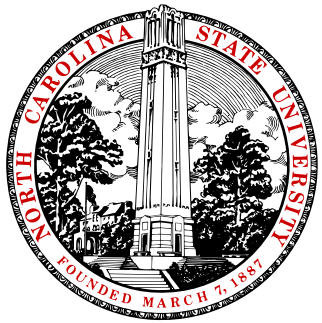
Deep Learning for Scenario Generation and Scenario Reduction in Short-Term Power System Operations

Wenyuan Tang

Department of Electrical and Computer Engineering

North Carolina State University



IEEE PES Big Data & Analytics Tutorial Series

February 26, 2021

Outline

Decision Making Under Uncertainty

- A two-stage stochastic programming approach

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \mathrm{E}_\omega[Q(\boldsymbol{x},\omega)] \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \boldsymbol{X} \end{aligned}$$

where $Q(\boldsymbol{x},\omega)$ is the optimal value of the second-stage problem

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & \boldsymbol{q}(\omega)^T\boldsymbol{y} \\ \text{subject to} \quad & \boldsymbol{T}(\omega)\boldsymbol{x} + \boldsymbol{W}(\omega)\boldsymbol{y} = \boldsymbol{h}(\omega) \\ & \boldsymbol{y} \in \boldsymbol{Y} \end{aligned}$$

- In the first stage, the here-and-now decisions $\boldsymbol{x}$ are made
- Then the random outcome ω is realized
- In the second stage, the wait-and-see decisions (also known as recourse actions) $\boldsymbol{y}$ are made, which depend on both $\boldsymbol{x}$ and the realization of ω

Decision Making Under Uncertainty

- An equivalent form of the two-stage stochastic programming problem

$$
\begin{aligned}
\underset{\boldsymbol{x}, \boldsymbol{y}(\omega), \forall \omega}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} + \mathrm{E}_\omega [\boldsymbol{q}(\omega)^T \boldsymbol{y}(\omega)] \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\
& \boldsymbol{T}(\omega)\boldsymbol{x} + \boldsymbol{W}(\omega)\boldsymbol{y}(\omega) = \boldsymbol{h}(\omega), \ \forall \omega \\
& \boldsymbol{x} \in \boldsymbol{X} \\
& \boldsymbol{y}(\omega) \in \boldsymbol{Y}, \ \forall \omega
\end{aligned}
$$

- A deterministic form when the sample space is finite ($\pi(\omega)$: probability of ω)

$$
\begin{aligned}
\underset{\boldsymbol{x}, \boldsymbol{y}(\omega), \forall \omega}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} + \textstyle\sum_\omega \pi(\omega) \boldsymbol{q}(\omega)^T \boldsymbol{y}(\omega) \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\
& \boldsymbol{T}(\omega)\boldsymbol{x} + \boldsymbol{W}(\omega)\boldsymbol{y}(\omega) = \boldsymbol{h}(\omega), \ \forall \omega \\
& \boldsymbol{x} \in \boldsymbol{X} \\
& \boldsymbol{y}(\omega) \in \boldsymbol{Y}, \ \forall \omega
\end{aligned}
$$

Short-Term Power System Operations

- Net-zero carbon emissions by 2050: increasing penetration of renewable generation
- Intermittent and non-dispatchable nature of renewable generation such as wind: challenges in renewable energy integration
- New operating paradigms for unit commitment and economic dispatch: from a deterministic approach to a stochastic programming approach

A stochastic day-ahead scheduling problem

minimize	commitment cost + expected dispatch cost
subject to	power balance constraints
	transmission constraints
	generating unit constraints

- Commitment cost: start-up cost, no-load cost, etc.
- Dispatch cost: generation cost, load shedding penalty, etc.
- Generating unit constraints: generation capacity constraints, ramping constraints, minimum up-time/down-time constraints, etc.

Scenarios in Stochastic Programming

- What is the randomness in the stochastic day-ahead scheduling problem?
- Random outcome ω: wind power generation (or its forecast error) on the next day
- Stochastic process $\boldsymbol{\xi}(\omega)$: mapping ω to a real-valued vector indexed by time

$$\boldsymbol{\xi}(\omega) = (\xi_1(\omega), \ldots, \xi_{24}(\omega))$$

- It is hard to characterize or find the multivariate probability distribution of $\boldsymbol{\xi}$
- Even if the true distribution is known, the resulting stochastic programming problem may not be computationally tractable: some discretization is needed

Approximating a stochastic process by scenarios

- A scenario is a single realization of the stochastic process

$$\boldsymbol{s} = (s_1, \ldots, s_{24})$$

- We typically assume equally likely scenarios

$$\pi(\boldsymbol{s}) = 1/|\boldsymbol{S}|, \ \forall \boldsymbol{s} \in \boldsymbol{S}$$

Scenarios of Wind Power Generation

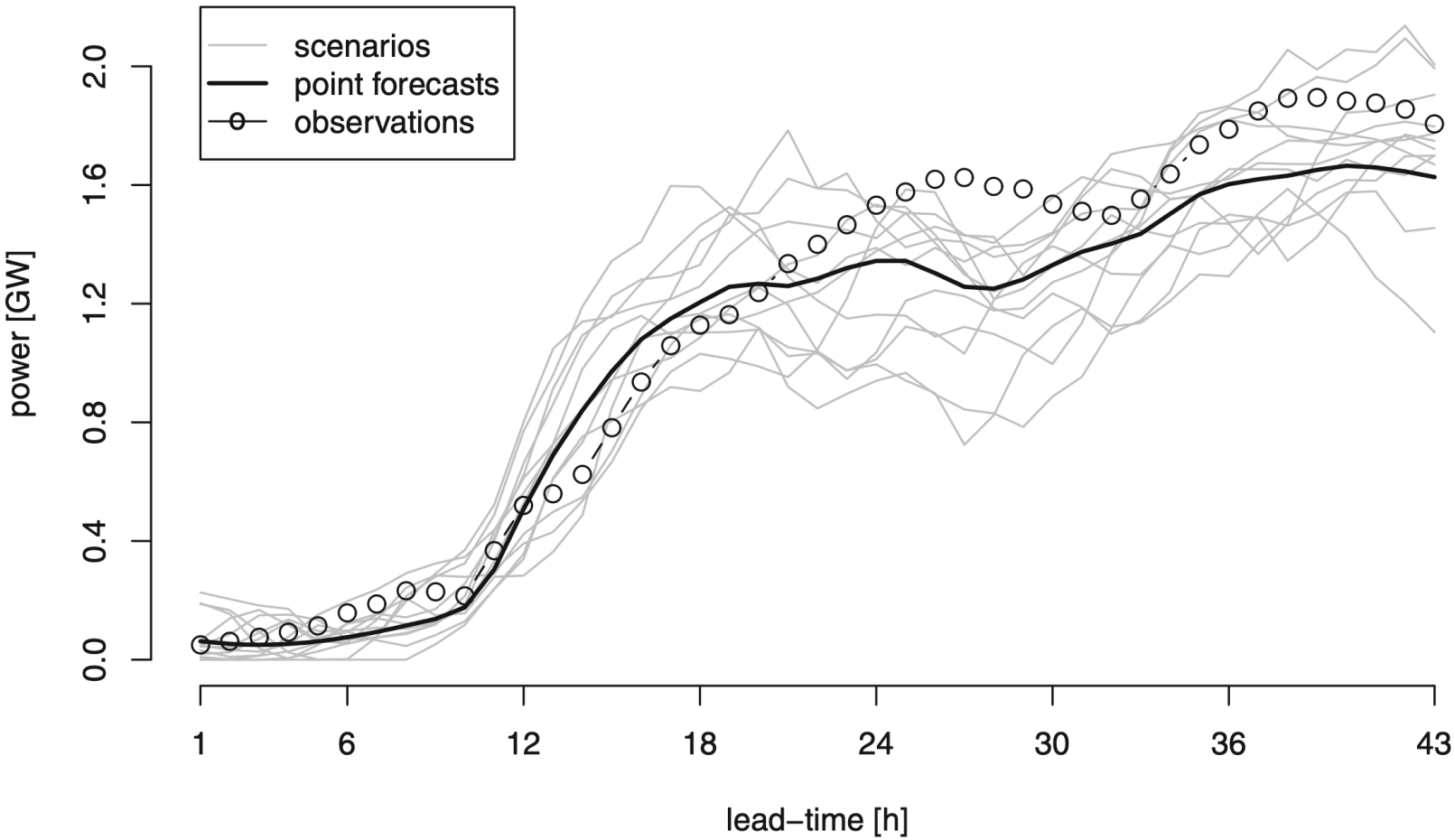


Figure: Scenarios (12 time trajectories) of wind power generation issued on 4th April 2007 at 00:00 UTC for the whole onshore capacity of Western Denmark (Source: [Morales et al. 2014])

Scenario Generation

- Scenarios can be viewed as a type of forecasts
- Solving the stochastic programming problem: sample average approximation (SAA)

$$\mathrm{E}_{\boldsymbol{\xi}}[Q(\boldsymbol{x}, \boldsymbol{\xi})] \approx \frac{1}{|\boldsymbol{S}|} \sum_{\boldsymbol{s} \in \boldsymbol{S}} Q(\boldsymbol{x}, \boldsymbol{s})$$

- The SAA method has desirable asymptotic properties
- To yield high quality decisions, we need to build a high quality scenario set
- Scenario generation: given historical time series data, generate a number of representative sample paths into the future

Traditional approach

- Use historical data to fit a statistical model (such as ARIMA)
- Then generate random samples from the statistical model (Monte Carlo sampling)

- However, tractable statistical models with simplified assumptions may not well capture the underlying temporal dynamics

