in θ_i. A single *local* model per time series (e.g., ETS, ARIMA)
- *global model*: $P(z_{i,t+1}|z_{i,1}, \ldots, z_{i,t}; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$. Note: θ_i corresponds to time series z_i, but only one weight vector W (for the entire panel Z).

Neural Networks: use as *global* models

- confusing terminology!
- let Z be a panel of time series, so $Z = \{\mathbf{z}_i = \{z_1, z_2, \ldots, z_t\} \in \mathbb{R}^t\}_{i=1,\ldots,n}$ all of which we want to forecast. X are co-variates/features.
- *local model*: $P(z_{i,t+1}|z_{i,1}, \ldots, z_{i,t}; X_i, \theta_i)$. Important is i in θ_i. A single *local* model per time series (e.g., ETS, ARIMA)
- *global model*: $P(z_{i,t+1}|z_{i,1}, \ldots, z_{i,t}; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$. Note: θ_i corresponds to time series z_i, but only one weight vector W (for the entire panel Z).
- it gets more complicated!
- we are actually interested in $P(Z_{t+1}|Z_1, \ldots, Z_t; X_i, \theta_i), \quad \theta_i = \Psi(X_i, W)$ or *multi-variate models*. We cover this at 7:30pm.

Neural Networks: use as *global* models

- each time series z is one instance in the training set
- the weights of the neural network are trained over all time series
- this is the natural way to handle neural networks in other sequence tasks such as natural language processing
- until recently somewhat foreign to forecasters: e.g., 1998 overview article [Zhang et al., 1998] considered local neural networks, similar to [Makridakis et al., 2018]
- known under many names: cross-learning, multi-task learning, panel data models,...
- recent theoretic justification: Principles and Algorithms for Forecasting Groups of Time Series: Locality and Globality by Montero-Manso and Hyndman

Generative vs Discriminative Models for Forecasting

Given a time series: $z_1, z_2, \ldots, z_T$

Generative Model: Assume an underlying data generation process

$$p(z_{1:T}; \theta)$$

Discriminative Model: Model directly the conditional distribution of future given past

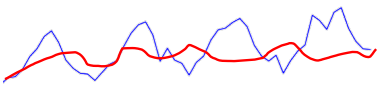
$$p(z_{K+1:T} | z_{1:K}; \theta)$$

Generative vs Discriminative Models for Forecasting

Given a time series: $z_1, z_2, \ldots, z_T$

Generative Model: Assume an underlying data generation process

$$p(z_{1:T}; \theta)$$



Training sequence

Training: find θ that reconstructs observations

Discriminative Model: Model directly the conditional distribution of future given past

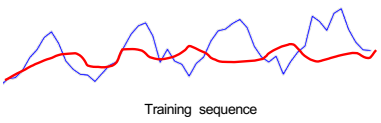
$$p(z_{K+1:T} | z_{1:K}; \theta)$$

Generative vs Discriminative Models for Forecasting

Given a time series: $z_1, z_2, \ldots, z_T$

Generative Model: Assume an underlying data generation process

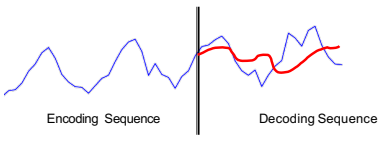
$$p(z_{1:T}; \theta)$$



Training: find θ that reconstructs observations

Discriminative Model: Model directly the conditional distribution of future given past

$$p(z_{K+1:T} | z_{1:K}; \theta)$$



Training: find θ such that prediction matches future given past

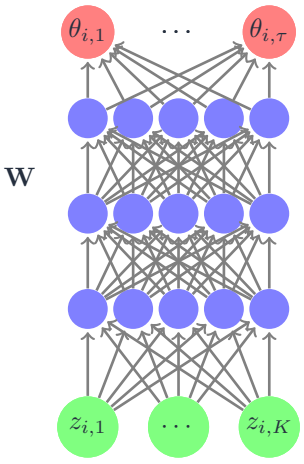
In this workshop...

- A simple feed forward neural network model (discriminative)
- DeepAR: Deep Autoregressive Neural network model (generative) [Flunkert et al., 2017]

Topics we will not really cover:

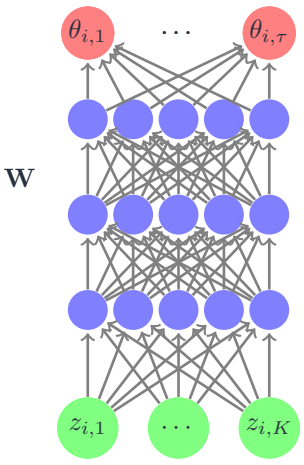
- NBEATS [Oreshkin et al., 2019] (only briefly)
- Transformers [Vaswani et al., 2017] and applications in forecasting [Li et al., 2019; Lim et al., 2019]
- Spline Quantile Forecaster [Gasthaus et al., 2019a]
- Multi-variate models (but there's interesting work [Salinas et al., 2019])
- More Deep Probabilistic models
  - Deep State Space Models for Time Series Forecasting [Rangapuram et al., 2018]
  - Deep Renewal Process for Intermittent Demand Forecasting [Turkmen et al., 2019]
  - Deep Factor Models for Forecasting [Wang et al., 2019]

Simple Feed-Forward Networks for Forecasting



- Discriminative model: given past target of length K, predict future distribution for τ time steps

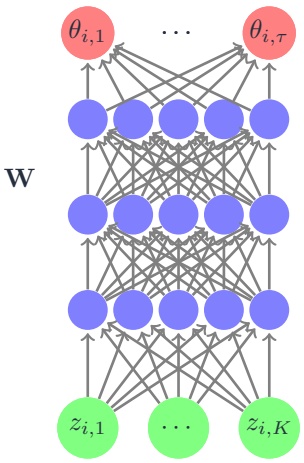
Simple Feed-Forward Networks for Forecasting



- Discriminative model: given past target of length K, predict future distribution for τ time steps
- Output: parameters of the probabilistic model $\theta_{i,t}$
  In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$

Simple Feed-Forward Networks for Forecasting

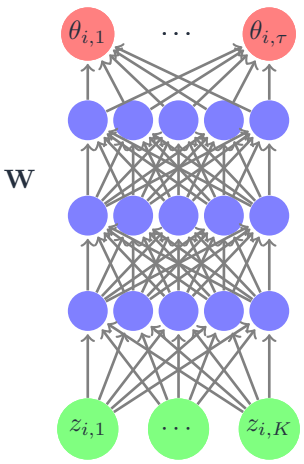
Loss: $-\sum_{t=1}^{\tau} \log p(z_{i,K+t}; \theta_{i,t})$



- Discriminative model: given past target of length K, predict future distribution for τ time steps
- Output: parameters of the probabilistic model $\theta_{i,t}$
  In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$
- Modelling assumption: Joint distribution over future is independent over time

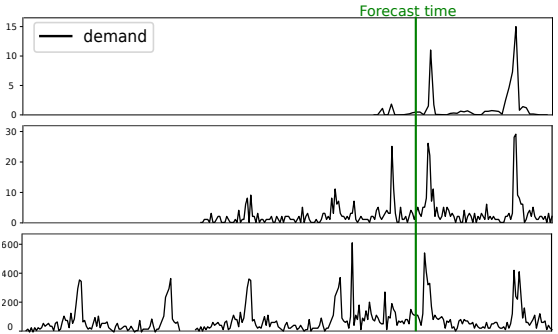
Simple Feed-Forward Networks for Forecasting

Loss: $-\sum_{t=1}^{\tau} \log p(z_{i,K+t}; \theta_{i,t})$

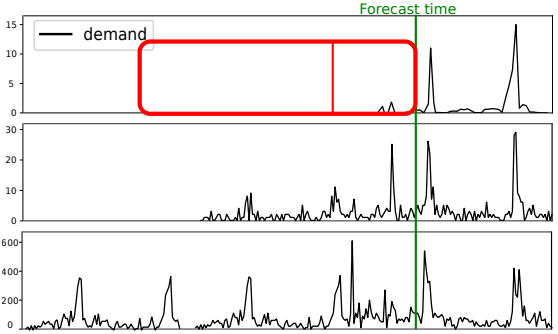


- Discriminative model: given past target of length K, predict future distribution for τ time steps
- Output: parameters of the probabilistic model $\theta_{i,t}$
  In case of Gaussian, $\theta_{i,t} = \{\mu_{i,t}, \sigma_{i,t}\}$
- Modelling assumption: Joint distribution over future is independent over time
- Global training: Same network with weights $\mathbf{W}$ is used for all time series in the dataset

Training: Augmented Data

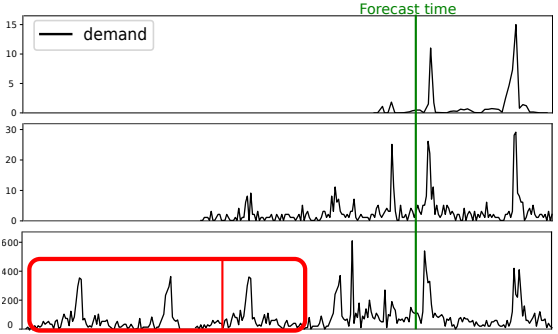


Training: Augmented Data



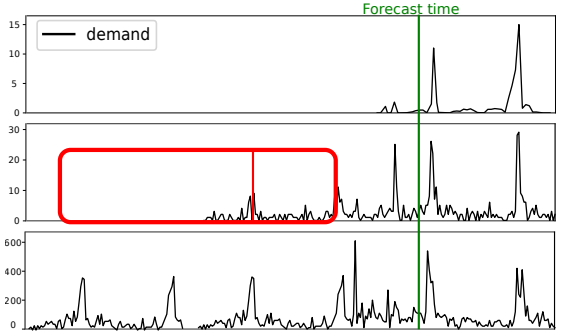
Training: Augmented Data

Training windows: Train on slicing windows across items and time steps
Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target
Window size: larger than prediction length (typically at least twice the forecast horizon)



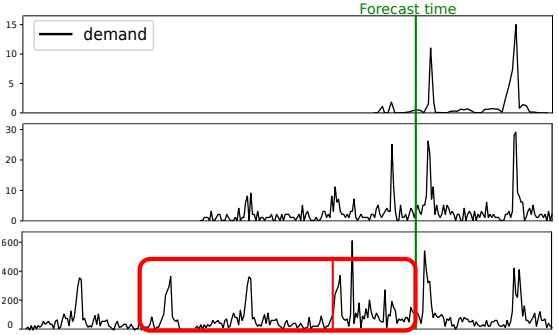
Training: Augmented Data

Training windows: Train on slicing windows across items and time steps
Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target
Window size: larger than prediction length (typically at least twice the forecast horizon)



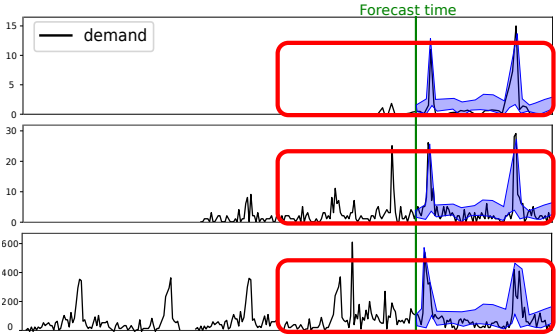
Training: Augmented Data

Training windows: Train on slicing windows across items and time steps
Context versus Prediction: Context target is fed as input to the network; loss is computed on the prediction target
Window size: larger than prediction length (typically at least twice the forecast horizon)



