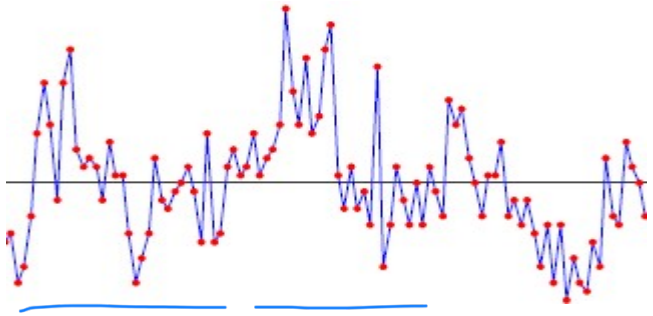
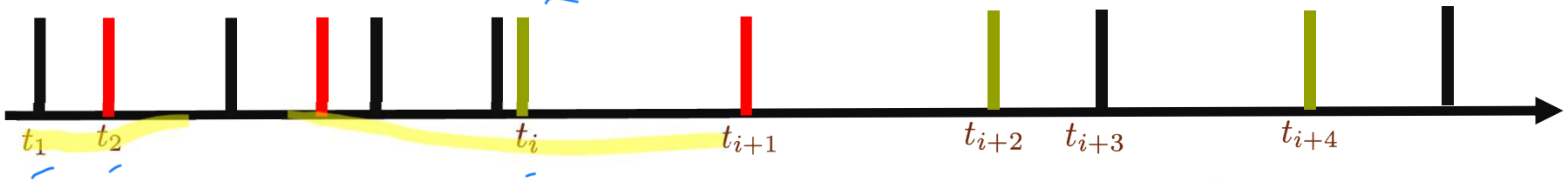
Temporal Sequences

Sunita Sarawagi

Temporal Sequences



- Regular time-series
  - Daily traffic on individual webpages from different regions in Wikipedia
  - Hourly load on various servers of different services in a Data center
  - Monthly demand for products from different regions in Flipkart
- Irregular Event sequences
  - User visits to a music service and the song played
  - Event logs of a system
  - Attacks on a system



Temporal Sequences

- A temporal sequence

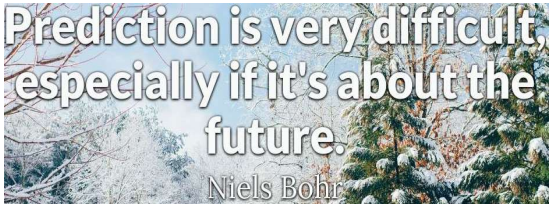
$$(t_1^i, x_1^i, y_1^i), (t_2^i, x_2^i, y_2^i), \ldots, (t_n^i, x_n^i, y_n^i)$$