Scenario Generation

Non-parametric methods

- Radial basis function network [Sideratos et al. 2012]
- Fuzzy prediction interval [Sáez et al. 2015]
- Regular vine copula [Wang et al. 2018]
- Sparse Bayesian learning, kernel density estimation [Lin et al. 2019]
- Infinite Markov switching autoregressive [Xie et al. 2019]

- However, those supervised models require extensive domain knowledge for feature selection, and are difficult to tune and implement
- Most of the methods utilize information at multiple sites to improve the accuracy: they may not be applicable when only a single time series is in consideration
- Unsupervised models are preferred: generative adversarial networks (GANs) [Goodfellow et al. 2014] and many variants of GANs, including the sequence generative adversarial network (SeqGAN) [Yu et al. 2017]
- We will adapt the SeqGAN for scenario generation of hourly wind power generation (or its forecast error) on the next day

Scenario Reduction

- Good discrete approximations often require the generation of a large number of scenarios, which may render the underlying optimization problem intractable, especially in the presence of integer variables (such as unit commitment decisions)
- To regain tractability, we need to trim down the number of scenarios while deteriorating the accuracy of the approximation as little as possible
- Scenario reduction: given a large scenario set, generate a small scenario set that is representative of the original one
- Scenario reduction can be viewed as a time series clustering task
- Like common clustering algorithms, the generated scenarios do not necessarily belong to the original scenario set

Traditional approach

- Define a measure (such as the Euclidean distance) to quantify the similarity or the distance between two scenario sets
- Solve an optimization problem (which is generally nonconvex) to maximize the measure

Scenario Reduction

Measure based methods

- Fortet–Mourier probability metrics [Heitsch and Römisch 2009]
- Space and moment distance [Li et al. 2016]
- Radial basis kernel function [Wang et al. 2017]
- Nested distance [Beltrán et al. 2017]
- Probability metrics and correlations [Hu and Li 2019]

- However, those measure based methods depend on a predefined measure, which may deteriorate the generalization capability
- It is difficult to describe all the patterns in the large scenario sets, since the number of scenarios is reduced dramatically
- Time series clustering methods are preferred: smoothed formulation of dynamic time warping (DTW) [Cuturi and Blondel 2017], and mixture of autoencoders [Dizaji et al. 2017, Chazan et al. 2019]
- We will integrate a mixture of autoencoders with fuzzy clustering for scenario reduction of forecast errors of hourly wind power generation on the next day

Outline

Fully Connected Neural Network

- The deep in deep learning: successive layers of increasingly meaningful representations, which are learned jointly via neural networks
- Fully connected neural network (FNN): every neuron (unit) in each layer is connected to every neuron in the next layer
- Universal approximation theorems imply that neural networks can represent a wide variety of interesting functions when given appropriate weights

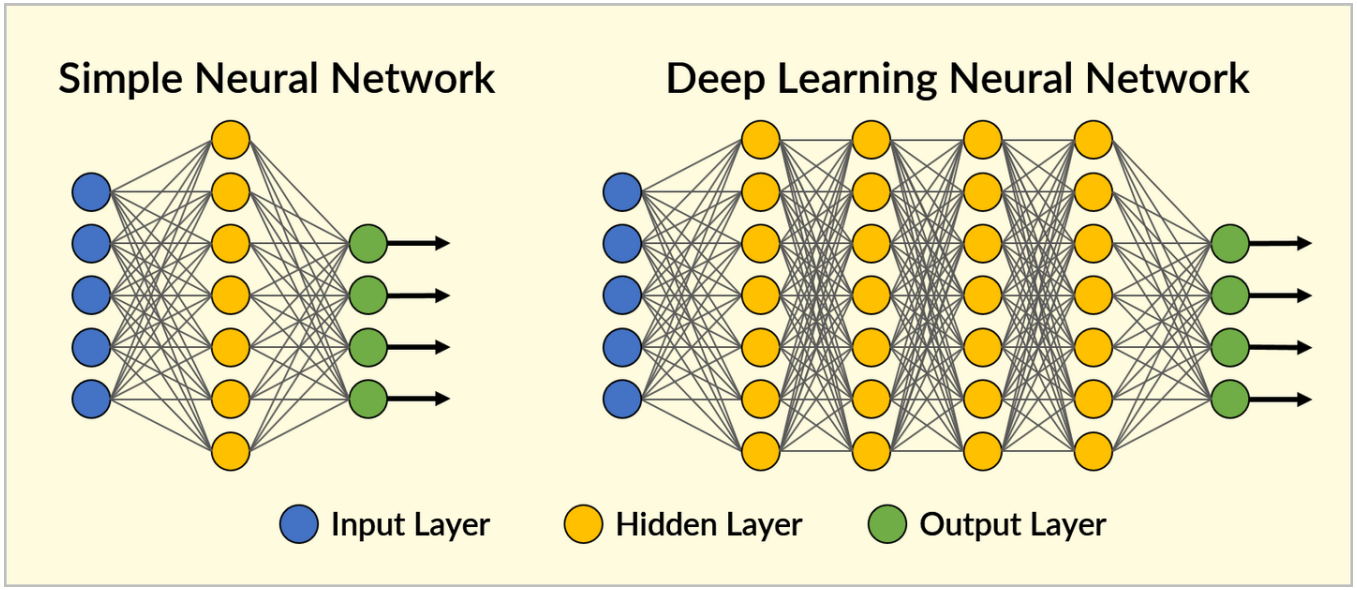


Figure: Simple and deep FNNs (Source: Internet)

Convolutional Neural Network

- Convolutional neural network (CNN): learn translation-invariant patterns and spatial hierarchies of patterns
- The convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches, producing an output feature map
- Filters (in the depth axis) encode specific aspects of the input data
- The max-pooling operation downsamples feature maps
- Finally, we need the fully connected layers to learn the global patterns

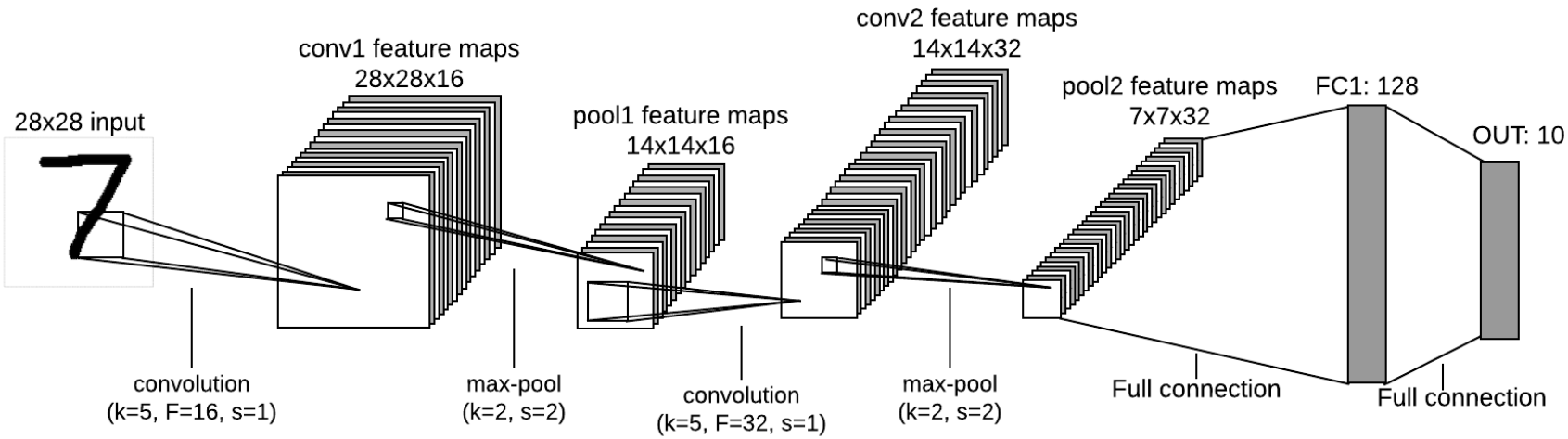


Figure: CNN for handwritten digit classification (Source: Internet)

Recurrent Neural Network

- FNNs and CNNs have no memory: they are called feedforward networks
- Recurrent neural network (RNN): process sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far
- In theory, the simple RNN is able to retain information about inputs seen many time steps before
- In practice, such long-term dependencies are impossible to learn, due to the vanishing gradient problem

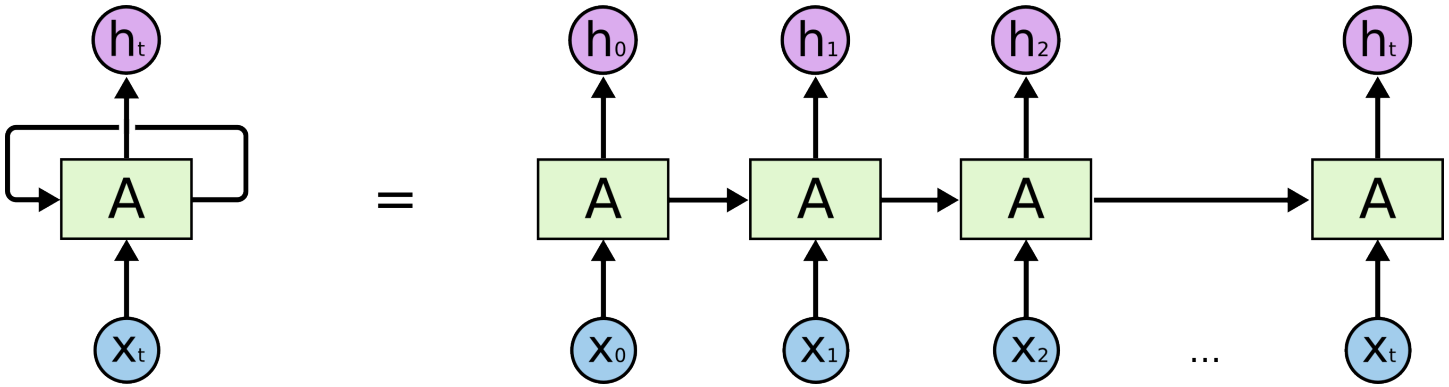


Figure: Simple RNN (`https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

Long Short-Term Memory

- Long short-term memory (LSTM): a special kind of RNNs that are explicitly designed to remember information for long periods of time
- Four interacting neural network layers (instead of one) in the repeating module
- Forget gate: what information to throw away from the cell state
- Input gate: what new information to store in the cell state
- Output gate: what information to output (and going to be the next hidden state)