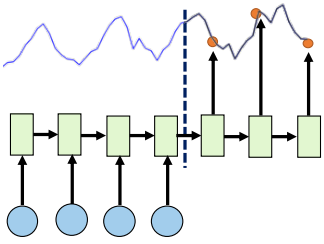
Prediction: Augmented Data

Prediction: Align the window so that start of prediction range coincides with forecast start time

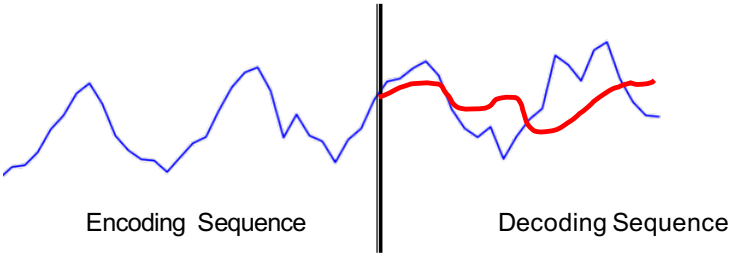


Discriminative Model

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

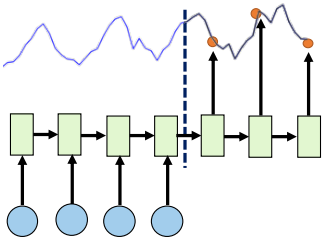


Seq2Seq (Many-to-Many)

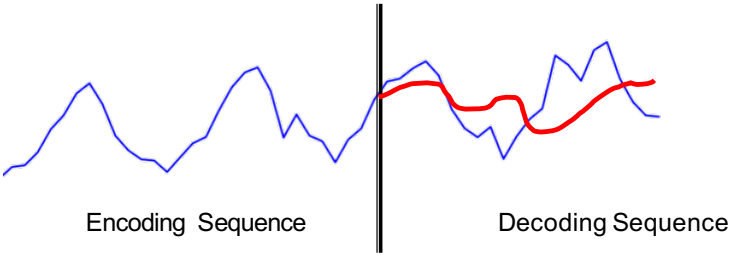


Discriminative Model

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



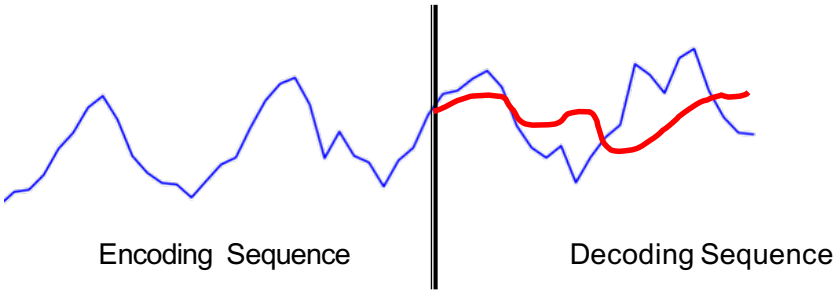
Seq2Seq (Many-to-Many)



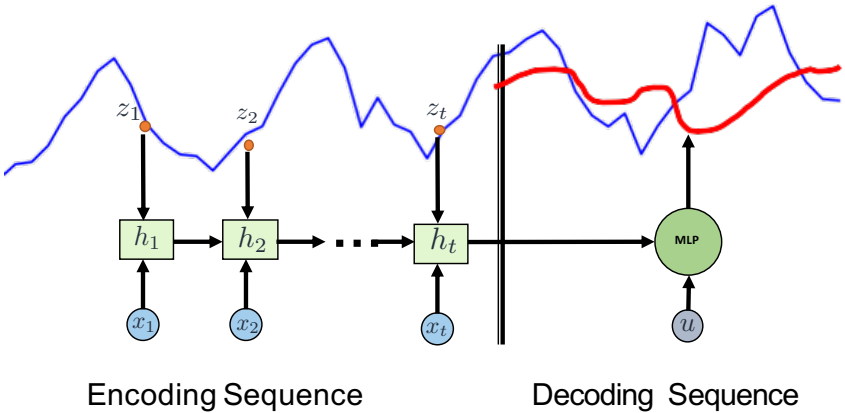
- How well does the *prediction* reconstruct the *decoding sequence* conditioned on the *encoding sequence*?
- Conceptually close to *multivariate regression*
- basic idea: encoder network summarizes encoding sequence in a high-dimensional vector; decoder network takes the high-dimensional vector as input and decodes it into the forecast

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

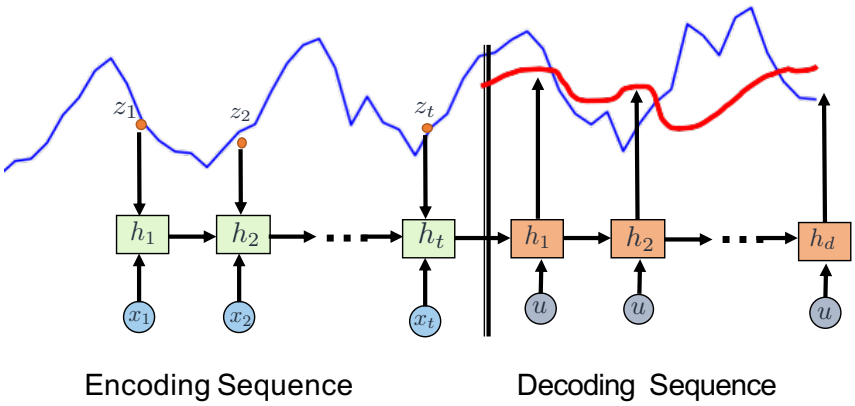
$$f_{encoder} : \{z_1, \cdots, z_{T_e}\} \mapsto \boldsymbol{h}_{T_e}$$
$$f_{decoder} : \boldsymbol{h}_{T_e} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



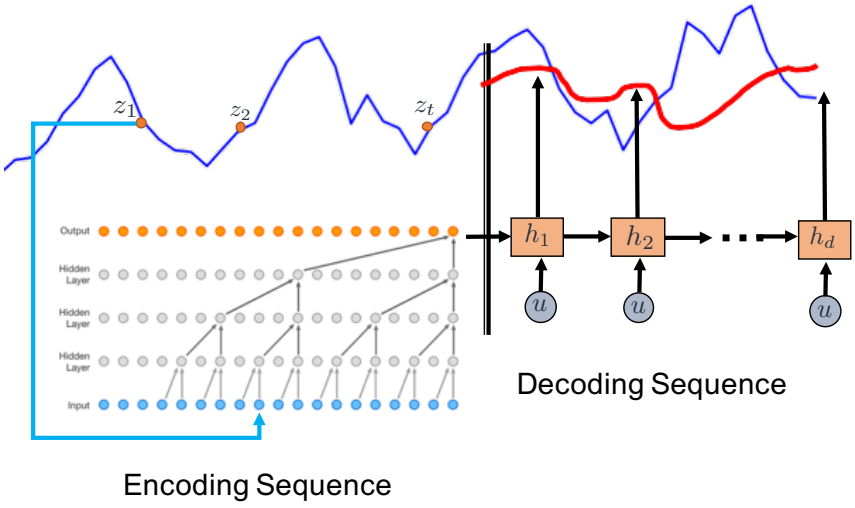
Seq2Seq: RNN-MLP [Wen et al., 2017]



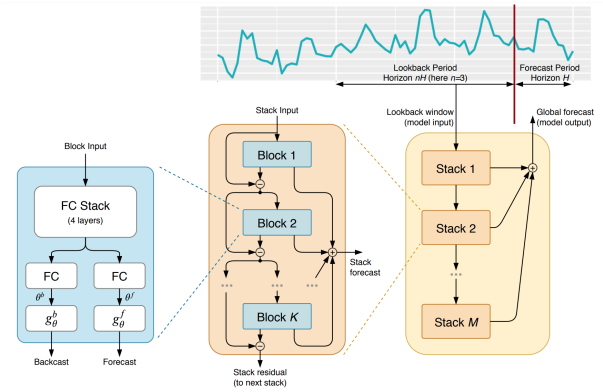
Seq2Seq: RNN-RNN



Seq2Seq: Causal CNN-RNN



N-BEATS: [Oreshkin et al., 2019]



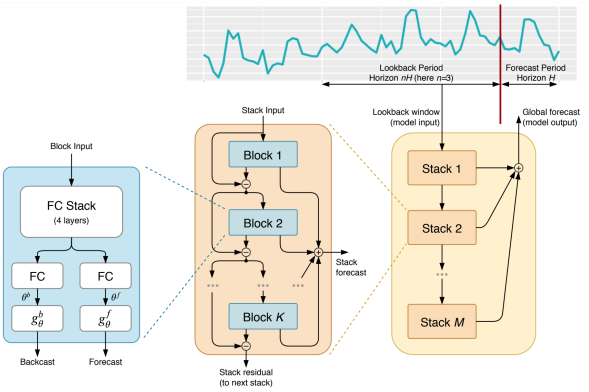
key idea (i) : decompose time series into basis functions

key idea (ii) : residual stacking

key idea (iii) : forecast & backcast at the same time

Generic neural network model (no time series specific ingredients), achieves SOTA on M4.

N-BEATS: [Oreshkin et al., 2019]



Generic neural network model (no time series specific ingredients), achieves SOTA on M4.

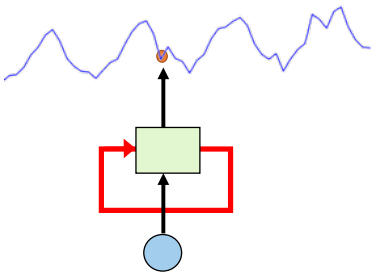
key idea (i) : decompose time series into basis functions

key idea (ii) : residual stacking

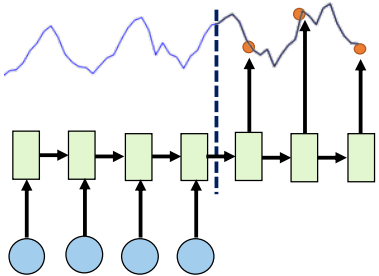
key idea (iii) : forecast & backcast at the same time

Also some wild ensembling with 180 models with clever tricks (using multiple loss functions).