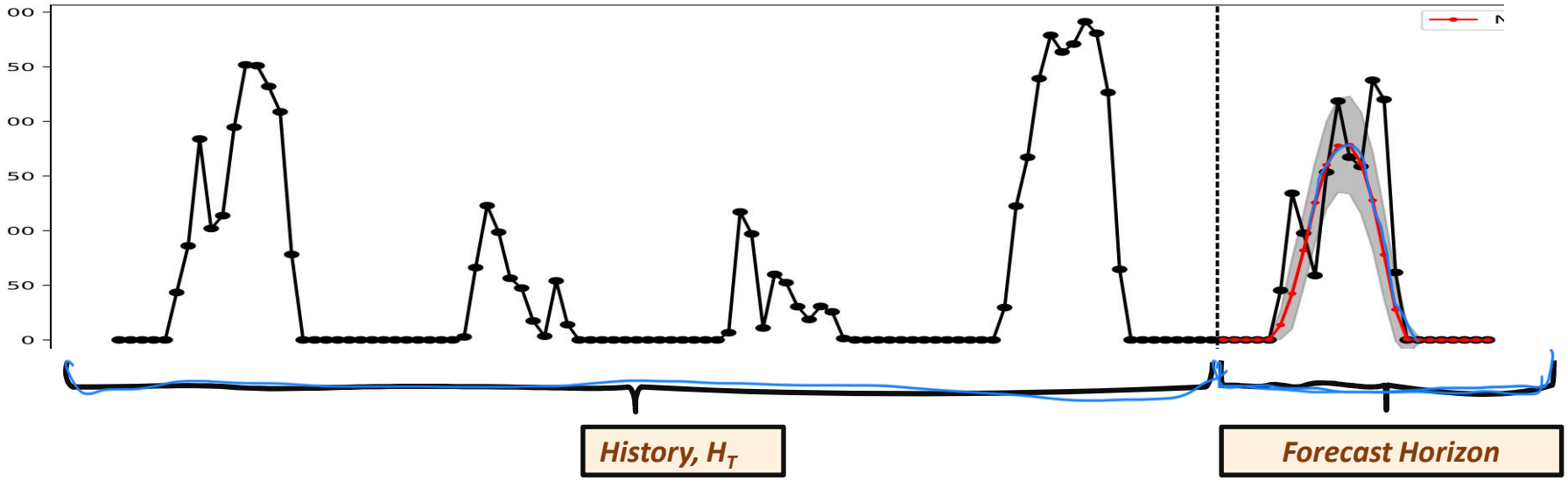
Time Input features Value

- Several such sequences indexed by i
  - Sequences are related to each other
  - The index i could span a multi-dimensional space e.g. server and application, or product-id and region.

Tasks on Temporal sequences

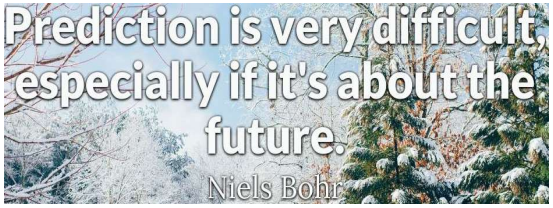
- Probabilistic Forecasting:
  - short Vs long-term.

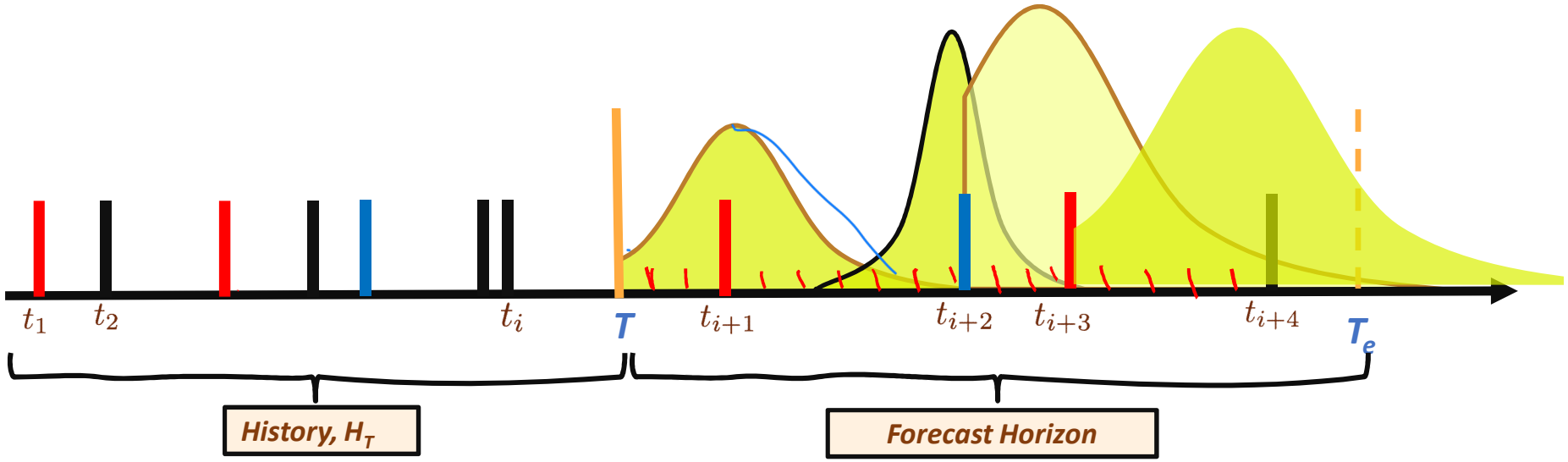




Tasks on Temporal sequences

- Forecasting in event sequences:
  - Predict both time of event and type of event.