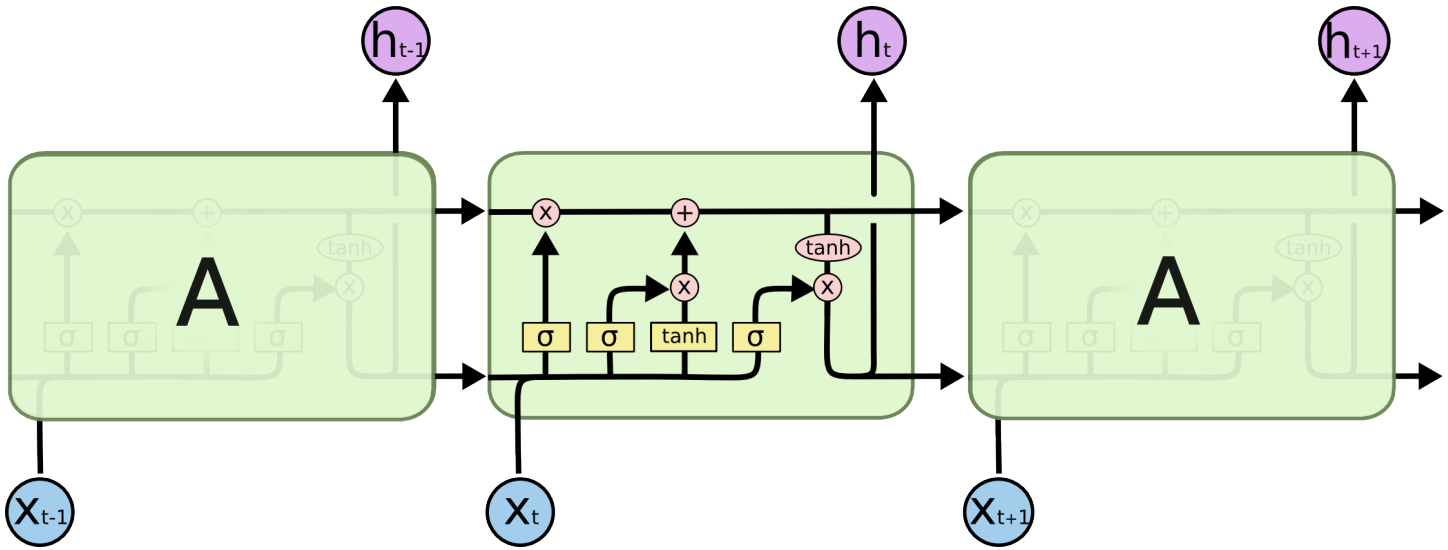


Figure: LSTM (`https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

Generative Adversarial Network

- Generative adversarial network (GAN): a generative model in which a generator network completes against an adversary, the discriminator network
- Generator: directly produce samples
- Discriminator: attempt to distinguish between samples drawn from the training data and samples drawn from the generator
- Learning in a GAN can be formulated as a zero-sum game: at convergence, the generator's samples are indistinguishable from real data

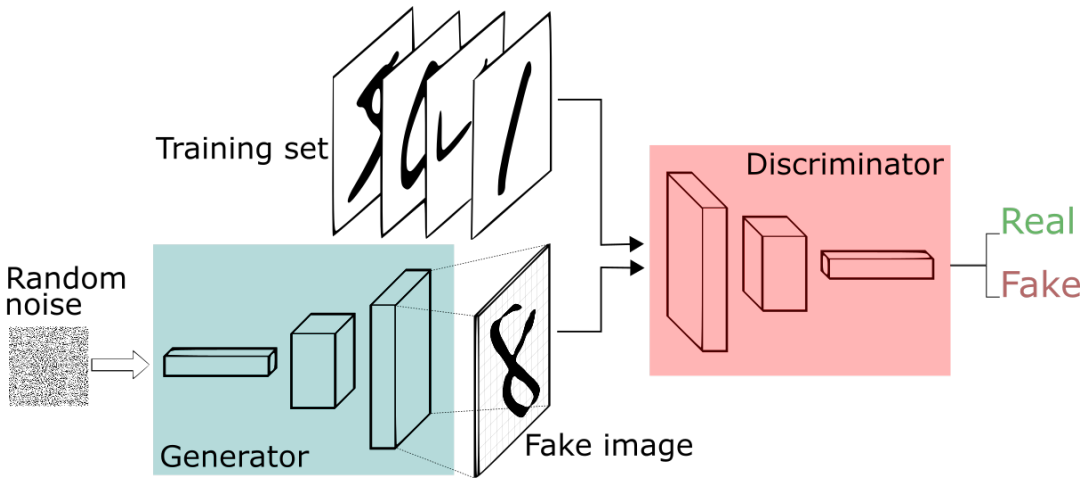


Figure: GAN (`https://sthalles.github.io/intro-to-gans/`)

Autoencoder

- Autoencoder: a neural network that is trained to attempt to copy its input to its output
- Hidden layer $\boldsymbol{h}$: describes a code (latent representation) used to represent the input
- Encoder $\boldsymbol{h} = \sigma(\boldsymbol{x})$: maps the input into the code
- Decoder $\boldsymbol{x}' = \sigma'(\boldsymbol{h})$: maps the code to a reconstruction of the input
- $\sigma'(\sigma(\boldsymbol{x})) = \boldsymbol{x}$ is not useful: autoencoders are restricted to copy only approximately
- We are interested in $\boldsymbol{h}$ (instead of $\boldsymbol{x}'$) taking on useful properties

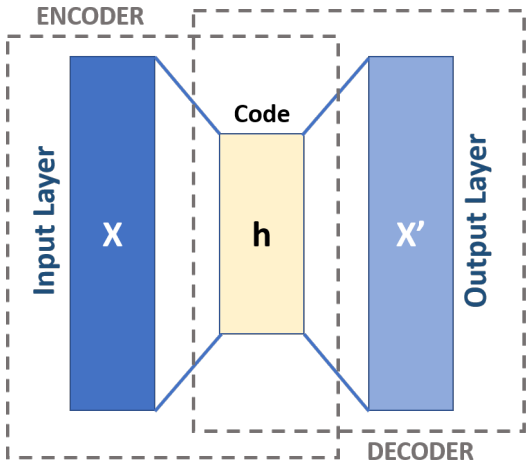


Figure: Autoencoder (`https://en.wikipedia.org/wiki/Autoencoder`)

Outline

Sequence Generative Adversarial Network

Issues when applying GAN to generating sequences

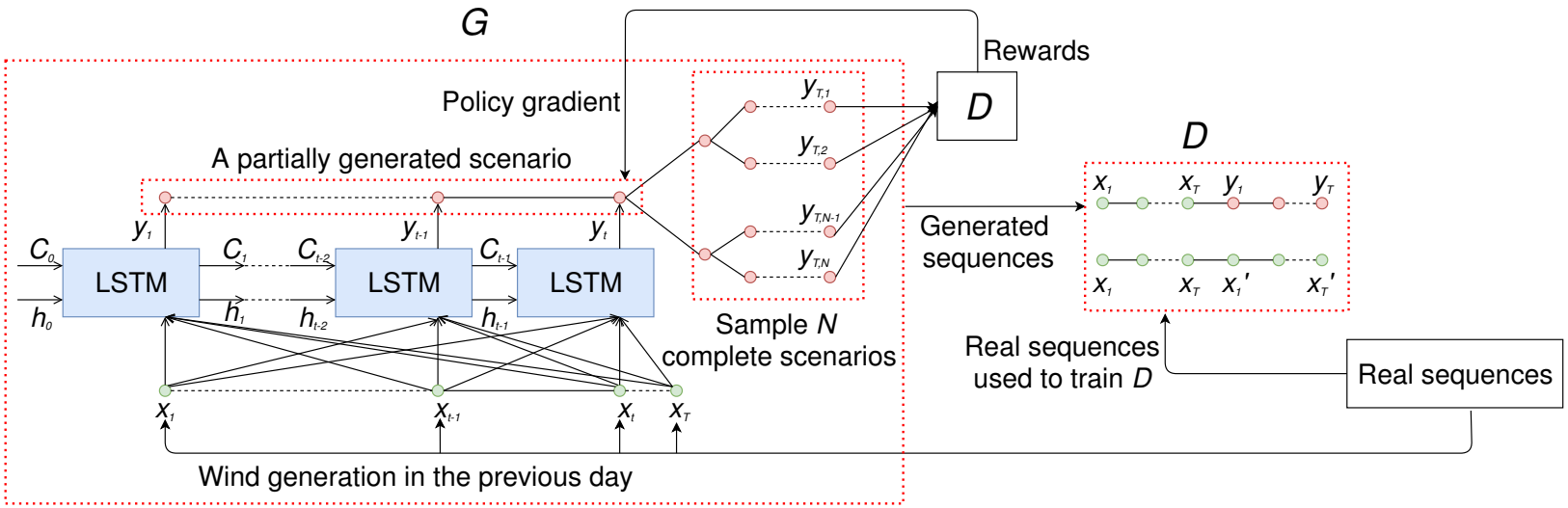
- First, GAN is designed for generating real-valued, continuous data but has difficulties in directly generating sequences of discrete tokens (such as texts)
- Second, GAN can only give the score for an entire sequence: for a partially generated sequence, it is non-trivial to balance its current score and its future score