Generative vs Discriminative (cont'd)



Canonical (One-to-One)

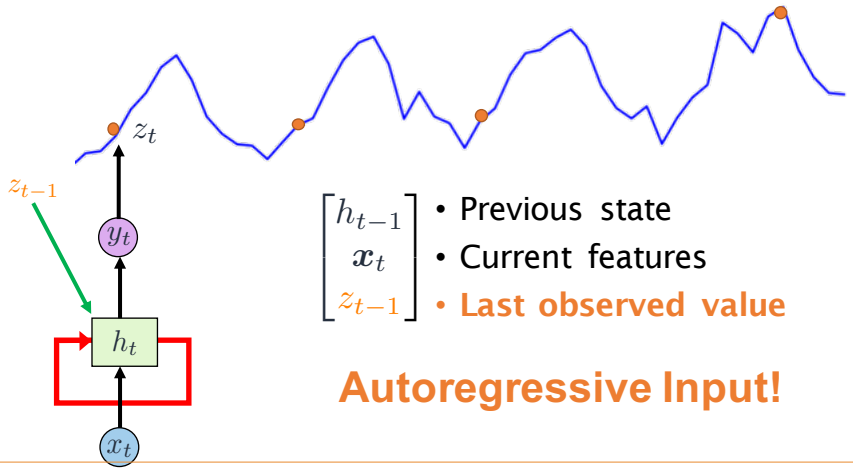


Seq2Seq (Many-to-Many)

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

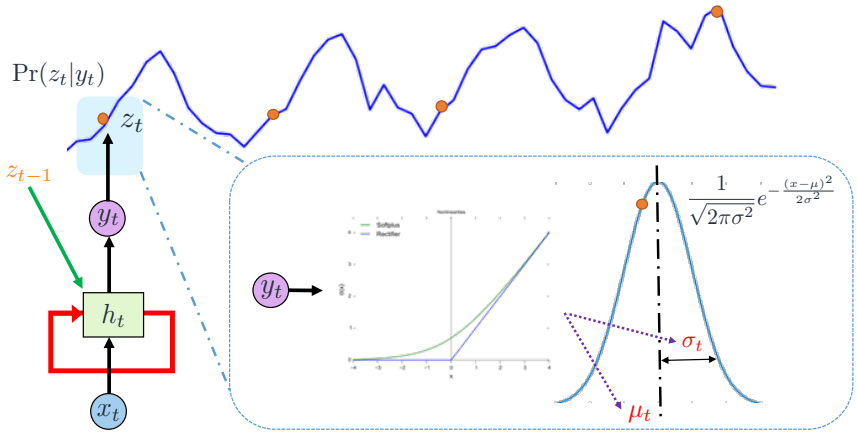
Deep Autoregressive Recurrent Neural Network [Flunkert et al., 2017]

DeepAR in a nutshell: A generative model for the time series



Deep Autoregressive Recurrent Neural Network [Flunkert et al., 2017]

DeepAR in a nutshell: A generative model for the time series

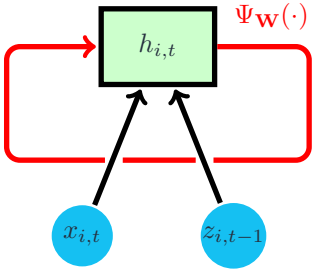


DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \ldots, z_{i,T_i}\}_{i=1}^{n}$

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \ldots, z_{i,T_i}\}_{i=1}^{n}$



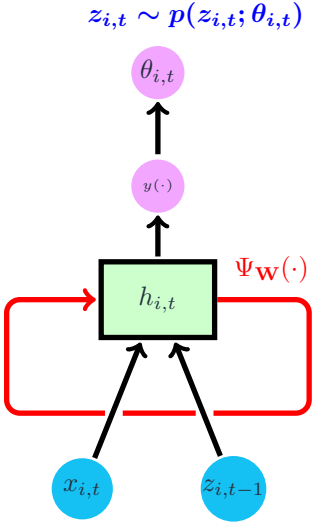
- Transition dynamics:

$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights $\mathbf{W}$ are independent of items and time i, t.

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \ldots, z_{i,T_i}\}_{i=1}^{n}$



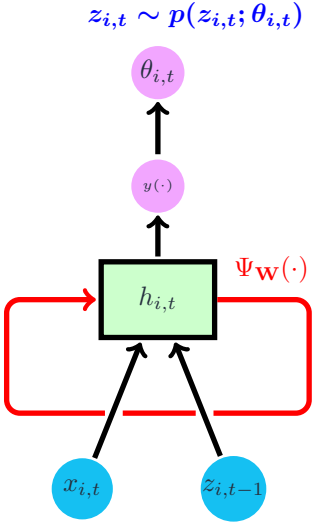
- Transition dynamics:

$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights $\mathbf{W}$ are independent of items and time i, t.

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \ldots, z_{i,T_i}\}_{i=1}^{n}$



- Observation model:

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t}) \quad t = 1, 2, \ldots, T_i$$

$$\theta_{i,t} = y(h_{i,t})$$

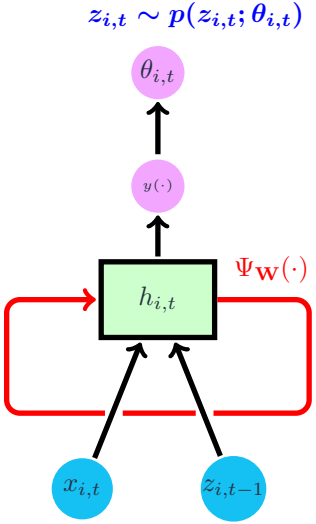
- Transition dynamics:

$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

RNN weights $\mathbf{W}$ are independent of items and time i, t.

DeepAR: In Detail...,

Generative model for the whole data: $D = \{z_{i,1}, z_{i,2}, \ldots, z_{i,T_i}\}_{i=1}^{n}$



- Observation model:

$$z_{i,t} \sim p(z_{i,t}; \theta_{i,t}) \quad t = 1, 2, \ldots, T_i$$

$$\theta_{i,t} = y(h_{i,t})$$

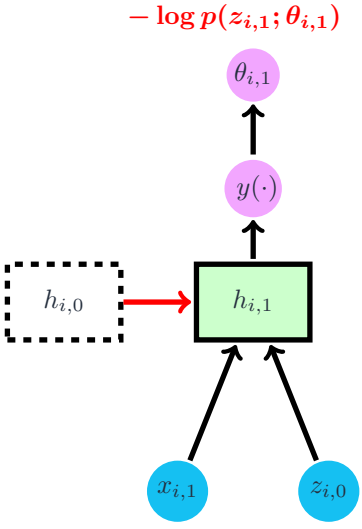
- Transition dynamics:

$$h_{i,t} = \Psi_{\mathbf{W}}(x_{it}, z_{i,t-1}, h_{t-1})$$

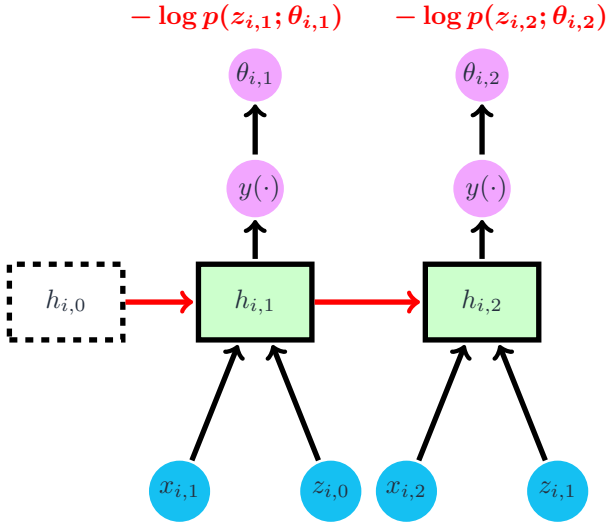
RNN weights $\mathbf{W}$ are independent of items and time i, t.

- More (seasonal) lags $z_{i,\bar{t}}$, $\bar{t} \in L_t$ as features at each time step t