Tasks on Temporal Sequences

- Outlier detection
- Missing value imputation

Our focus here is on forecasting.

Key modelling questions often carry over to other tasks.

Conventional methods

- Statistical time-series methods e.g. ARIMA, Box-Jenkins
  - Local, cannot handle features x_j, non-stationarity, interactions
- State space models e.g. Kalman filters
  - Simplifying assumption (linear Gaussian) may not hold in practice.
- Classical machine learning: e.g. Regression methods
  - Domain experts featurize history e.g. wavelets, Fourier coefficient
  - Powerful regressors like Boosted regression trees to predict y
- Can deep learning provide higher gains than these?

Demand Forecasting in Flipkart

- Eight weeks out inventory management
- Huge catalogue of products --- 100 millions.
- Sold across different regions.
- Large number of confounding factors.
- Large divergence in scale and variance in the output values

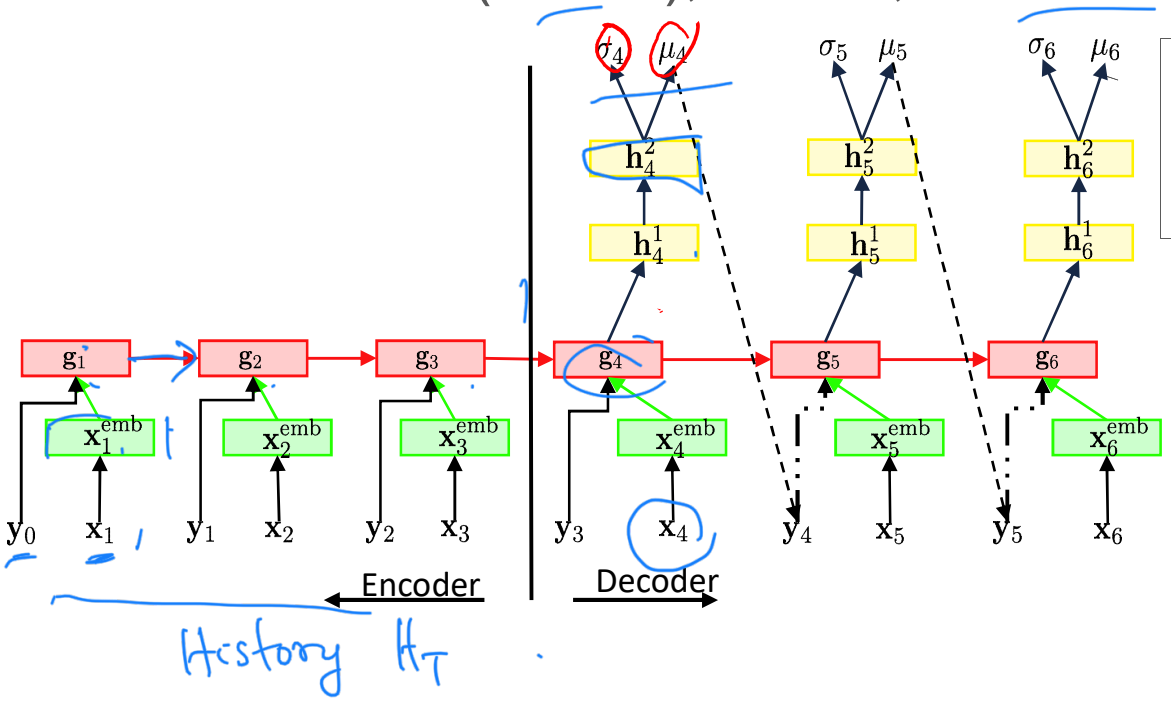
AR-MDN: Associative and Recurrent Mixture Density Networks for eRetail Demand Forecasting, 2018