- The first issue is minor: wind power generation (or its forecast error) is continuous; but discretization may still be preferred for convenience
- The second is critical: some values may be abnormal while the overall sequence has a good score, which may lead to unreasonable unit commitment decisions

Sequence Generative Adversarial Network (SeqGAN)

- Consider the sequence generation procedure as a sequential decision making process, solved by reinforcement learning: the state is the generated tokens so far and the action is the next token to be generated
- Regard the generative model as a stochastic parameterized policy: employ Monte Carlo search to approximate the state-action value for evaluating policy gradient

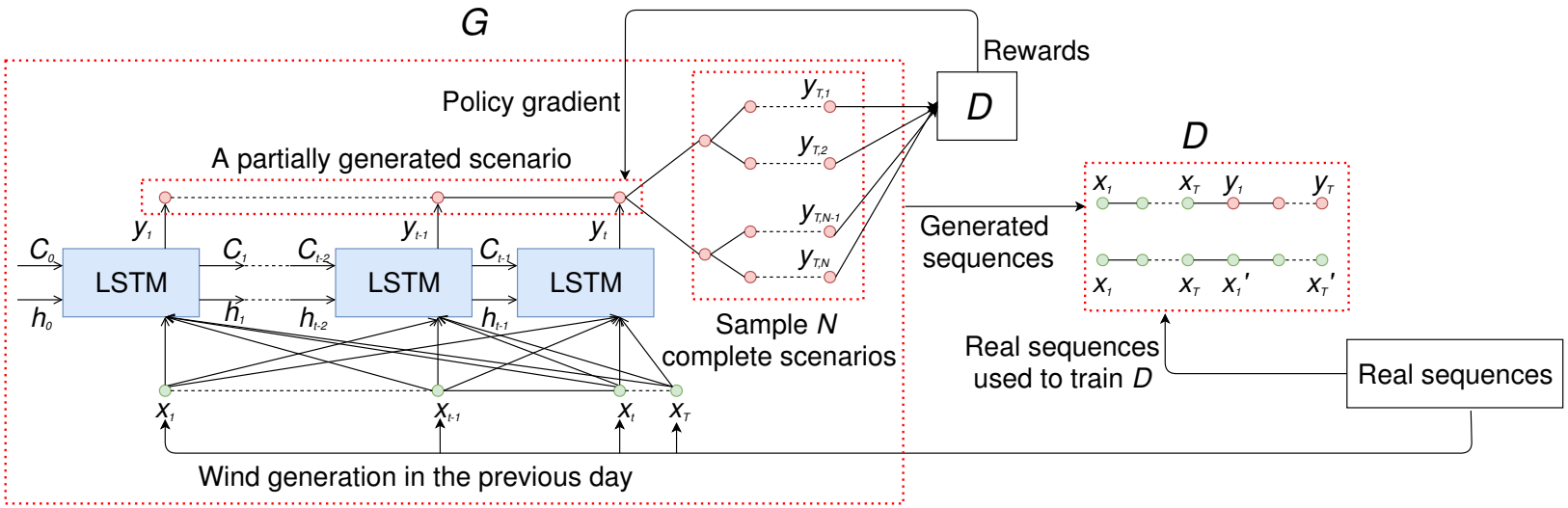
Sequence Generative Adversarial Network



- Input of the generator: 24 hourly wind power generations in the previous day
- Output of the generator: for each hour, softmax is used to obtain the discrete distribution of wind power generation (which can then be sampled for scenarios)
- The generator maximizes the reward to go from the beginning (state in t: $s_t = y_{1:t-1} = (y_1, \ldots, y_{t-1})$, action in t: y_t)

$$\max_{\theta} \; J(\theta) = \sum_{y_1} G_\theta(y_1|s_1) Q_{D_\phi}^{G_\theta}(s_1, y_1)$$

Sequence Generative Adversarial Network



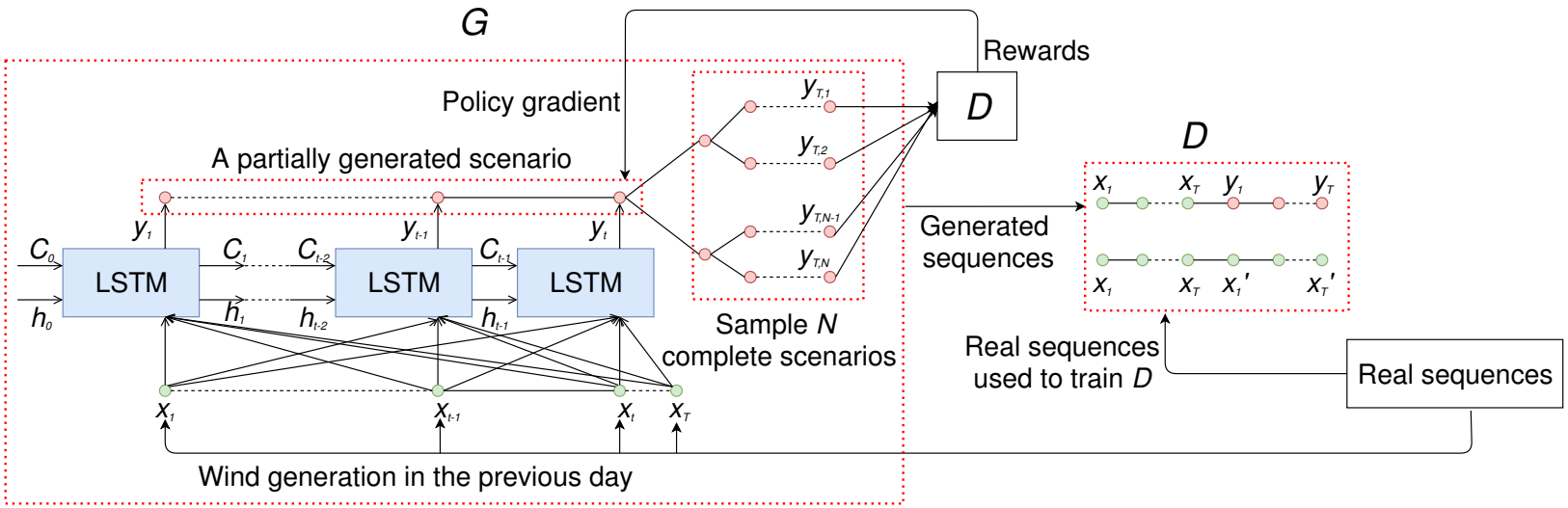
- An unbiased estimation for policy gradient (via which the generator is trained)

$$\nabla_\theta J(\theta) \approx \sum_{t=1}^{T} \mathrm{E}_{y_t \sim G_\theta(y_t|y_{1:t-1})}[(\nabla_\theta \log G_\theta(y_t|y_{1:t-1}))Q_{D_\phi}^{G_\theta}(y_{1:t-1}, y_t)]$$

- The discriminator minimizes the cross-entropy loss

$$\min_\phi -\mathrm{E}_{y\sim p_{\text{data}}}[\log D_\phi(y)] - \mathrm{E}_{y\sim G_\theta}[\log(1 - D_\phi(y))]$$

Sequence Generative Adversarial Network



- The discriminator only provides a reward value $D_\phi(y)$ (probability of being real) for a finished sequence y
- How to evaluate the state-action value function for each t?

$$Q_{D_\phi}^{G_\theta}(y_{1:t-1}, y_t) = \begin{cases} (1/N)\sum_{n=1}^{N} D_\phi(y_{1:t}, y_{t+1:T}^n), & t < T \\ D_\phi(y_{1:T}), & t = T \end{cases}$$

where $y_{t+1:T}^1, \ldots, y_{t+1:T}^N$ are sampled by Monte Carlo given $y_{1:t}$ and G_θ

Outline

Generated Scenarios of Forecast Errors

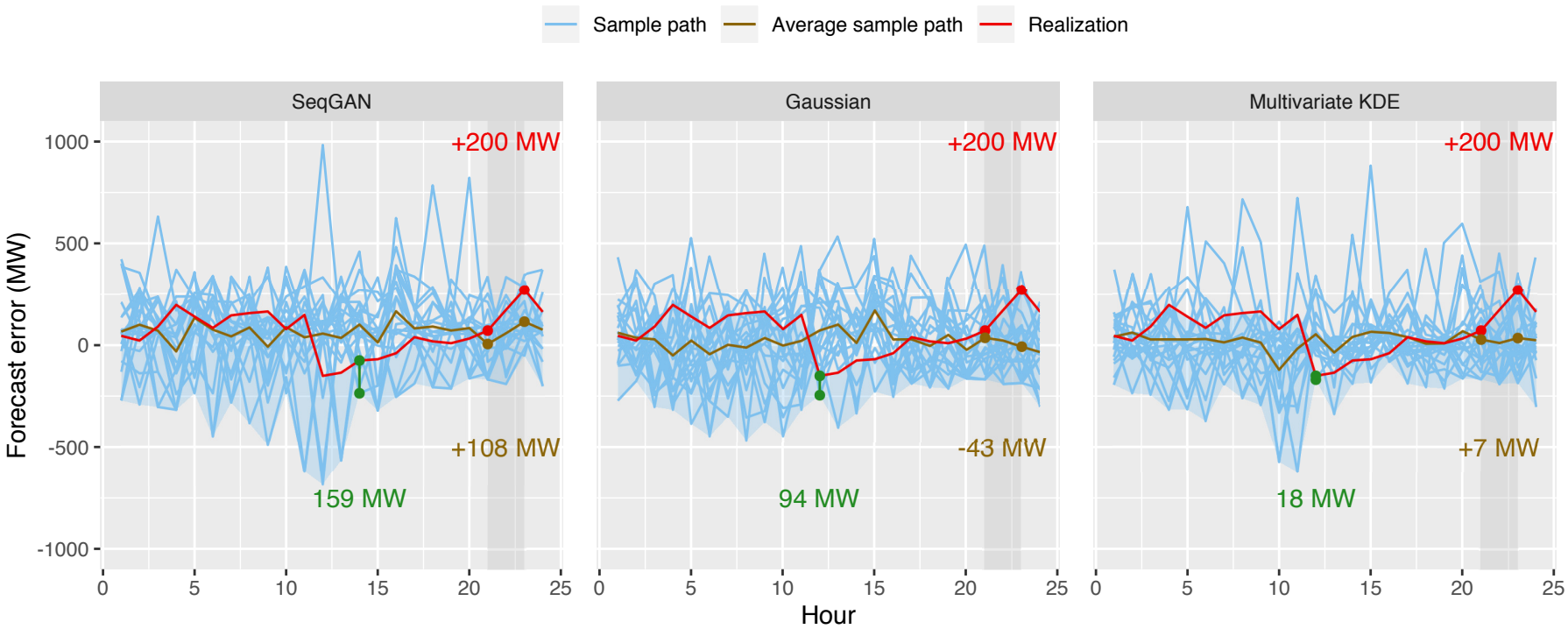


Figure: 20 scenarios of forecast errors for April 29, 2019, based on the BPA data