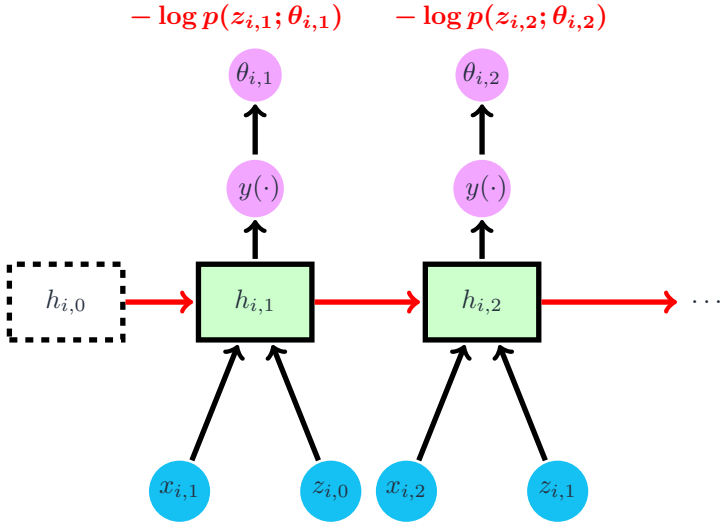
DeepAR: Training



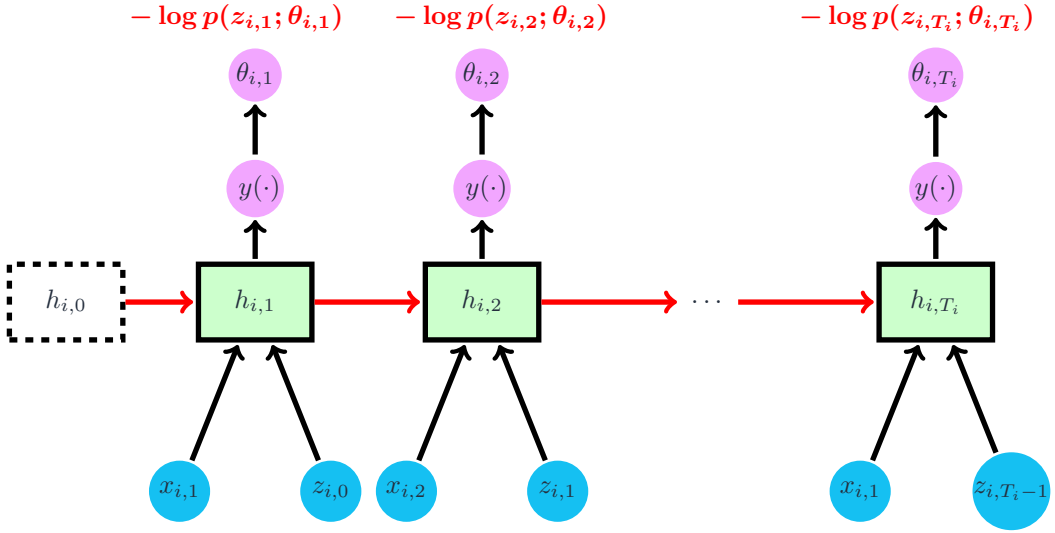
DeepAR: Training



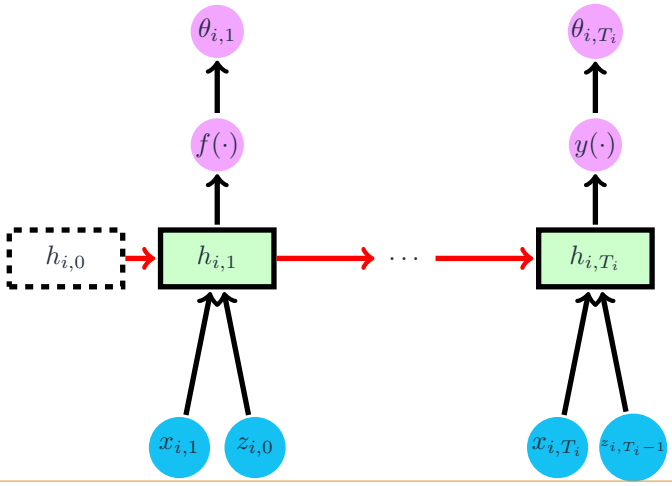
DeepAR: Training



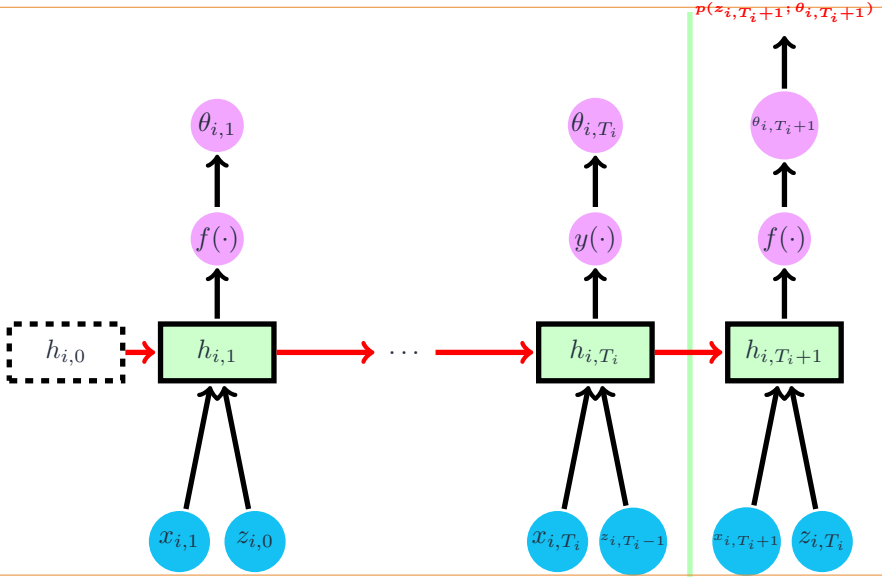
DeepAR: Training



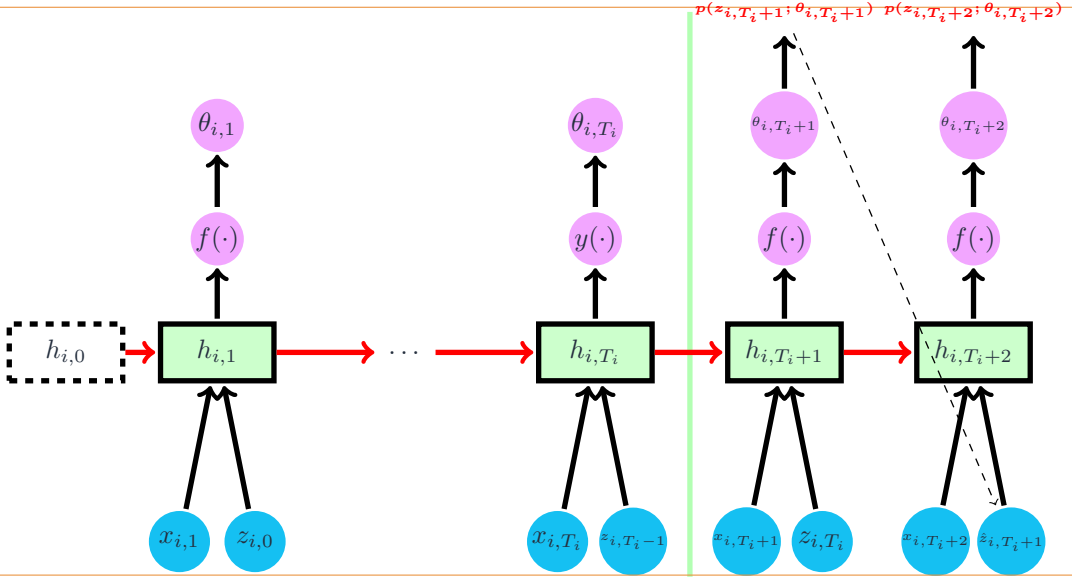
DeepAR: Prediction



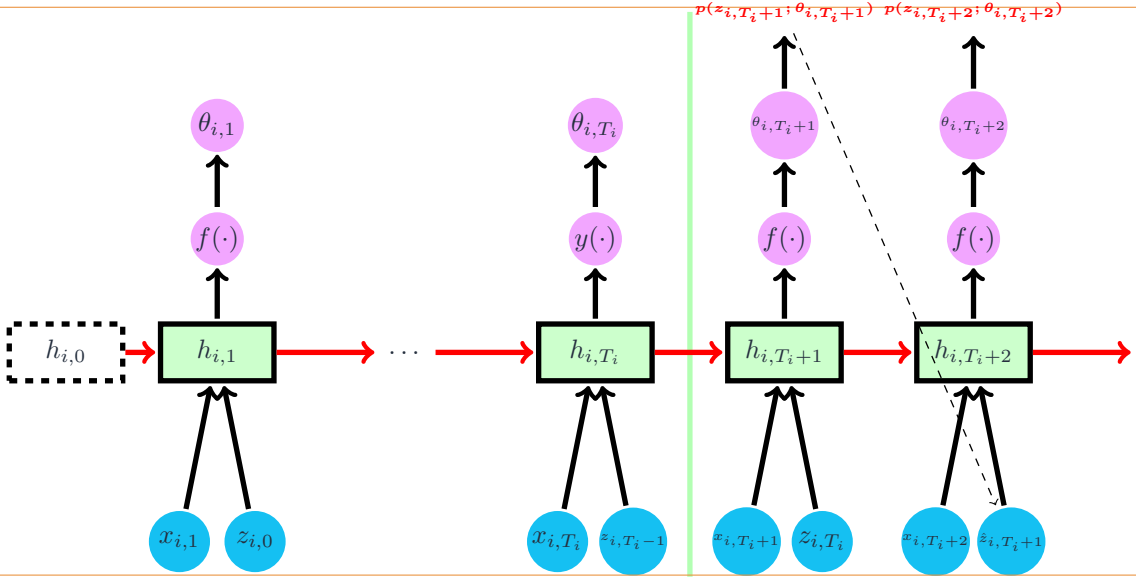
DeepAR: Prediction



DeepAR: Prediction

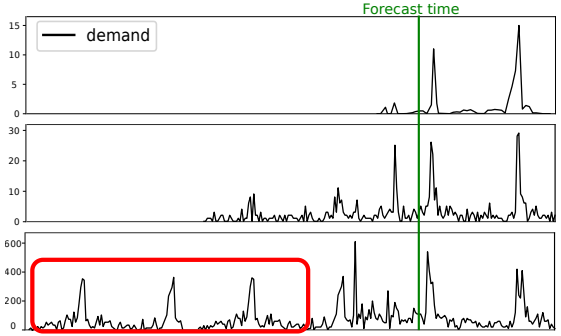


DeepAR: Prediction



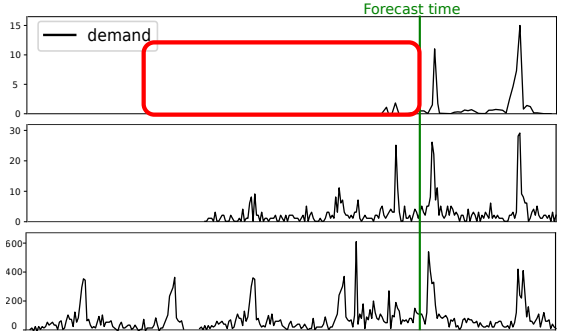
DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)
Window size: larger than prediction length (typically at least twice the forecast horizon)



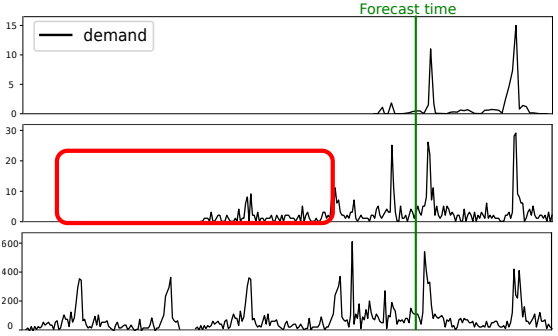
DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)
Window size: larger than prediction length (typically at least twice the forecast horizon)



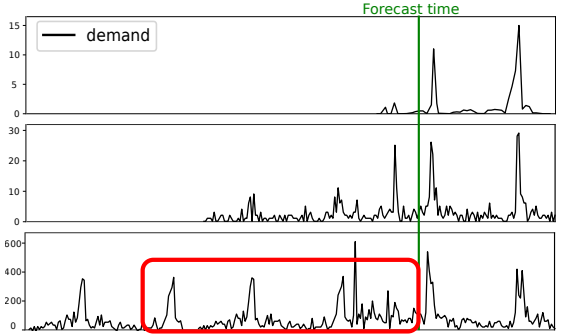
DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)
Window size: larger than prediction length (typically at least twice the forecast horizon)



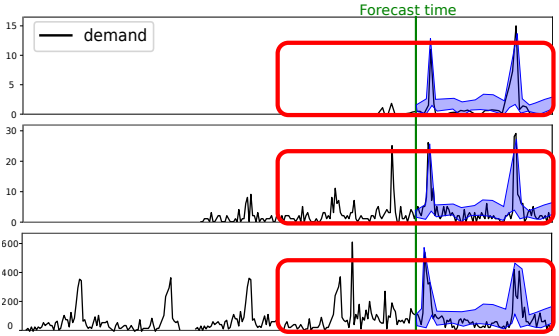
DeepAR: Augmented Training

Training windows: Train on slicing windows across items and time steps (RNN is unrolled only on the window)
Window size: larger than prediction length (typically at least twice the forecast horizon)



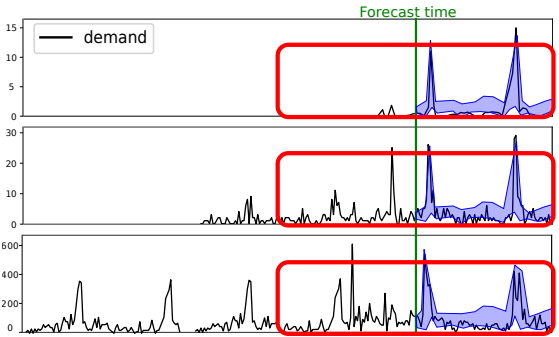
DeepAR: Prediction

Prediction: Align the window to the end of the forecast horizon
Unroll the RNN before the forecast time usually and start making predictions from the forecast start date



DeepAR: Prediction

Prediction: Align the window to the end of the forecast horizon
Unroll the RNN before the forecast time usually and start making predictions from the forecast start date



(RNN always unrolled for same number of time steps both during training and prediction!)

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$

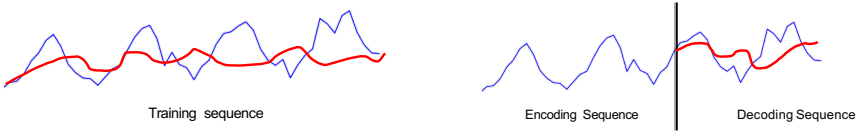
- What choices do I have for modelling observations?

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$

- What choices do I have for modelling observations?
  1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past

$$\underbrace{p(z_{1:T}; \theta)}_{\text{generative}} \quad \text{vs} \quad \underbrace{p(z_{T+1:T+\tau} | z_{1:T}; \theta)}_{\text{discriminative}}$$

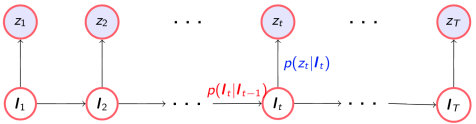


Goal: reconstructing observations vs predicting future given past

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$

- What choices do I have for modelling observations?
  1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
  2. **How do you model the time series?** Independent, Autoregressive process or State space model



Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$

- What choices do I have for modelling observations?
  1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
  2. **How do you model the time series?** Independent, Autoregressive process or State space model
- Modelling choices for the feature map/architecture:

Classification of Forecasting Approaches in Deep Learning

$$z_1, z_2, \ldots, z_T \implies P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$

- What choices do I have for modelling observations?
  1. **What do you model?** Joint distribution of past and future vs conditional distribution of future given past
  2. **How do you model the time series?** Independent, Autoregressive process or State space model
- Modelling choices for the feature map/architecture:
  1. MLP
  2. Recurrent neural network
  3. Convolutional neural network/Wavenet
  4. Transformers

How do I choose a neural forecasting model?