Large number and variety of features

Demand Factor	Feature Name	Description	Data Type
Product	Product ID	Captures "latent" factors of the product	1-Hot
	Product Tier	Tiers are based on popularity of products	Categorical
	Historical Out-of-Stock %	Assumed to be always In-Stock during Testing phase	Numerical
Product Visibility	Sale event type	Category-level of the Sale event	Categorical
	Deal Card		Binary
	Banner on Homepage		Binary
Price	Listed Price	Govt. mandated Max. Retail Price	Numerical
	Discounted Price	Price offered by Flipkart	Numerical
	Effective Price	Final price after cash-back, product exchange, freebie, etc.	Numerical
	Difference in price from historical mean	Captures "stable" price of the product	Numerical
	Historical Min Price		Numerical
	Historical Max Price		Numerical
	Avg. discount on similar products	Products in the same category are considered similar	Numerical
Convenience	No-cost EMI		Binary
	Product Exchange		Binary
	Exclusive to Flipkart		Binary
Time	Week of the month	Captures seasonality	Numerical
	Lag feature - Price	Mean of past n weeks' price	Numerical
	Lag feature - Sale	Mean of past n weeks' sale units	Numerical
	Time elapsed since last price change	Sales peak when price changes, but reverts quickly to previous volume	Numerical
	Time since last event or upcoming event	Units sold dips immediately before and after a Sale event	Numerical
	Time since first order	Captures "Newness" of Product	Numerical

Neural Model for Time Series Forecasting

Encode history using neural sequence model: Recurrent Neural Network (RNNs), CNNs, Transformer w. self-attention



Flexible distributions over predictions

Gaussian: $P(y_j|.) \sim N(\widehat{\mu_j}, \widehat{\sigma_j^2})$

Mixture of Gaussians, Quantile…

Multi-layer RNN capture context

Feed-forward network transform input features

Predicting a Gaussian distribution on forecasts

$$\mathbf{g}_n^i = RNN([y_{j-1}^i, \mathbf{x}_j^i] : j = 1 \ldots n | \theta_{\text{enc}})$$

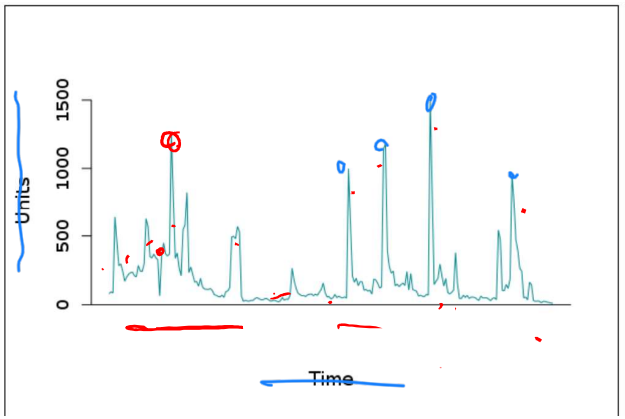
$$\mathbf{h}_j^i = FF([\mathbf{g}_n^i, \mathbf{x}_j^i] : j = n+1 \ldots n+K | \theta_{\text{dec}})$$

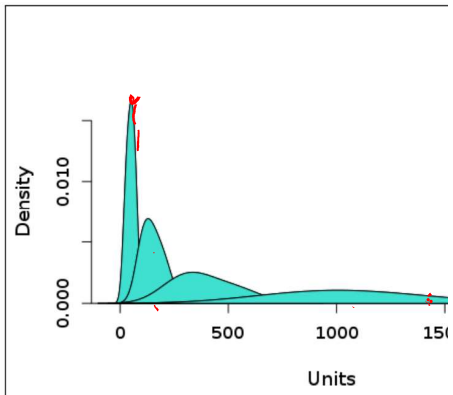
$$\mu_j^i = \theta_\mu [\mathbf{h}_j^i, 1], \quad \sigma_j^i = \log(1 + \exp(\theta_\sigma [\mathbf{h}_j^i, 1]))$$