- The least safe margin for SeqGAN is larger: more robust decisions
- SeqGAN follows the fast ramping events better

Generated Scenarios of Wind Power Generation

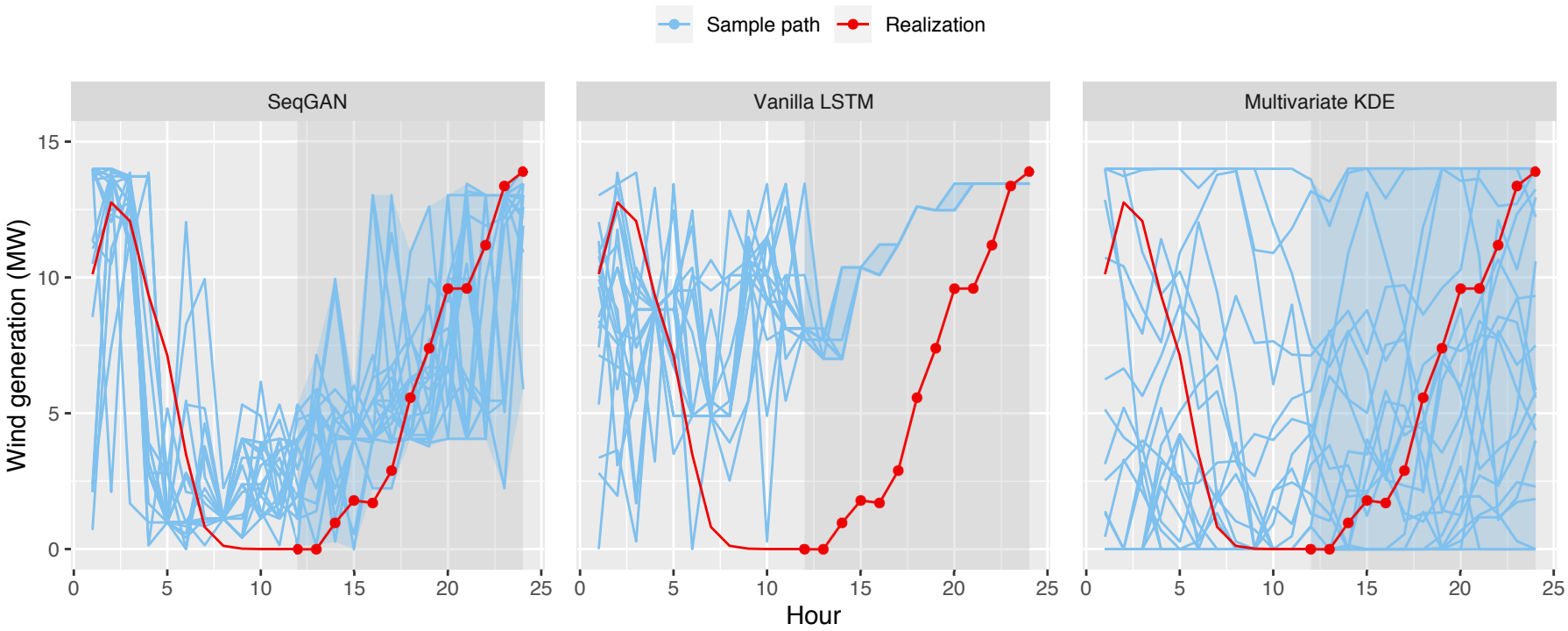


Figure: 20 scenarios of wind power generation for December 25, 2012, based on the NREL data

- SeqGAN generates more diverse scenarios which lead to more robust decisions
- SeqGAN captures the patterns better (not too diverse to be useful)

Assessment of Generated Scenarios of Forecast Errors

- Skill score: measure of accuracy of forecasts such as scenarios
- All the skills scores introduced here are negatively oriented (the lower the better)
- N generated scenarios

$$\boldsymbol{s}^{(i)} = (s_1^{(i)}, \ldots, s_{24}^{(i)}),\ i = 1, \ldots, N$$

- Observation

$$\boldsymbol{z} = (z_1, \ldots, z_{24})$$

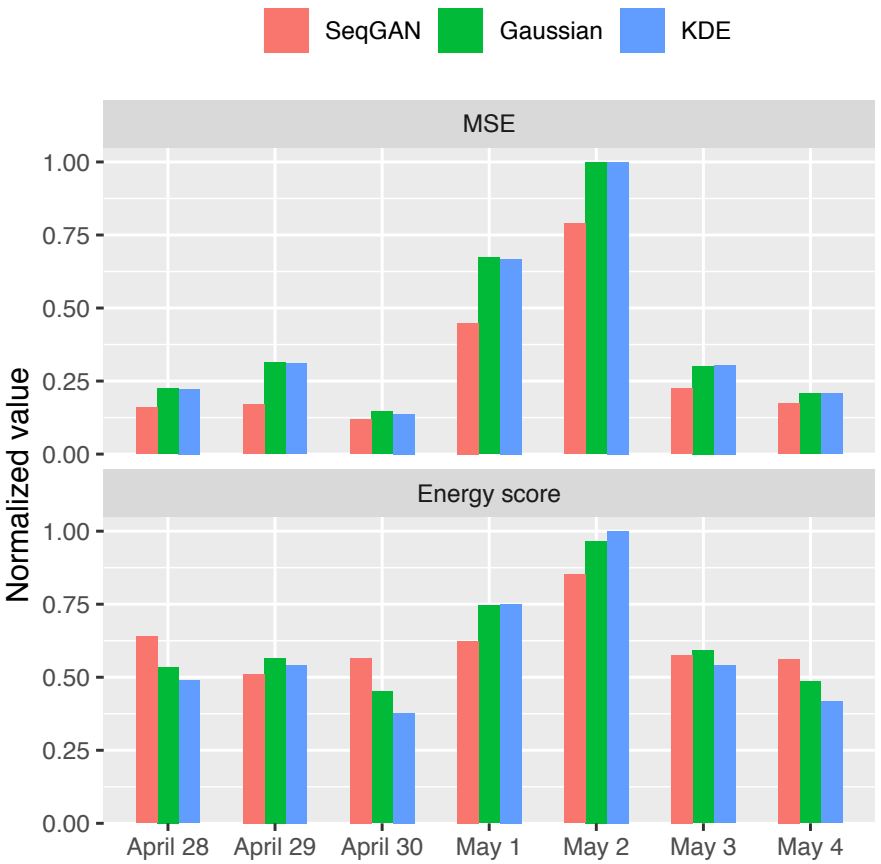
- Mean squared error (MSE)

$$\frac{1}{24}\sum_{t=1}^{24}\left(\frac{1}{N}\sum_{i=1}^{N} s_t^{(i)} - z_t\right)^2$$

- Energy score

$$\frac{1}{24}\sum_{t=1}^{24}\left(\frac{1}{N}\sum_{i=1}^{N}|z_t - s_t^{(i)}| - \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}|s_t^{(i)} - s_t^{(j)}|\right)$$

Assessment of Generated Scenarios of Forecast Errors



- SeqGAN achieves the best MSE
- SeqGAN achieves a better energy score when the uncertainty is high

Assessment of Generated Scenarios of Wind Power Generation

- For wind power generation, we are more interested in event-based measures
- Brier score: measure the deviation of the generated scenarios from the observation with respect to capturing the predefined events (such as large ramping events)

$$\frac{1}{24}\sum_{t=1}^{24}\left(\frac{1}{N}\sum_{i=1}^{N} f_t(\boldsymbol{s}^{(i)}) - f_t(\boldsymbol{z})\right)^2$$

where $f_t(\boldsymbol{s})$ indicates whether the predefined event occurs in hour t in $\boldsymbol{s}$

Event	SeqGAN	LSTM	KDE
10% up in 1 hour	**0.2148**	0.2500	0.2240
20% up in 2 hours	**0.1866**	0.2167	0.1960
10% down in 1 hour	**0.1745**	0.1930	0.2151
20% down in 2 hours	**0.1617**	0.2178	0.1917