- Assumption: you've understood your data and your problem
- start with a mature baseline model: DeepAR, NBEATS, MQCNN
- compare with a classical approach (R Forecast, Prophet)
- understand where improvement could come from
- if there's structure we can take advantage of: choose an advanced model or get cracking yourself

Benchmark Open-Source Dataset Results

- GluonTS github repository: https://github.com/awslabs/gluon-ts
- Evaluation criteria:

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

  More evaluation metrics available from the open source code! as well as in [Alexandrov and other, 2019]
- DeepAR-Spl Gasthaus et al. [2019b] DeepAR that learns to predict the quantile function directly at each time step; the quantile function is parametrized by piecewise linear splines.
- data sets from Monash Time Series Repository also available in GluonTS.

estimator dataset	Auto-ARIMA	Auto-ETS	Prophet	NPTS	Transformer	CNN-QR	DeepAR	DeepAR-Spl	GP
SP500-returns	0.975±0.000	0.982±0.001	0.985±0.001	**0.832±0.000**	0.836±0.001	0.906±0.006	0.838±0.003	0.836±0.005	0.858±0.000
electricity	0.056±0.000	0.067±0.000	0.094±0.000	**0.055±0.000**	0.062±0.001	0.081±0.003	0.065±0.007	0.065±0.009	0.111±0.001
m4-Daily	0.024±0.000	**0.023±0.000**	0.090±0.000	0.145±0.000	0.028±0.000	0.026±0.001	0.028±0.000	0.025±0.002	0.074±0.000
m4-Hourly	0.040±0.001	0.044±0.000	0.043±0.000	0.048±0.000	0.042±0.010	0.064±0.006	**0.035±0.006**	0.124±0.038	0.132±0.000
m4-Monthly	**0.097±0.000**	0.099±0.000	0.132±0.000	0.233±0.000	0.134±0.002	0.127±0.002	0.136±0.002	0.106±0.001	0.242±0.000
m4-Quarterly	0.080±0.000	**0.078±0.000**	0.123±0.000	0.255±0.000	0.095±0.003	0.091±0.000	0.091±0.001	0.081±0.002	0.335±0.000
m4-Weekly	0.050±0.000	0.051±0.000	0.108±0.000	0.296±0.001	0.075±0.005	0.056±0.000	0.072±0.001	**0.043±0.001**	0.137±0.000
m4-Yearly	0.124±0.000	0.126±0.000	0.156±0.000	0.355±0.000	0.127±0.004	0.121±0.000	0.121±0.001	**0.111±0.002**	0.455±0.000
parts	1.401±0.002	1.342±0.002	1.637±0.002	1.356±0.002	1.000±0.003	**0.901±0.000**	0.970±0.005	0.943±0.025	1.591±0.000
traffic	-	0.462±0.000	0.273±0.000	0.162±0.000	0.132±0.010	0.186±0.002	0.127±0.004	**0.087±0.001**	0.152±0.000
wiki10k	0.610±0.001	0.841±0.001	0.681±0.000	0.452±0.000	0.294±0.008	0.314±0.002	0.295±0.028	**0.273±0.007**	0.452±0.000

Mean and std CRPS error over 10 runs

Q & A and/or Coffee Break (back at 7:30pm)

GluonTS github repository: https://github.com/awslabs/gluon-ts

Multivariate

High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes
Salinas, Bohlke-Schneider, Callot, G.
NeurIPS 2019

Normalizing Kalman Filters for Multivariate Time Series Analysis
de Bézenac, Rangapuram, Benidis, Bohlke-Schneider, Kurle, Stella, Hasson, Gallinari, J.
NeurIPS 2020

Hierarchical

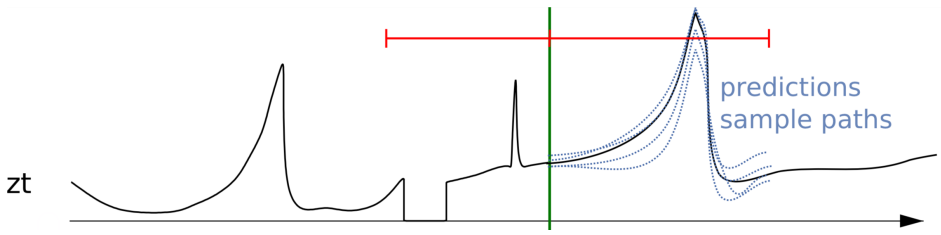
End-to-End Learning of Coherent Probabilistic Forecasts for Hierarchical Time Series
Rangapuram, Werner, Benidis, Mercado, G., J.
ICML 2021

Multivariate Forecasting

Recall: Classical Setup: single time series, exogenous variables.

Predict the future behavior of a time series given its past

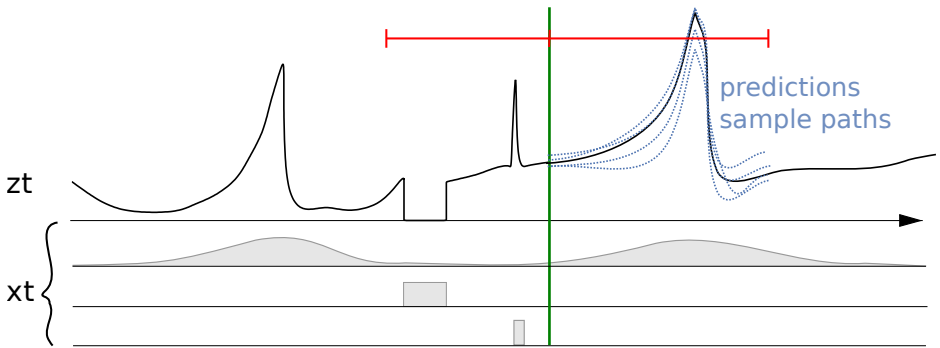
$$z_1, z_2, \ldots, z_T \Longrightarrow P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$



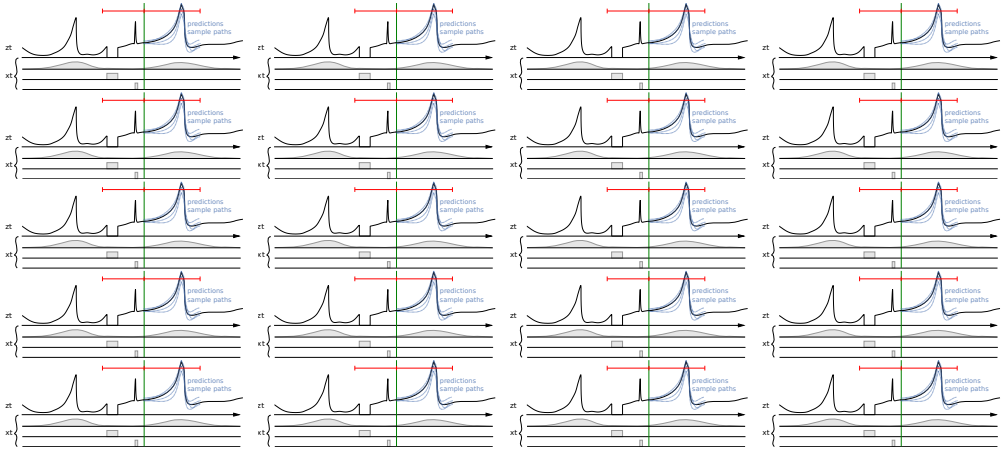
Recall: Classical Setup: single time series, exogenous variables.

Predict the future behavior of a time series given its past

$$z_1, z_2, \ldots, z_T \Longrightarrow P(z_{T+1}, z_{T+2}, \ldots z_{T+\tau})$$



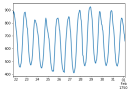
Typical Setup: panel of *related* time series, exogenous variables.

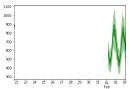


We have a panel of time series $Z = \{\mathbf{z_i}\}_{i \in I}$ and we want to forecast each time series in the panel. Typically $|I|$ is large.

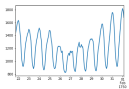
Recall: Local, Univariate Model

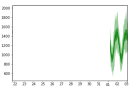
$$f\left(\quad\right) = \quad$$



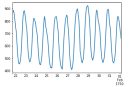


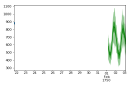
$$f\left(\quad\right) = \quad$$



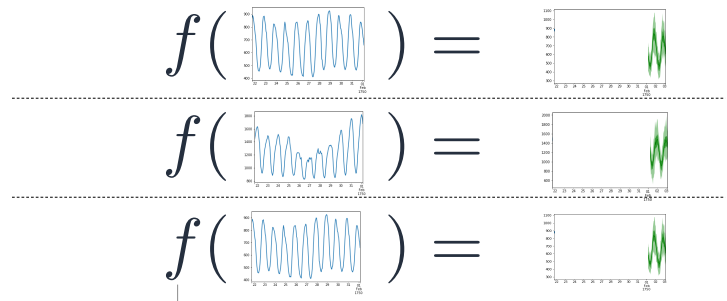


$$f\left(\quad\right) = \quad$$





Recall: Local, Univariate Model

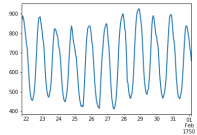


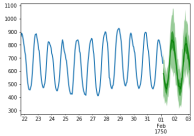
$$f(\quad) = g_{\phi^*}(\quad)$$
$$\phi^* = \arg\min_{\phi} L(\quad, \quad)$$