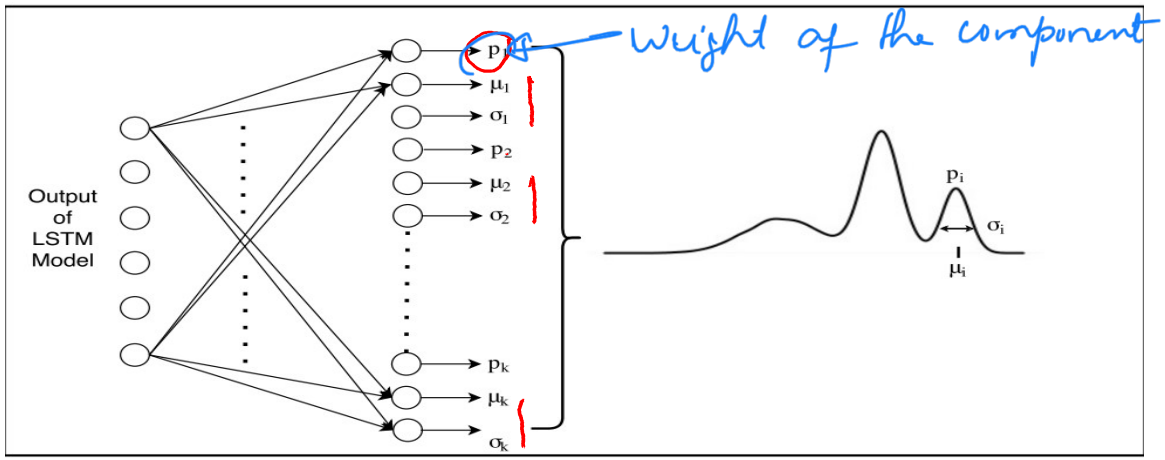
$$\Pr(y_j^i | \mathbf{x}_j^i, (\mathbf{x}_1^i, y_1^i), \ldots, (\mathbf{x}_n^i, y_n^i)) = \mathcal{N}(\mu_j^i, \sigma_j^i)$$

Output Values highly multi-modal

- Single Gaussian insufficient
- Use mixture of Gaussians
- Last layer outputs k mixture components, means, variance.