Table: The Brier scores for 1 peak week of 1 site in the NREL data

Assessment of Generated Scenarios of Wind Power Generation

- Variogram score of order p

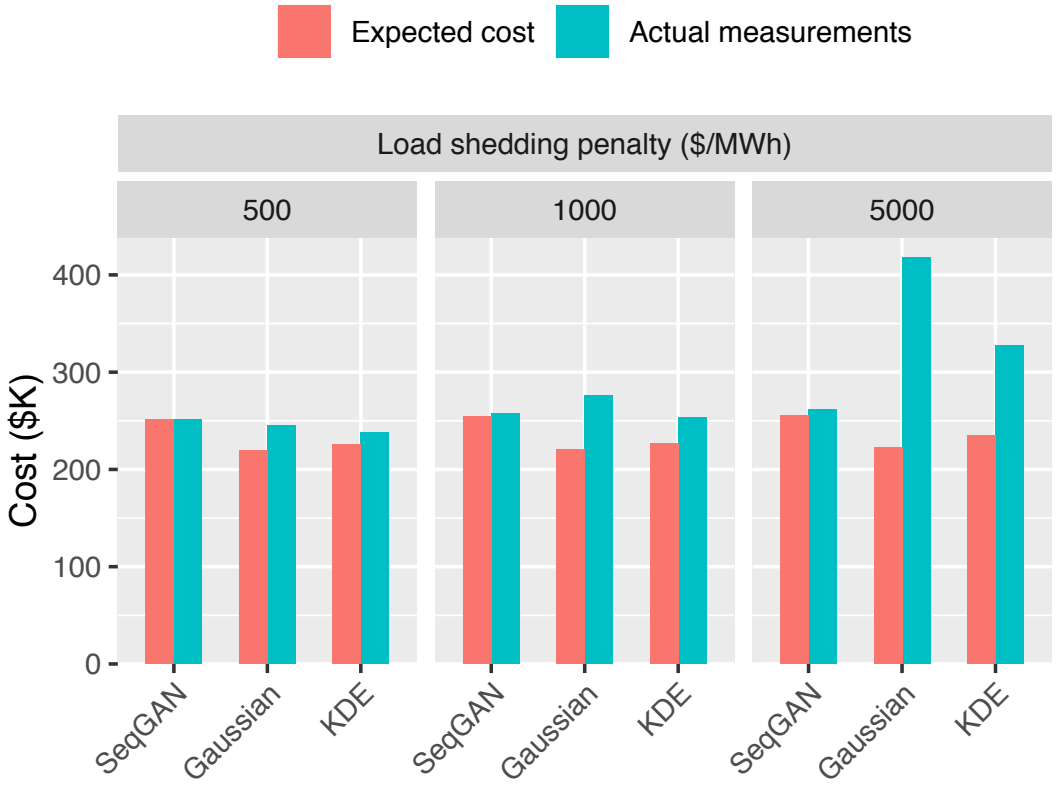
$$\sum_{t_1=1}^{24}\sum_{t_2=1}^{24}\left(|z_{t_1}-z_{t_2}|^p-\frac{1}{N}\sum_{i=1}^{N}|s_{t_1}^{(i)}-s_{t_2}^{(i)}|^p\right)^2$$

	$p=1$			$p=2$		
Case	SeqGAN	LSTM	KDE	SeqGAN	LSTM	KDE
1a	**0.505**	0.553	0.530	**0.497**	0.584	0.546
1b	**0.346**	0.350	0.388	**0.278**	0.287	0.306
2a	**0.530**	0.572	0.626	**0.448**	0.461	0.491
2b	**0.506**	0.588	0.554	**0.421**	0.538	0.457
3a	**0.462**	0.574	0.513	**0.346**	0.439	0.381
3b	**0.349**	0.389	0.432	**0.324**	0.363	0.424

Table: The Variogram scores for 1 peak and 1 off-peak week of 3 sites in the NREL data

Case Study on a Modified IEEE 24-Bus System

- Day-ahead scheduling: two-stage stochastic mixed-integer linear programming
- For each algorithm, we generate 100 scenarios, and evaluate the expected cost
- We also evaluate the realized cost which depends on the actual observation
- The scenario set generated by SeqGAN is the most representative



Outline

Scenario Reduction

- Problem statement: given N scenarios (with equal probabilities)

$$\boldsymbol{s}^{(i)} = (s_1^{(i)}, \ldots, s_{24}^{(i)}),\ i = 1, \ldots, N$$

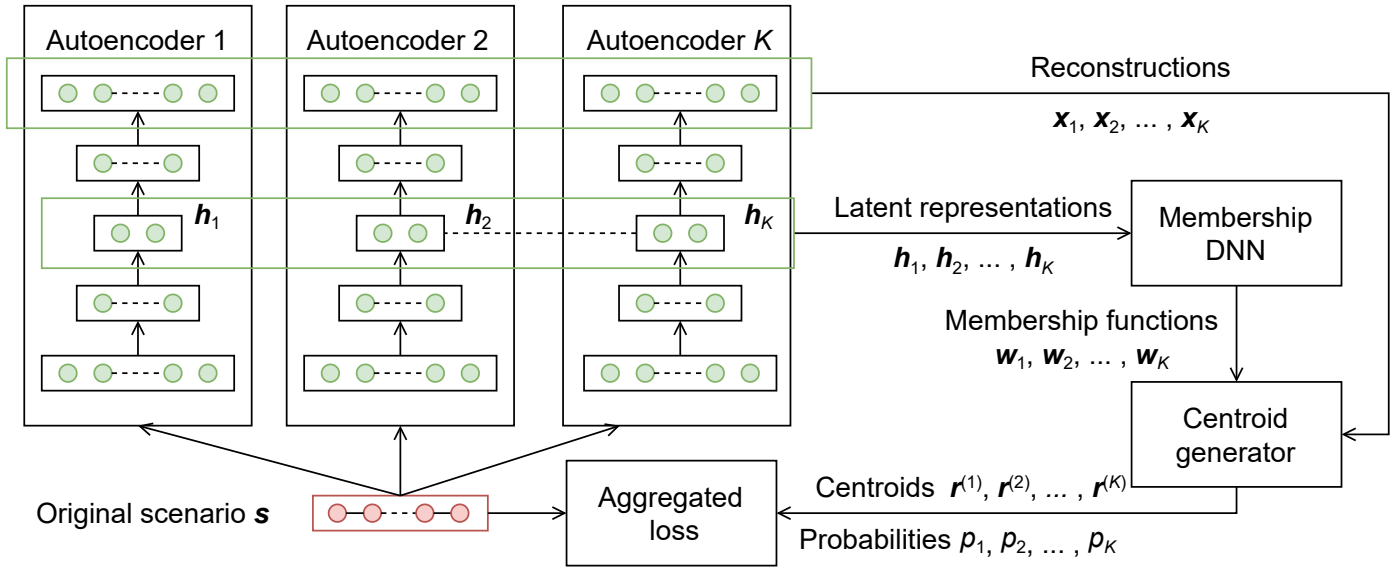
generate K ($K \ll N$) scenarios and assign probability p_k for each k

$$\boldsymbol{r}^{(k)} = (r_1^{(k)}, \ldots, r_{24}^{(k)}),\ k = 1, \ldots, K$$

Mixed Autoencoder Based Fuzzy Clustering (MAFC)

- First, several deep autoencoders map time series to latent representations
- The deep hypothesis spaces in the autoencoders provide high generalization capability and capture the complex temporal dynamics
- Second, those latent representations are fed into a membership DNN, which outputs the degree to which each scenario belongs to each cluster
- Third, based on the membership degrees, fuzzy time series clustering is employed to evaluate the centroids
- Compared with hard clustering, fuzzy clustering is better at preserving the patterns in the original scenario set

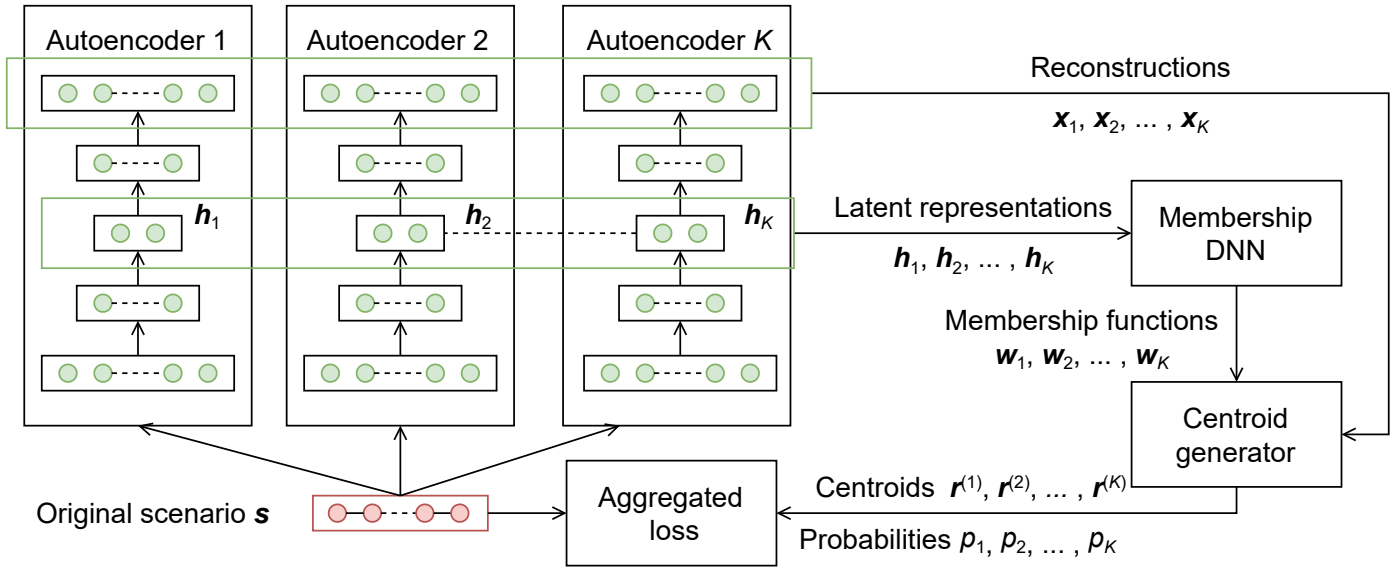
Mixed Autoencoder Based Fuzzy Clustering