Examples: ARIMA(X), ETS, Prophet

Recall: Global, Univariate model

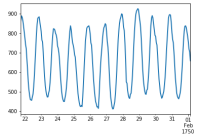
$$f_\theta\left(\quad\right) =$$

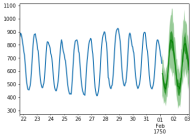


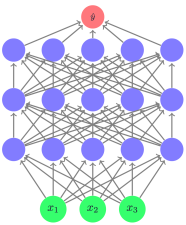


Recall: Global, Univariate model

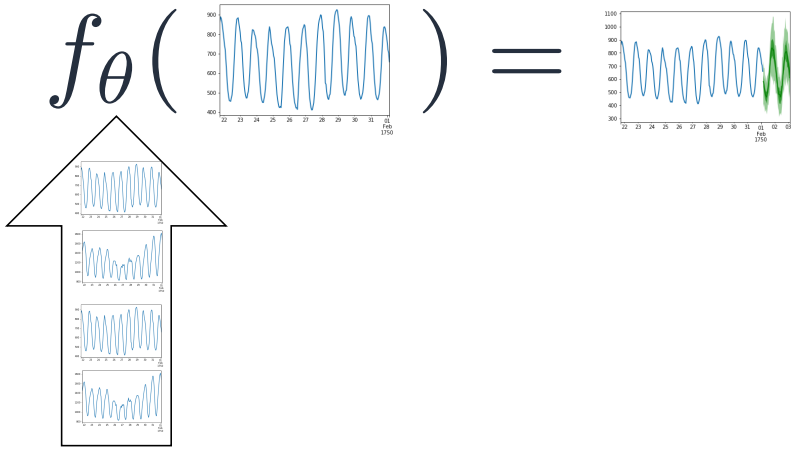
$$f_\theta\left(\quad\right) = $$







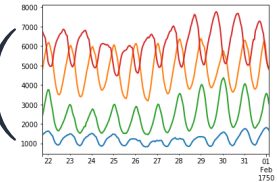
Recall: Global, Univariate model

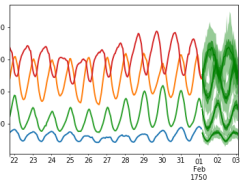


Example: Dynamic Factor models, DeepAR

New: Local Multivariate Model

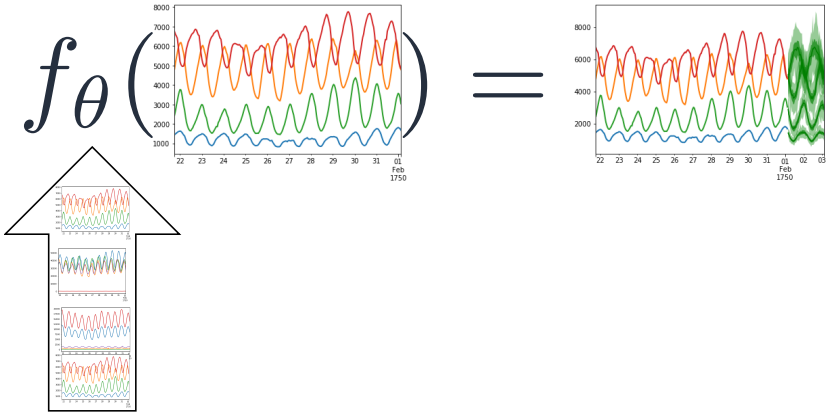
$$f\left(\right) =$$





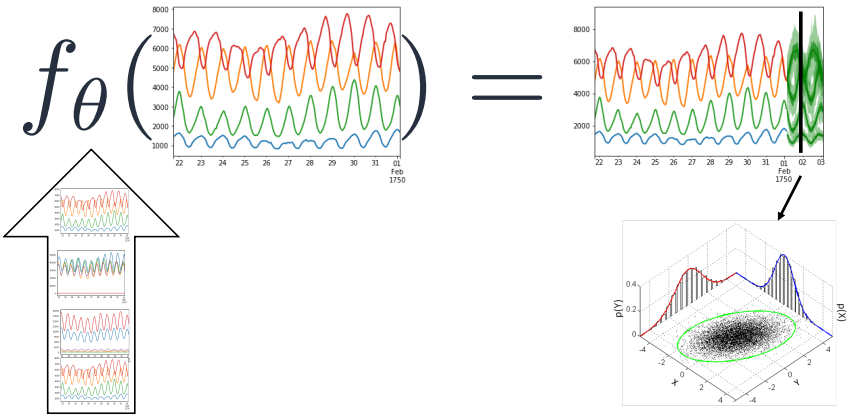
Examples: VARMA, VARIMAX, Multivariate State-Space Models

New: Global Multivariate Model



Examples: DeepVAR

Global Multivariate **Joint** Model



Examples: This talk (DeepAR-GP-Copula; Normalizing flows).

Remarks

Missing Details

- brushed over treatment of co-variates

Remarks

Missing Details

- brushed over treatment of co-variates

Why do we need all this?

- This is the way. Principled.
- Allows to condition on future values (but practically still hard!)
- Ideas from the audience?

General Challenges with Probabilistic Joint Multivariate Forecasting

1. Different scales of time series
2. Marginally non-Gaussian data
3. Evaluating the likelihood of multivariate Gaussian is computationally expensive
4. How to capture non-Gaussian dependency structures?
5. All time series at training time need to be available during inference

General Challenges with Probabilistic Joint Multivariate Forecasting

1. Different scales of time series
2. Marginally non-Gaussian data
3. Evaluating the likelihood of multivariate Gaussian is computationally expensive
4. How to capture non-Gaussian dependency structures?
5. All time series at training time need to be available during inference

In this talk:

1. Turn DeepAR (an autoregressive recurrent neural network model) into a multivariate model via
   - Low-Rank Gaussian Copula Processes
   - Normalizing Flows

General Challenges with Probabilistic Joint Multivariate Forecasting

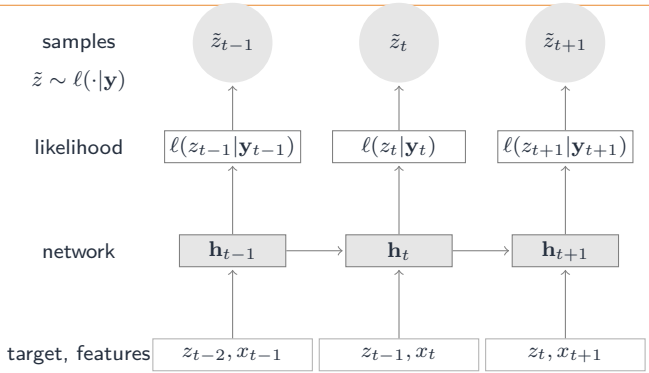
1. Different scales of time series
2. Marginally non-Gaussian data
3. Evaluating the likelihood of multivariate Gaussian is computationally expensive
4. How to capture non-Gaussian dependency structures?
5. All time series at training time need to be available during inference

In this talk:

1. Turn DeepAR (an autoregressive recurrent neural network model) into a multivariate model via
   - Low-Rank Gaussian Copula Processes
   - Normalizing Flows
2. Use multivariate models for hierarchical forecasting

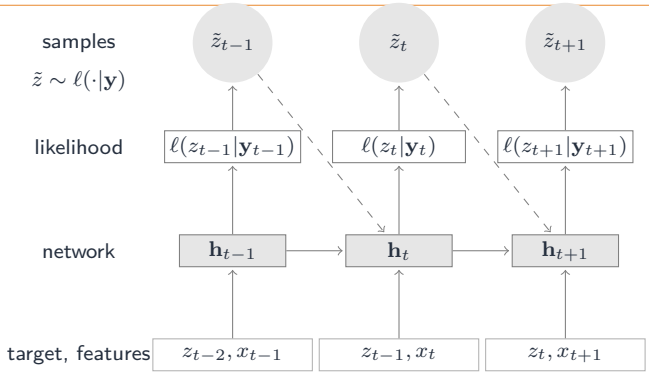
DeepAR

Recall: DeepAR - training



- z_t target, x_t features, h_t stack of several LSTM
- $\ell(z_t|\mathbf{y}_t)$ likelihood: Gaussian, negative binomial
- loss $= \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log \ell(z_{i,t}|y(\mathbf{h}_{i,t}))$

Recall: DeepAR - prediction



- z_t target, x_t features, h_t stack of several LSTM
- $\ell(z_t|\mathbf{y}_t)$ likelihood: Gaussian, negative binomial
- loss $= \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log \ell(z_{i,t}|y(\mathbf{h}_{i,t}))$
- prediction: put sample $\tilde{z} \sim \ell(\cdot|\mathbf{y})$ instead of true target for unknown (future) values
- estimate free parameters θ over entire panel of time series (global model)

Multivariate DeepAR

DeepAR as a Multivariate Model

- The transition dynamics are parametrized using a LSTM-RNN.

$$\mathbf{h}_{i,t} = \varphi_{\theta_h}(\mathbf{h}_{i,t-1}, z_{i,t-1}).$$

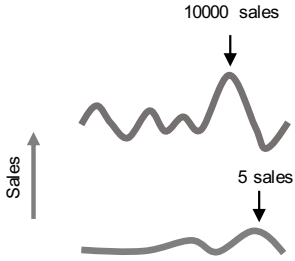
- We can factorize **the joint distribution of the observations** as:

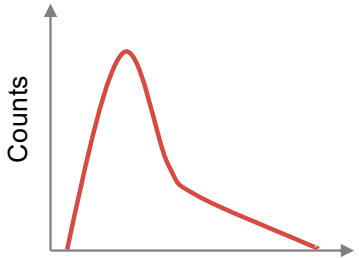
$$p(\mathbf{z}_1, \dots \mathbf{z}_{T+\tau}) = \prod_{t=1}^{T+\tau} p(\mathbf{z}_t | \mathbf{z}_1, \dots, \mathbf{z}_{t-1}) = \prod_{t=1}^{T+\tau} p(\mathbf{z}_t | \mathbf{h}_t).$$

- We train by minimizing the negative, Multivariate Gaussian **log-likelihood**:

$$-\log p(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_T) = -\sum_{t=1}^{T} \log p(\mathbf{z}_t | \mathbf{h}_t).$$

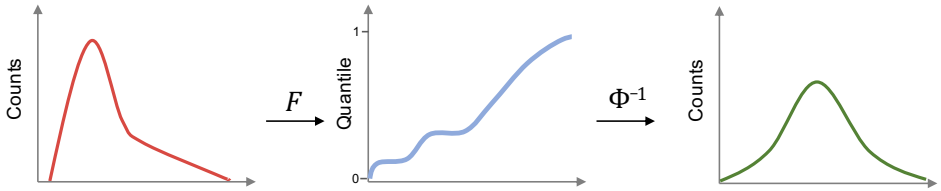
Issue 1: Scaling and non-Gaussian Data





Addressing Issue 1: Gaussian Copula

$$C(F_1(z_1), \ldots, F_d(z_N)) = \phi_{\boldsymbol{\mu},\Sigma}(\Phi^{-1}(F_1(z_1)), \ldots, \Phi^{-1}(F_N(z_N)))$$



Addressing Issue 2: Computational Cost

$$\Sigma(\mathbf{h}_t) = \begin{bmatrix} cov(x_1, x_1) & & cov(x_1, x_N) \\ & \ddots & \\ cov(x_N, x_1) & & cov(x_N, x_N)) \end{bmatrix}$$

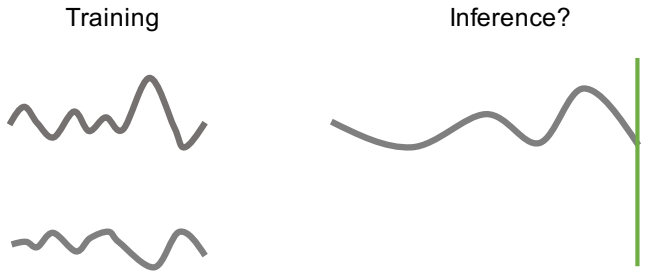
- The covariance matrix $\Sigma(\mathbf{h}_t)$ is a SPD matrix with $O(N^2)$ parameters.
- Naïve likelihood evaluation requires $O(N^3)$ operations, low-rank only $O(Nr^2 + r^3)$.

Addressing Issue 2: Low-rank Approximation

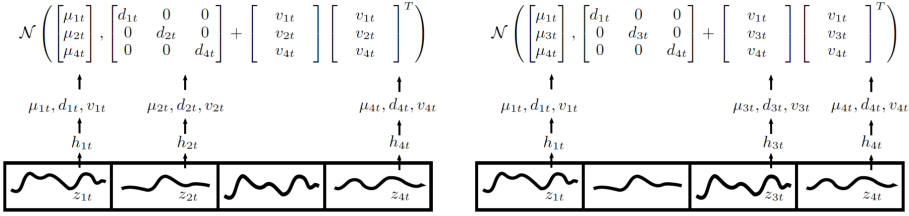
$$\Sigma(\mathbf{h}_t) = \begin{bmatrix} d_1(\mathbf{h}_{1,t}) & & 0 \\ & \ddots & \\ 0 & & d_N(\mathbf{h}_{N,t}) \end{bmatrix} + \begin{bmatrix} \mathbf{v}_1(\mathbf{h}_{1,t}) \\ \dots \\ \mathbf{v}_N(\mathbf{h}_{N,t}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_1(\mathbf{h}_{1,t}) \\ \dots \\ \mathbf{v}_N(\mathbf{h}_{N,t}) \end{bmatrix}^T = D_t + V_t V_t^T.$$

- $D_t \in \mathbb{R}^{N \times N}$ is diagonal. $V_t \in \mathbb{R}^{N \times r}$. $O(N \times r)$ parameters.
- $V_t V_t^T$ is a low-rank matrix with **rank hyperparameter** $r \ll N$.
- Low-rank likelihood evaluation only $O(Nr^2 + r^3)$.