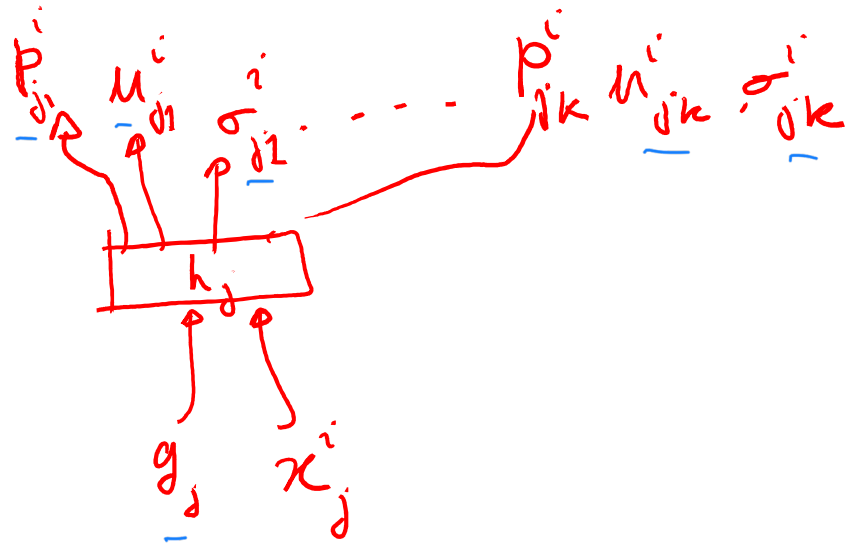


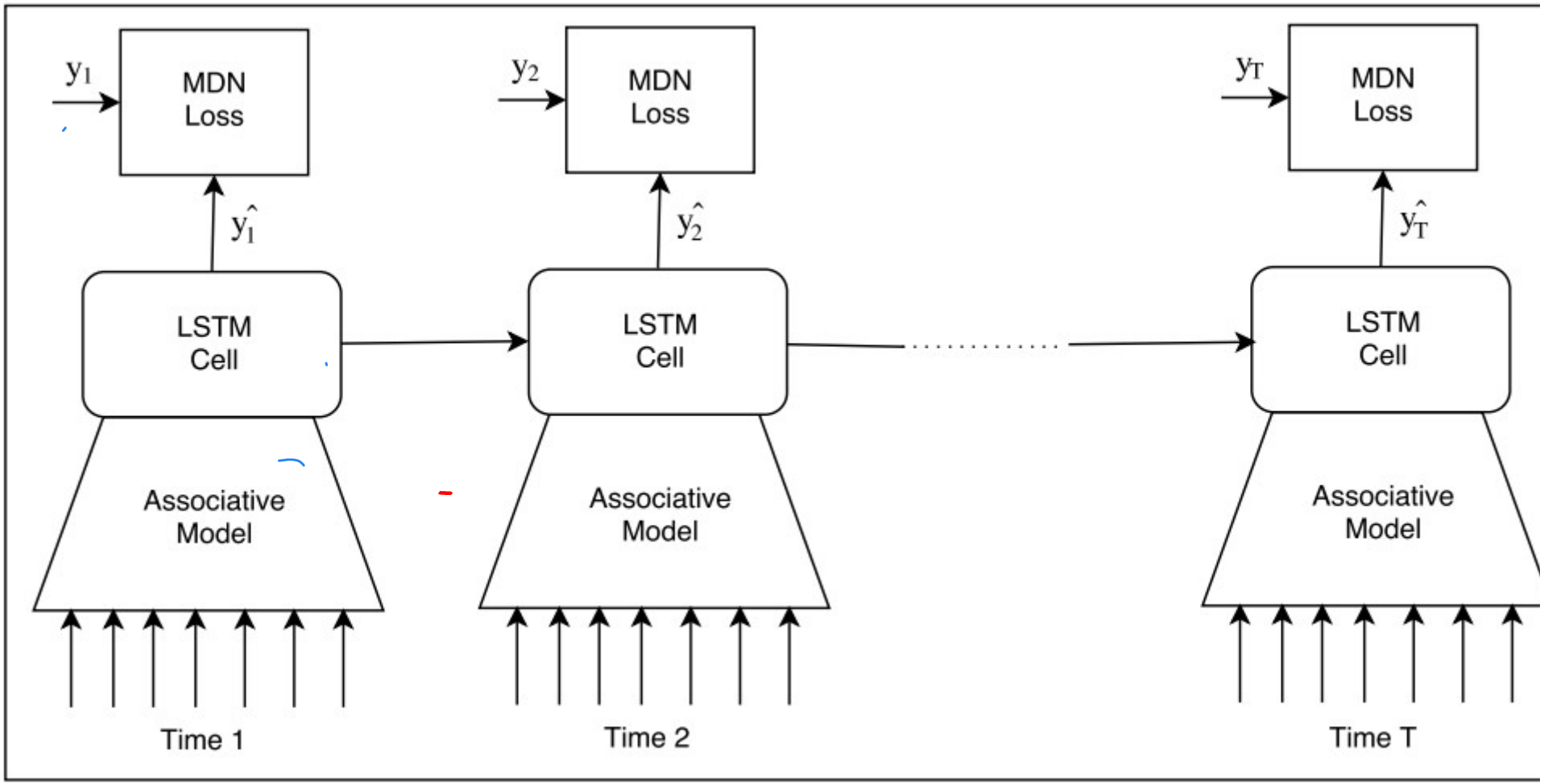


weight of the component

0.1, 0.1, 0.2, 100, 200,

$$P(y_j^i \mid H_i, x_j^i) = \sum_{k=1}^{K} p_{jk}^i \, \mathcal{N}(y_j^i \mid \mu_{jk}^i, \sigma_{jk}^i)$$




y_1
MDN Loss
$\hat{y_1}$
LSTM Cell
Associative Model
Time 1
y_2
MDN Loss
$\hat{y_2}$
LSTM Cell
Associative Model
Time 2
y_T
MDN Loss
$\hat{y_T}$
LSTM Cell
Associative Model
Time T

Training

- Full joint distribution modelled and maximized

$$\sum_{i:series} \sum_{j:positions} \log P\left(y_j^i \middle| \theta\left(x_j^i, \left(x_1^i, y_1^i\right), \ldots, \left(x_{j-1}^i, y_{j-1}^i\right)\right)\right)$$

$$\sim \mathcal{N}\left(y_j^i \middle| \mu_j^i, \sigma_j^{i\,2}\right)$$

- Parameters $\boldsymbol{\theta}$ trained end to end jointly over all series
- Training is efficient and easily parallelizable