- K autoencoders: extract the latent representations $\boldsymbol{h}_1^{(i)}, \ldots, \boldsymbol{h}_K^{(i)}$ of each $\boldsymbol{s}^{(i)}$
- $z_k^{(i)}$: output of the membership DNN before the last activation function (softmax)
- Membership degree $w_k^{(i)}$: the degree to which $\boldsymbol{s}^{(i)}$ belongs to cluster k

$$w_k^{(i)} = \frac{e^{z_k^{(i)}}}{\sum_{l=1}^{K} e^{z_l^{(i)}}}$$

Mixed Autoencoder Based Fuzzy Clustering

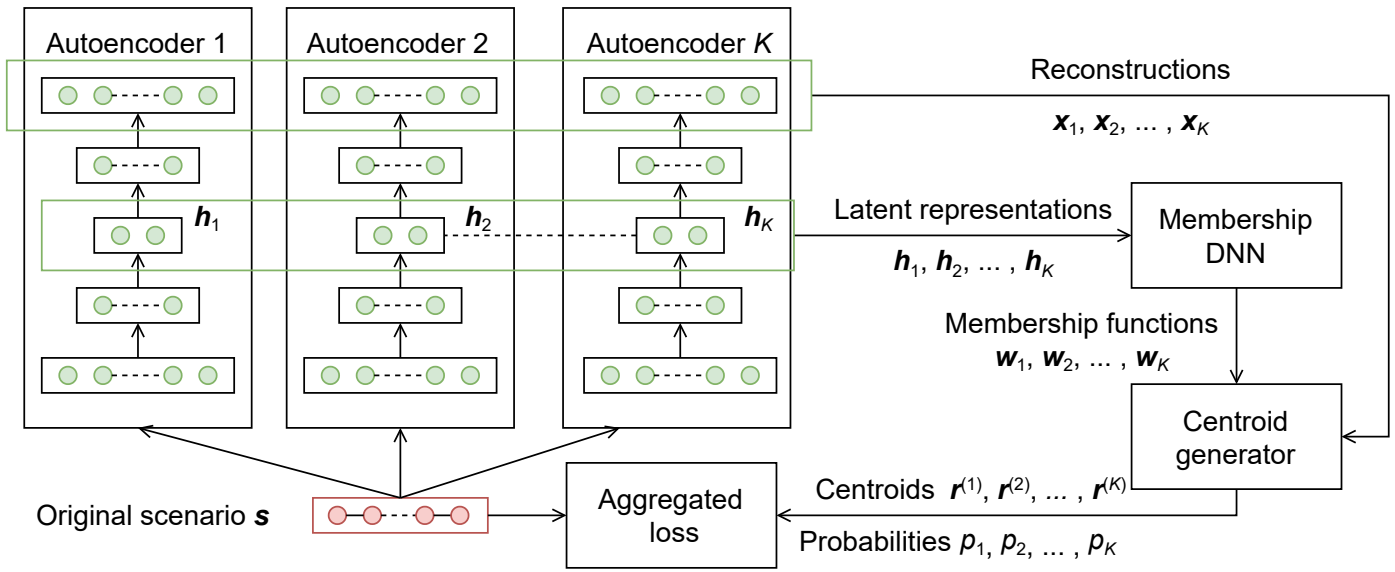


- $\boldsymbol{x}_1^{(i)}, \ldots, \boldsymbol{x}_K^{(i)}$: reconstructions of $\boldsymbol{s}^{(i)}$
- Centroids under fuzzy clustering

$$\boldsymbol{r}^{(k)} = \frac{\sum_{i=1}^{N} (w_k^{(i)})^m \boldsymbol{x}_k^{(i)}}{\sum_{i=1}^{N} (w_k^{(i)})^m}$$

where the hyperparameter m controls the level of fuzziness (the higher the fuzzier)

Mixed Autoencoder Based Fuzzy Clustering



- Assign probability p_k to each cluster k

$$p_k = \frac{1}{N} \sum_{i=1}^{N} w_k^{(i)}$$

- Loss function augmented by sample-wise entropy

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} p_k \| \boldsymbol{s}^{(i)} - \boldsymbol{c}^{(k)} \|^2 - \alpha \sum_{i=1}^{N} \sum_{k=1}^{K} (w_k^{(i)})^m \log(w_k^{(i)})$$

Outline

Setup

- Bonneville Power Administration (BPA): capacity of 4,450 MW
- Elia (Belgian grid operator): capacity of 3,667 MW
- Both data sets have day-ahead forecasts so that we can focus on forecast errors
- For each data set, we fit a kernel density estimation model, and then sample 2,000 scenarios as the original scenario set
- For each original scenario set, the task is to produce reduced scenario sets with 10 and 20 scenarios, respectively

Benchmark algorithms

- Simultaneous backward reduction (SBR)
- k-Shape
- Global alignment kernel (GAK)
- Smoothed formulation of dynamic time warping (SoftDTW)
- k-means
- Principal component analysis (PCA) with k-means
- Mixed autoencoders with DNN (MAFC without fuzzy clustering)

Assessment of Scenario Reduction

- Cosine distance

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} p_k \left(1 - \frac{\boldsymbol{s}^{(i)} \cdot \boldsymbol{r}^{(k)}}{\|\boldsymbol{s}^{(i)}\|\|\boldsymbol{r}^{(k)}\|}\right)$$

- While none is close to 0, MAFC achieves the best score

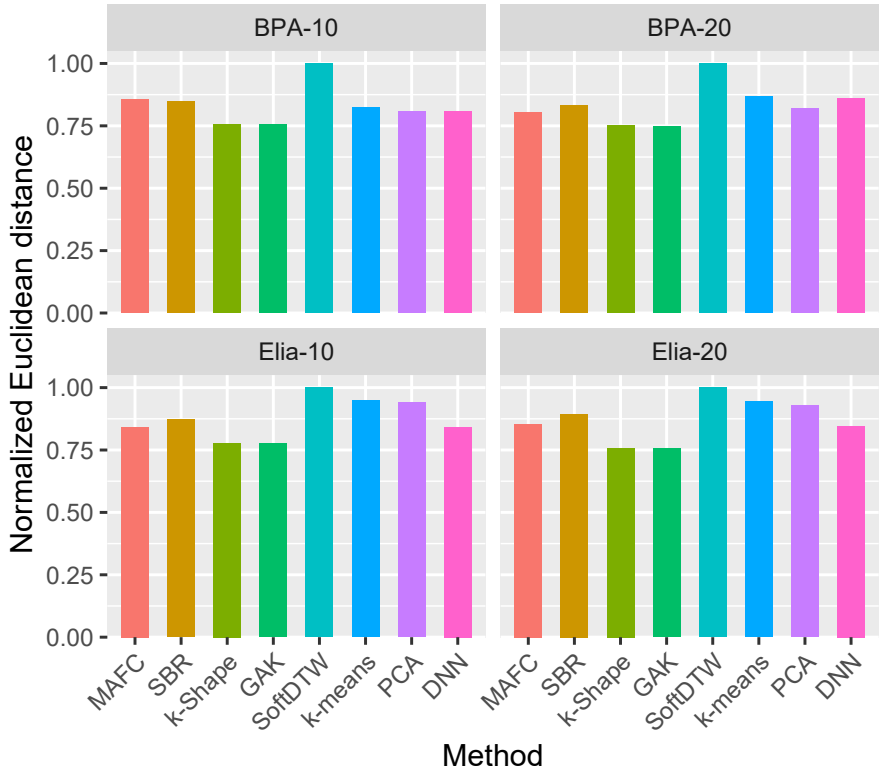
Method	BPA-10	BPA-20	Elia-10	Elia-20
MAFC	**0.885**	**0.888**	**0.896**	**0.896**
SBR	1.009	1.023	0.924	0.944
k-Shape	0.998	0.999	0.986	1.002
GAK	0.915	0.914	0.951	0.960
SoftDTW	0.981	0.986	0.959	0.974
k-means	0.944	0.962	0.962	0.966
PCA	0.935	0.942	0.942	0.954
DNN	0.905	0.896	0.928	0.914