Issue 3: Not All Time Series Available During Inference



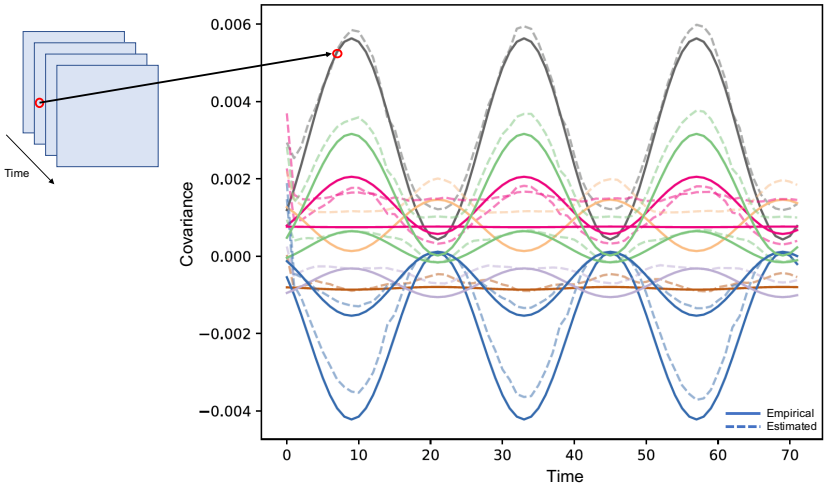
Addressing Issue 3: Gaussian Process Parametrization



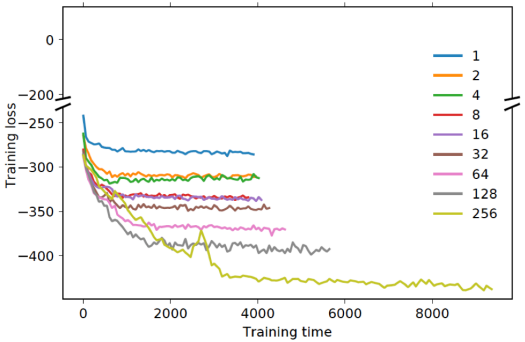
$$\mu_i(\mathbf{h}_{i,t}) = \tilde{\mu}(\mathbf{y}_{i,t}), \qquad d_i(\mathbf{h}_{i,t}) = \tilde{d}(\mathbf{y}_{i,t}), \qquad \mathbf{v}_i(\mathbf{h}_{i,t}) = \tilde{\mathbf{v}}(\mathbf{y}_{i,t}).$$

- $\mathbf{y}_{i,t} = [\mathbf{h}_{i,t}; \mathbf{e}_i]^T \in \mathbb{R}^{p \times 1}$, with $\mathbf{e}_i$ known features or learnt embeddings.
- These shared functions can **parametrize a Gaussian Process** $g_t(\mathbf{y}_{i,t})$.
- $g_t \sim \mathsf{GP}(\tilde{\mu}(\cdot), k(\cdot,\cdot))$, with $k(\mathbf{y}, \mathbf{y}') = \mathbb{1}_{\mathbf{y}=\mathbf{y}'}\tilde{d}(\mathbf{y}) + \tilde{\mathbf{v}}(\mathbf{y})^T\tilde{\mathbf{v}}(\mathbf{y}')$.

Model recovers Covariances of Artificial Dataset



Effect of Low-rank Approximation

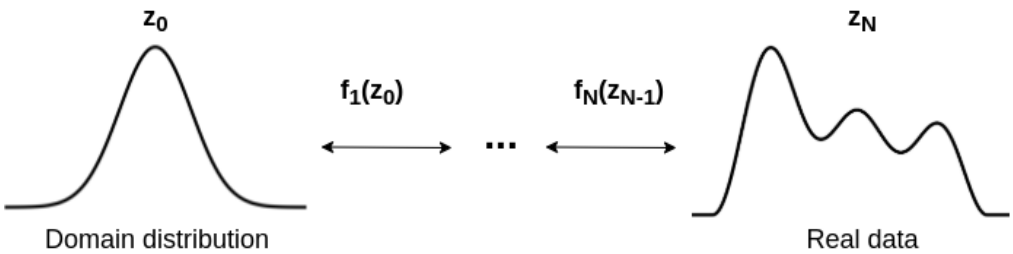


rank	test NLL	train NLL
1	-291.4+/-8.2	-288.9+/-8.2
2	-306.2+/-6.7	-304.8+/-5.7
4	-319.3+/-4.9	-312.1+/-3.5
8	-333.6+/-7.7	-330.2+/-6.3
16	-334.8+/-4.9	-337.5+/-4.
32	**-341.8+/-6.8**	-345.2+/-17.0
64	-338.5+/-10.9	-360.5+/-10.7
128	-326.6+/-20.1	-393.7+/-26.1
256	-238.0+/-38.4	**-423.1+/-20.7**

Alternative: Handling Non-Gaussian Data via Transformation

- Normalizing Flow: Sequence of invertible, differentiable transformations

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

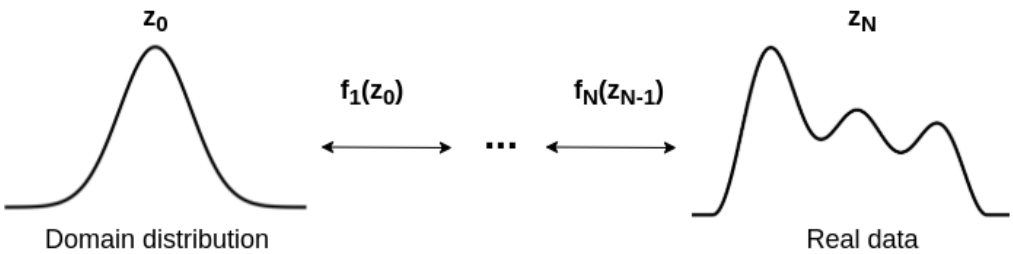


Simple example of a (univariate) normalizing flow: Box-Cox transformation.

Alternative: Handling Non-Gaussian Data via Transformation

- Normalizing Flow: Sequence of invertible, differentiable transformations

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



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

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

Simple example of a (univariate) normalizing flow: Box-Cox transformation.

Normalizing Flows can be used on top of DeepAR or other neural architectures:

- Transformers: Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. Rasul et al. ICLR 2021
- Deep State Space Models: Normalizing Kalman Filters for Multivariate Time Series Analysis. Rangapuram, et al and J. NeurIPS 2020

Some open challenges: count data.

Experiments

Quantitative evaluation:

$$\Lambda_\alpha(q, y) = (\alpha - \mathbf{1}_{\{y<q\}})(y - q),$$

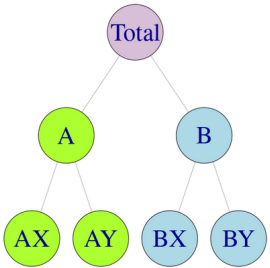
$$\text{CRPS}(F^{-1}, y) = \int_0^1 2\Lambda_\alpha(F^{-1}(\alpha), y)\, d\alpha.$$

	CRPS-Sum				
dataset estimator	exchange	solar	elec	traffic	wiki
VAR	0.010+/-0.000	0.524+/-0.001	0.031+/-0.000	0.144+/-0.000	3.400+/-0.003
GARCH	0.020+/-0.000	0.869+/-0.000	0.278+/-0.000	0.368+/-0.000	-
Vec-LSTM-ind	0.009+/-0.000	0.470+/-0.039	0.731+/-0.007	0.110+/-0.020	0.801+/-0.029
Vec-LSTM-ind-scaling	0.008+/-0.001	0.391+/-0.017	0.025+/-0.001	0.087+/-0.041	0.133+/-0.002
Vec-LSTM-fullrank	0.646+/-0.114	0.956+/-0.000	0.999+/-0.000	-	-
Vec-LSTM-fullrank-scaling	0.394+/-0.174	0.920+/-0.035	0.747+/-0.020	-	-
Vec-LSTM-lowrank-Copula	0.007+/-0.000	0.319+/-0.011	0.064+/-0.008	0.103+/-0.006	0.241+/-0.033
GP	0.011+/-0.001	0.828+/-0.010	0.947+/-0.016	2.198+/-0.774	0.933+/-0.003
GP-scaling	0.009+/-0.000	0.368+/-0.012	0.022+/-0.000	0.079+/-0.000	1.483+/-1.034
GP-Copula	0.007+/-0.000	0.337+/-0.024	0.024+/-0.002	0.078+/-0.002	0.086+/-0.004
DeepAR	0.008+/-0.002	0.389+/-0.005	0.035+/-0.007	**0.058+/-0.004**	0.067+/-0.011
DeepState	**0.005+/-0.000**	0.420+/-0.003	0.028+/-0.000	0.143+/-0.003	0.065+/-0.001
NKF-Local (Ours)	0.006+/-0.000	0.416+/-0.003	0.026+/-0.002	0.068+/-0.000	**0.052+/-0.007**
NKF (Ours)	**0.005+/-0.000**	**0.318+/-0.007**	**0.021+/-0.001**	0.098+/-0.002	0.067+/-0.002

Part II - Hierarchical Forecasting, a multivariate forecasting application

Hierarchical Time Series

Multivariate time series with a hierarchical structure

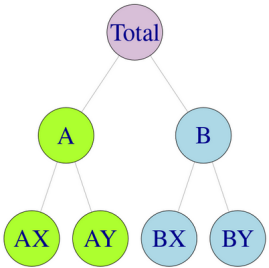


An example hierarchy (Hyndman et al., 2018).

- Daily sales of all laptops
- Daily sales of a particular brand of laptops
- Daily sales of a given brand in a particular location

Hierarchical Time Series

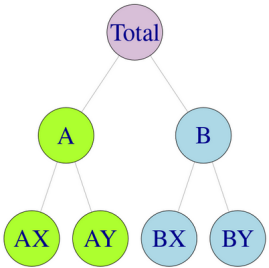
Hierarchical time series satisfies linear aggregation constraints



$$\mathbf{y}_t = S\mathbf{b}_t$$

Hierarchical Time Series

Hierarchical time series satisfies linear aggregation constraints

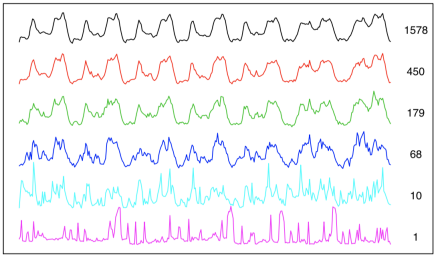


$$\mathbf{y}_t = S\mathbf{b}_t$$

$$\begin{bmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{bmatrix}$$

Problem: Hierarchical Time Series Forecasting

- Aggregation-level vs Accuracy trade-off
- Exploit information at all levels and produce probabilistic forecasts that are consistent (coherent)



One week of electricity demand with different number of aggregated series (Ben Taieb et al., 2017).