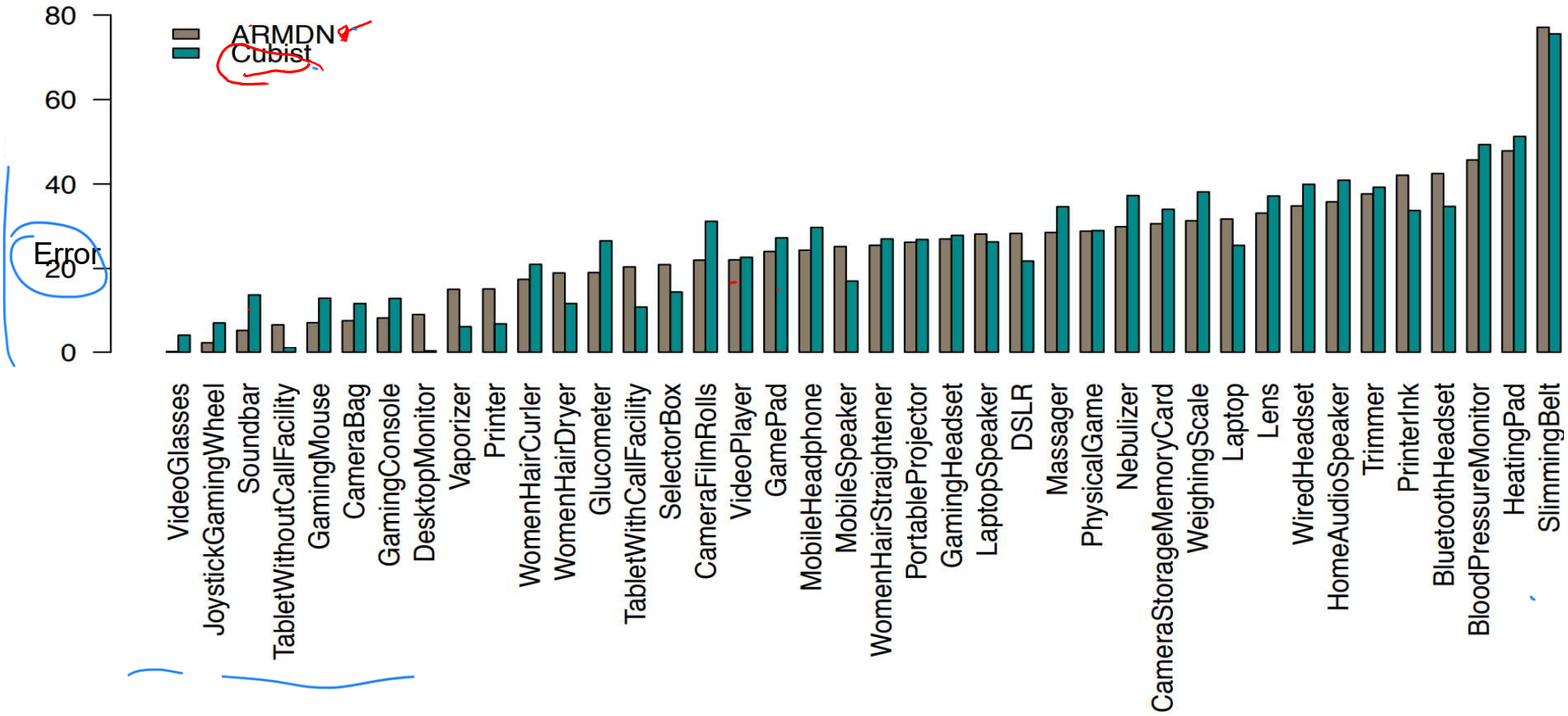
$$\nabla_\theta \log P(y_j^i \mid \hat{\mu}_j^i, \hat{\sigma}_j^{i2}) = \nabla_\theta \frac{-\frac{1}{2}(y_j^i - \hat{\mu}_j^i(\theta))^2}{\sigma_j^{i2}(\theta)} - \log \sigma(\theta)$$

RNN parameters
input encoding layers
output layers.