Assessment of Scenario Reduction

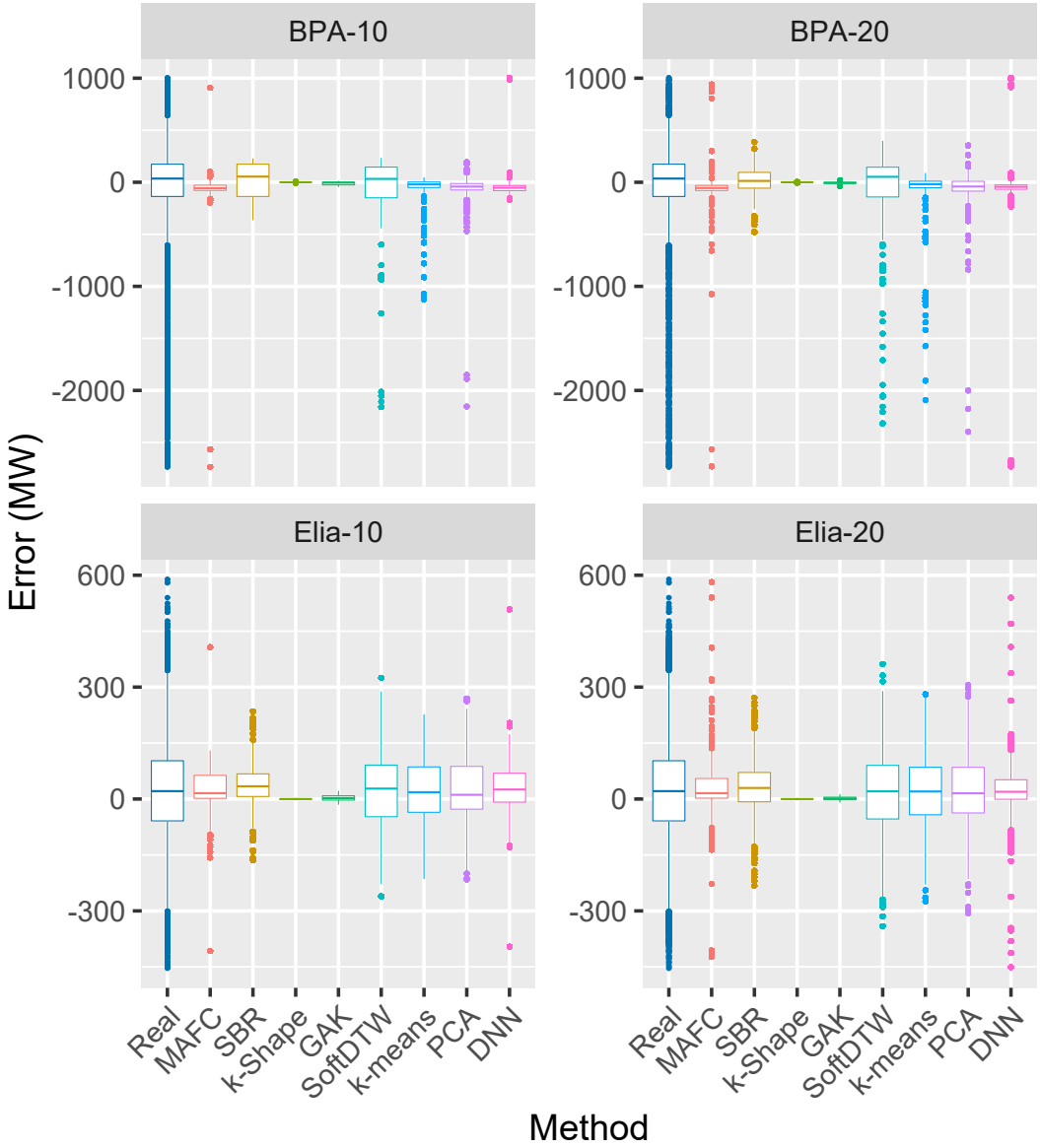
- Euclidean distance

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} p_k \left\| \boldsymbol{s}^{(i)} - \boldsymbol{r}^{(k)} \right\|^2$$

- MAFC achieves average scores, because this metric penalizes HILP scenarios

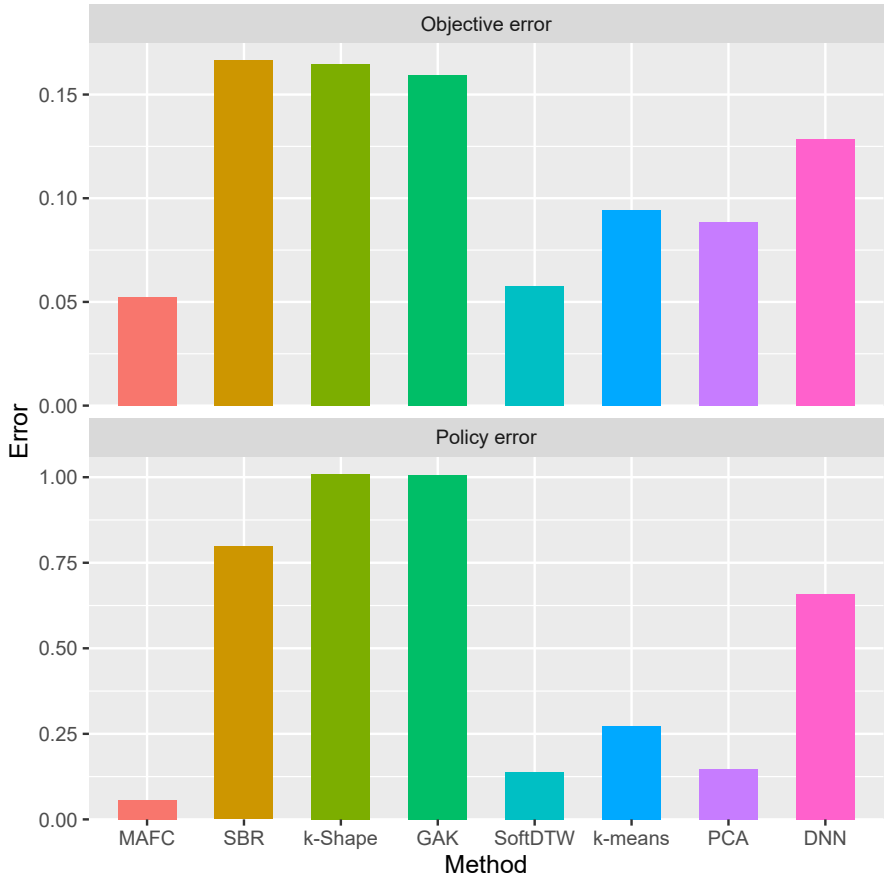


Assessment of Scenario Reduction



Case Study on a Modified IEEE 24-Bus System

- Objective error: difference between expected costs under $\boldsymbol{S}$ and under $\boldsymbol{R}$
- Policy error: obtain first-stage decisions under $\boldsymbol{S}$ and $\boldsymbol{R}$ respectively, and then evaluate the realized costs under $\boldsymbol{S}$ and calculate the difference



Outline

Conclusion

- Stochastic programming is a natural approach to short-term power system operations under high penetration of renewable generation
- To yield high quality decisions (such as day-ahead unit commitment decisions), we need to build high quality scenario sets that are representative of the underlying multivariate probability distributions
- We may also need to trim down the scenario set for the tractability of the optimization problem (which possibly involves integer variables)
- While scenario generation and scenario reduction have been extensively studied, generalization capability and feature engineering remain challenges
- In SeqGAN for scenario generation, the observation in the previous day (instead of a random noise) is used as the input, and the Monte Carlo method is employed to estimate the scores of partially generated sequences at each time step
- In MAFC for scenario reduction, a mixture of deep autoencoders map the original scenarios to latent representations which are fed into a membership deep neural network, and fuzzy clustering is employed to better preserve the patterns
- Both statistical metrics and test cases demonstrate the superior performance of the proposed deep learning approaches

References

- Felipe Beltrán, Welington de Oliveira, and Erlon Cristian Finardi, “Application of scenario tree reduction via quadratic process to medium-term hydrothermal scheduling problem,” IEEE Transactions on Power Systems, vol. 32, no. 6, pp. 4351–4361, 2017.
- Shlomo E. Chazan, Sharon Gannot, and Jacob Goldberger, “Deep clustering based on a mixture of autoencoders,” IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2019.
- Marco Cuturi and Mathieu Blondel, “Soft-DTW: A differentiable loss function for time-series,” International Conference on Machine Learning (ICML), 2017.
- Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang, “Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization,” International Conference on Computer Vision (ICCV), 2017.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative adversarial nets”, Conference on Neural Information Processing Systems (NIPS), 2014.
- Holger Heitsch and Werner Römisch, “Scenario tree modeling for multistage stochastic programs,” Mathematical Programming, vol. 118, no. 2, pp. 371–406, 2009.
- Jinxing Hu and Hongru Li, “A new clustering approach for scenario reduction in multi-stochastic variable programming,” IEEE Transactions on Power Systems, vol. 34, no. 5, pp. 3813–3825, 2019.
- Jinghua Li, Fei Lan, and Hua Wei, “A scenario optimal reduction method for wind power time series,” IEEE Transactions on Power Systems, vol. 31, no. 2, pp. 1657–1658, 2016.

References

- Junkai Liang and Wenyuan Tang, "Scenario reduction for stochastic day-ahead scheduling: A mixed autoencoder based time-series clustering approach," IEEE Transactions on Smart Grid, accepted.
- Junkai Liang and Wenyuan Tang, "Sequence generative adversarial networks for wind power scenario generation," IEEE Journal on Selected Areas in Communications, vol. 38, no. 1, pp. 110–118, 2020.
- You Lin, Ming Yang, Can Wan, Jianhui Wang, and Yonghua Song, "A multi-model combination approach for probabilistic wind power forecasting," IEEE Transactions on Sustainable Energy, vol. 10, no. 1, pp. 226–237, 2019.
- Juan M. Morales, Antonio J. Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno, *Integrating Renewables in Electricity Markets: Operational Problems*. Springer, 2014.
- Doris Sáez, Fernand Ávila, Daniel Olivares, Claudio Cañizares, and Luis Marín, "Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids," IEEE Transactions on Smart Grid, vol. 6, no. 2, pp. 548–556, 2015.
- George Sideratos and Nikos D. Hatziargyriou, "Probabilistic wind power forecasting using radial basis function neural networks," IEEE Transactions on Power Systems, vol. 27, no. 4, pp. 1788–1796, 2012.
- Yishen Wang, Yuzong Liu, and Daniel S. Kirschen, "Scenario Reduction With Submodular Optimization," IEEE Transactions on Power Systems, vol. 32, no. 3, pp. 2479–2480, 2017.

References

- Zhao Wang, Weisheng Wang, Chun Liu, Zheng Wang, and Yunhe Hou, "Probabilistic forecast for multiple wind farms based on regular vine copulas," IEEE Transactions on Power Systems, vol. 33, no. 1, pp. 578–589, 2018.
- Wei Xie, Pu Zhang, Rong Chen, and Zhi Zhou, "A nonparametric bayesian framework for short-term wind power probabilistic forecast," IEEE Transactions on Power Systems, vol. 34, no. 1, pp. 371–379, 2019.
- Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," AAAI Conference on Artificial Intelligence (AAAI), 2017.