Literature: Hierarchical Time Series Forecasting

State-of-the-art (almost all except for (Ben Taieb et al., 2017)) focusses on point forecasts:

1. Learn a univariate model for each time series independently
2. Reconcile the point forecasts in a post-processing step

Literature: Hierarchical Time Series Forecasting

State-of-the-art (almost all except for (Ben Taieb et al., 2017)) focusses on point forecasts:

1. Learn a univariate model for each time series independently
2. Reconcile the point forecasts in a post-processing step

$$\text{Projection:} \quad \hat{\mathbf{y}}_t = \underset{\mathbf{y} \in \mathbb{R}^n}{\operatorname{argmin}} \quad \|\mathbf{y} - \bar{\mathbf{y}}_t\|_P^2$$
$$\text{sb. to:} \quad A\mathbf{y} = 0.$$

Literature: Hierarchical Time Series Forecasting

State-of-the-art (almost all except for (Ben Taieb et al., 2017)) focusses on point forecasts:

1. Learn a univariate model for each time series independently
2. Reconcile the point forecasts in a post-processing step

$$\text{Projection:} \quad \hat{\mathbf{y}}_t = \underset{\mathbf{y} \in \mathbb{R}^n}{\operatorname{argmin}} \quad \|\mathbf{y} - \bar{\mathbf{y}}_t\|_P^2$$
$$\text{sb. to:} \quad A\mathbf{y} = 0.$$

(Ben Taieb et al., 2017) learns a multivariate *probabilistic* model but requires an explicit post-processing step

Literature: Hierarchical Time Series Forecasting

State-of-the-art (almost all except for (Ben Taieb et al., 2017)) focusses on point forecasts:

1. Learn a univariate model for each time series independently
2. Reconcile the point forecasts in a post-processing step

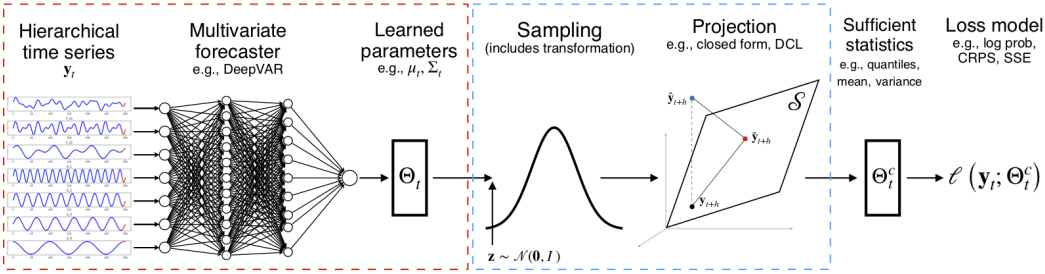
$$\text{Projection:} \quad \hat{\mathbf{y}}_t = \underset{\mathbf{y} \in \mathbb{R}^n}{\operatorname{argmin}} \quad \|\mathbf{y} - \bar{\mathbf{y}}_t\|_P^2$$
$$\text{sb. to:} \quad A\mathbf{y} = 0.$$

(Ben Taieb et al., 2017) learns a multivariate *probabilistic* model but requires an explicit post-processing step

Note: Available as `gluonts` models

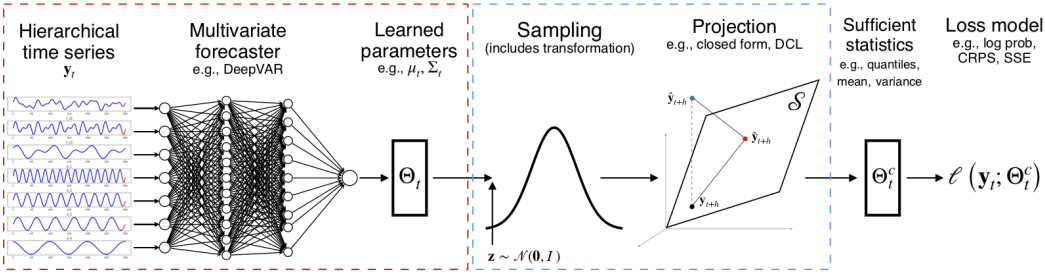
Our Approach: End-to-End Learning for Hierarchical Forecasting

Idea: Combine the two steps to form a single *trainable* model



Our Approach: End-to-End Learning for Hierarchical Forecasting

Idea: Combine the two steps to form a single *trainable* model



Ensure differentiability

- Sampling: reparameterization trick
- Projection: multiplication with a pre-determined matrix.

Experiments

Combination of forecasting methods and reconciliation techniques

- ETS, ARIMA, PERMBU (Ben Taieb et al., 2017)
- Bottom-Up (Naive), MinT, ERM

Variants of the proposed approach as an ablation study

Experiments

Combination of forecasting methods and reconciliation techniques

- ETS, ARIMA, PERMBU (Ben Taieb et al., 2017)
- Bottom-Up (Naive), MinT, ERM

Variants of the proposed approach as an ablation study

Mean quantile loss averaged over 5 runs:

method	Labour	Traffic	Tourism	Tourism-L	Wiki
ARIMA-NaiveBU	0.0453	0.0808	0.1138	0.1741	0.3771
ETS-NaiveBU	0.0431	0.0664	0.1008	0.1689	0.4672
ARIMA-MinT-shr	0.0466	0.0769	0.1171	0.1609	0.2467
ARIMA-MinT-ols	0.0463	0.1115	0.1194	0.1728	0.2782
ETS-MinT-shr	0.0454	0.0963	0.1013	0.1627	0.3622
ETS-MinT-ols	0.0458	0.1110	0.1001	0.1668	0.2701
ARIMA-ERM	0.0398	0.0466	0.5886	0.5634	0.2206
ETS-ERM	0.0455	0.1026	2.3755	0.5080	0.2217
PERMBU-MinT	0.0395±0.0003	0.0671±0.0064	**0.0763±0.0002**	—	0.2786±0.0039
Hier-E2E (Ours)	**0.0365±0.0094**	**0.0386±0.0060**	0.0843±0.0029	**0.1489±0.0039**	**0.2020±0.0082**
ablation study: DeepVAR	0.0389±0.0068	0.0403±0.0063	0.0891±0.0030	0.1525±0.0050	0.2157±0.0096
ablation study: DeepVAR+	0.0415±0.0082	0.0466±0.0069	0.0936±0.0022	0.1883±0.0244	0.2400±0.0232

Experiments

Forecast errors by levels of the hierarchy (**best in bold**, *second-best in italics*):

Dataset	Level	Hier-E2E (Ours)	DeepVAR	DeepVAR+	Best of Competing Methods
Labour	1	**0.0311±0.0120**	*0.0352±0.0079*	0.0416±0.0094	0.0406±0.0002 (PERMBU-MinT)
	2	**0.0336±0.0089**	*0.0374±0.0051*	0.0437±0.0078	0.0389±0.0002(PERBMBU-MINT)
	3	**0.0336±0.0082**	0.0383±0.0038	0.0432±0.0076	*0.0382±0.0002*(PERMBU-MINT)
	4	**0.0378±0.0060**	0.0417±0.0038	0.0448±0.0066	*0.0397±0.0003*(PERMBU-MINT)
Traffic	1	*0.0184±0.0091*	0.0225±0.0109	0.0250±0.0082	**0.0087**(ARIMA-ERM)
	2	*0.0181±0.0086*	0.0204±0.0044	0.0244±0.0063	**0.0112**(ARIMA-ERM)
	3	*0.0223±*0.0072	**0.0190±0.0031**	0.0259±0.0054	0.0255(ARIMA-ERM)
	4	**0.0914±0.0024**	*0.0982±0.0012*	0.0982±0.0017	0.1410(ARIMA-ERM)
Tourism-L	1	*0.1027±0.0062*	0.1100±0.0139	0.1370±0.0289	**0.0927** (ETS-BU)
	2	**0.1403±0.0047**	0.1485±0.0099	0.1776±0.0221	*0.1484 (ETS-BU)*
	3	**0.2050±0.0028**	*0.2078±0.0076*	0.2435±0.0170	0.2408(ETS-BU)
	4	**0.2727±0.0017**	*0.2731±0.0066*	0.3108±0.0164	0.3291(ETS-BU)
Tourism	1	**0.0402±0.0040**	0.0519±0.0057	0.0508±0.0085	*0.0472±0.0012* (PERBMBU-MINT)
	2	*0.0658±0.0084*	0.0755±0.0011	0.0750±0.0066	**0.0605±0.0006** (PERBMBU-MINT)
	3	*0.1053±0.0053*	0.1134±0.0049	0.1180±0.0053	**0.0903±0.0006** (PERBMBU-MINT)
	4	*0.1223±0.0039*	0.1294±0.0060	0.1393±0.0048	**0.1106±0.0005** (PERBMBU-MINT)
Wiki	1	**0.0419±0.0285**	0.0905±0.0323	*0.0755±0.0165*	0.1558 (ETS-ERM)
	2	**0.1045±0.0151**	0.1418±0.0249	*0.1289±0.0171*	0.1614 (ETS-ERM)
	3	*0.2292±0.0108*	0.2597±0.0150	0.2583±0.0281	**0.2010**(ETS-ERM)
	4	*0.2716±0.0091*	0.2886±0.0112	0.3108±0.0298	**0.2399**(ETS-ERM)
	5	0.3720±0.0150	*0.3664±0.0068*	0.4460±0.0271	**0.3507**(ETS-ERM)

Conclusions

- A global model that generates coherent, probabilistic forecasts
- Generic approach: can swap DeepVAR with another multivariate model
- Can handle general, domain-specific constraints
- Easy-to-implement: Available as a `gluonts` model
- Forth-coming at ICML 2021

What's next in neural forecasting

- Gradient-boosted-tree-like out-of-the-box accuracy
- Meta and transfer learning
- Explainability and Causality
- More work on multi-variate models
- Graph Neural Networks
- GANs (maybe, I'm not yet convinced for time series because we don't need the "discriminator")
- Neural ODEs
- more deep probabilistic models, especially for irregular and intermittent time series
- how to take advantage of multi-resolution
- much more...

What's next in neural forecasting

- Gradient-boosted-tree-like out-of-the-box accuracy
- Meta and transfer learning
- Explainability and Causality
- More work on multi-variate models
- Graph Neural Networks
- GANs (maybe, I'm not yet convinced for time series because we don't need the "discriminator")
- Neural ODEs
- more deep probabilistic models, especially for irregular and intermittent time series
- how to take advantage of multi-resolution
- much more...

For an overview of further models, check out https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/ and [Benidis et al., 2020]

Coffee Break (20 mins) (back at 8:10pm)

GluonTS github repository: https://github.com/awslabs/gluon-ts

References

Alexandrov, A. and other (2019). GluonTS: Probabilistic Time Series Models in Python. *arXiv preprint arXiv:1906.05264*.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview.

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

Gasthaus, J., Benidis, K., Flunkert, V., Salinas, D., Wang, Y., and Januschowski, T. (2019a). Probabilistic forecasting with spline quantile function RNNs. In *Proceedings of AISTATS*, volume 89, pages 1901–1910.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. (2019b). Probabilistic forecasting with spline quantile function rnns. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1901–1910. PMLR.

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

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems 32*, pages 5244–5254. Curran Associates, Inc.

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

Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*.

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*.

References (cont.)

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.

Rangapuram, S. S., Seeger, M., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems 32*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Turkmen, A. C., Wang, Y., and Januschowski, T. (2019). Intermittent demand forecasting with deep renewal processes.

van den Oord, A., Dieleman, Sanderand 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 *The 9th ISCA Speech Synthesis Workshop*.

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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. (2019). Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617.

Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

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

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.