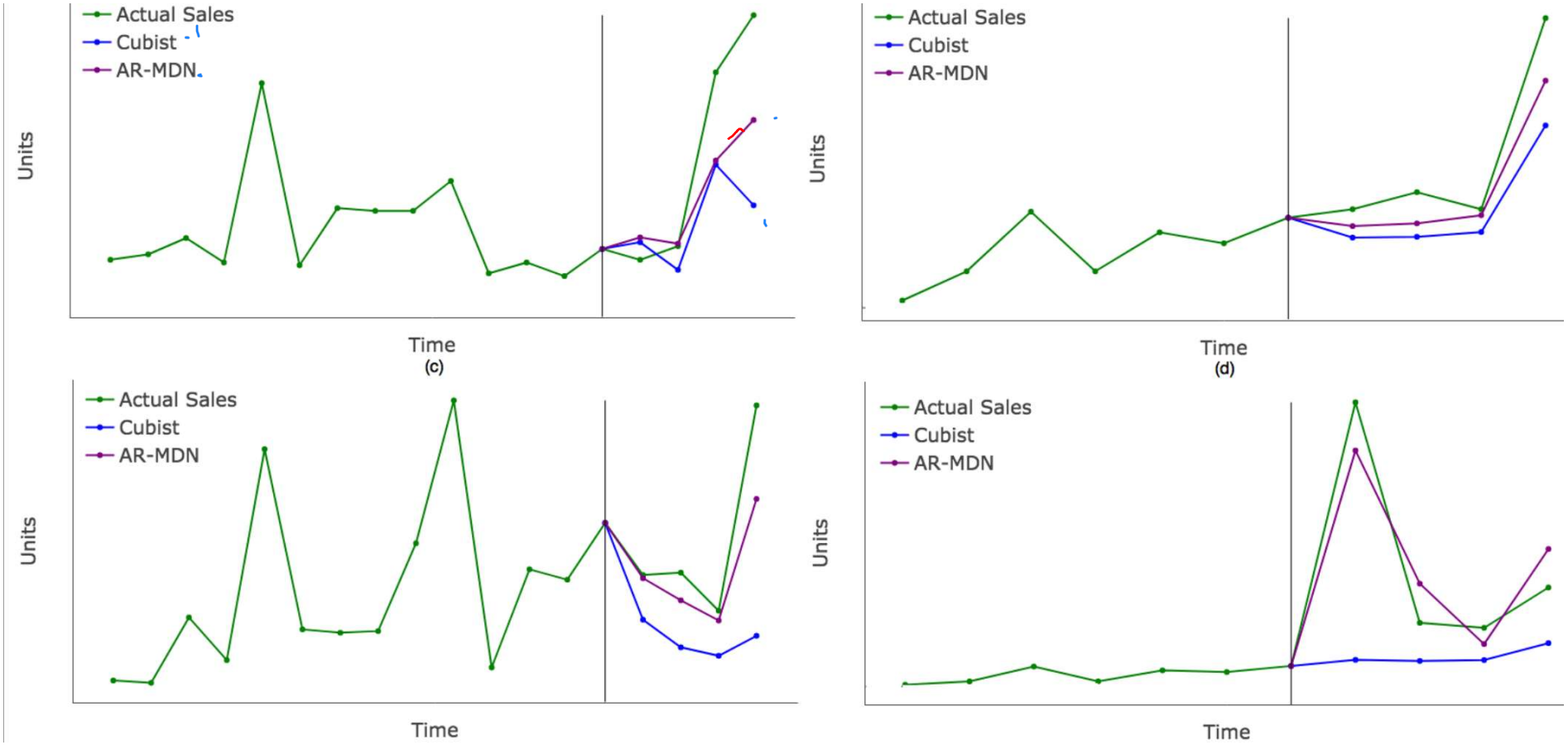
MoG:

$$\nabla_\theta \log \sum_{k=1}^{K} p_{jk}(\theta) \frac{e^{-\frac{1}{2}(y_j^i - \mu_{jk}^i(\theta))^2}}{\sqrt{\sigma(\theta)}\ \sigma_{jk}^{i2}(\theta)}$$

Neural method better than SOTA non-neural method



Some anecdotes



Limitations of this model

- Does not output the joint distribution over multiple predictions it makes both along time and along different items.
- Tools to address these limitations
  - Gaussian Processes
  - Gaussian Copula

How do you detect if your data is a time series?

If " data follows the distribution - al assumption.

Check whether the standardized values: $z_{ij} = \frac{y_{ij} - \hat{u}_{ij}}{(\sigma_j^i)}$

used for modeling likelihood.

Do $\{z_{ij}\}$ values look like samples from a $N(0,1)$ distribution?