수요예측 - part 2

2021.03.12

개요

비용과 정확도의 trade-off

- 수요예측 기법
  - 주관적 / 정성적 분석법:
    - 과거 자료가 충분치 않은 경우 주관적인 판단이나 의견에 기초하여 수행하는 예측 방법
    - 소요되는 시간과 비용이 높음. 전문가나 외부 기관으로부터의 정보등을 중요하게 사용
  - 계량적 기법:
    - 동일 제품/서비스의 과거 자료가 존재하는 경우 적용 가능. 신제품의 경우 적용 불가
    - 시계열의 경우 평균에 회귀하거나 설명변수의 변화, 외생 변수의 발생에 민감
    - 단기적인 예측에 주로 활용 가능

인과형 예측 방법

선형 모형 / ML 모형 계열

- 현업에서 필요한 예측기법은 인과형 예측기법이라고 생각됨
  - 여러가지 독립변수를 통해 시계열이 설명 될 수 있어야 하고,
  - 현업의 데이터는 보통 정상성을 만족하지 않고 (그래서 차분을 통해 정상성 확보)
  - 대내외 수많은 이벤트들이 있기 때문에, Box-Jenkins 모형은 특히 잘 맞지 않음
- 선형 모형으로는 회귀분석, 계량경제 모형 등이 있으며,
- 다양한 Machine learning 방법들도 사용됨
  - XGBoost
  - Randomforest
  - SVR

선형 모형 (Econometric approach)

- 가장 기본적인 인과형 예측기법으로 회귀분석을 사용할 수 있음

$$y_t = \beta_0 + \beta_1 x_{1t} + \ldots + \beta_p x_{pt} + \epsilon_t$$

  - 설명변수로는 이전 시점의 관측치를 넣어도 됨
  - 순환변동 요인 및 이벤트들은 회귀 분석에서 indicator 형태로 반영
  - 회귀분석 이후 오차항의 상관관계가 제대로 제거 되었는지 회귀진단을 통해서 확인 가능

선형 모형 (Econometric approach)

- 가장 기본적인 인과형 예측기법으로 회귀분석을 사용할 수 있음

$$y_t = \beta_0 + \beta_1 x_{1t} + \ldots + \beta_p x_{pt} + \epsilon_t$$

  - 다른 시계열을 설명변수로 사용할 때에는 단위근 검정을 볼 필요가 있음
    - 단위근이 존재하는 불안정적인 시계열을 그대로 사용하면 표본 수가 증가함에 따라 회귀계수의 t-값도 증가하여 상관관계가 없 는 변수간에도 매우 강한 상관관계가 있 는 것으로 나타나는 가성회귀(허구적 회귀 : spurious regression)의 문제가 발생함
    - Test 방법에는 Dicky-Fuller test가 있음

선형 모형 (Econometric approach)

NOTE: Nelson and Plosser(1982)의 연구를 기점으로 그 이후 에 발표된 실증적×이론적 연구들에 의해 다수의 경제변수들이 안정적인 시계열(stationary time series)보다는 단위근(또는 확률적 추세)을 가지는 불안정적인 시계열(non-stationary time series)에 의해 보다 잘 모형화됨이 알려지게 됨

선형 모형 (Econometric approach)

- 가장 기본적인 인과형 예측기법으로 회귀분석을 사용할 수 있음

$$y_t = \beta_0 + \beta_1 x_{1t} + \ldots + \beta_p x_{pt} + \epsilon_t$$

  - 여러 시계열이 단위근을 가지고 있다고 하더라도, 공적분 상태이면 의미있는 회귀분석이 될 수 있음
  - 공적분 검정법으로는 최근에 다변량 시계열 분석에 의한 요한 슨 공적분 검정 (Johansen’s cointegration test)이 다른 공적분 검정법보다 우월한 것으로 인정되어 널리 사용되고 있음

http://kanggc.iptime.org/grad-em/Lec4/Lec-4.pdf

http://elearning.kocw.net/contents4/document/lec/2013/Konkuk/Leegiseong/9.pdf

Markridakis Competition

- The **Makridakis Competitions** (also known as the **M Competitions** or **M-Competitions**) are a series of open competitions to evaluate and compare the accuracy of different time series forecasting methods. They are organized by teams led by forecasting researcher Spyros Makridakis and were first held in 1982.

http://kanggc.iptime.org/grad-em/Lec4/Lec-4.pdf

http://elearning.kocw.net/contents4/document/lec/2013/Konkuk/Leegiseong/9.pdf

M4 - 2018

- The M4 extended and replicated the results of the previous three competitions, using an extended and diverse set of time series to identify the most accurate forecasting method(s) for different types of predictions.

- It aimed to get answers on how to improve forecasting accuracy and identify the most appropriate methods for each case. To get precise and compelling answers, the M4 Competition utilized 100,000 real-life series, and incorporates all major forecasting methods, including those based on Artificial Intelligence (Machine Learning, ML), as well as traditional statistical ones.

http://kanggc.iptime.org/grad-em/Lec4/Lec-4.pdf

http://elearning.kocw.net/contents4/document/lec/2013/Konkuk/Leegiseong/9.pdf

5 Major findings from M4

1. The combination of methods was the king of the M4. Out of the 17 most accurate methods, 12 were "combinations" of mostly statistical approaches.
2. The biggest surprise, however, was a "hybrid" approach utilizing both Statistical and ML features. This method, produced the most accurate forecasts as well as the most precise PIs and was submitted by Slawek Smyl, Data Scientist at Uber Technologies. According to sMAPE, it was close to 10% (a huge improvement) more accurate than the Combination (Comb) benchmark of the Competition (see below). It is noted that in the M3 Competition (Makridakis & Hibon, 2000) the best method was 4% more accurate than the same Combination.
3. The second most accurate method was a combination of seven statistical methods and one ML one, with the weights for the averaging being calculated by a ML algorithm, trained to minimize forecasting error through holdout tests. This method was jointly submitted by Spain's University of A Coruña and Australia's Monash University.
4. The first and the second most accurate methods also achieved an amazing success in specifying correctly the 95% PIs. These are the first methods we know that have done so and do not underestimate uncertainty considerably.
5. The six pure ML methods submitted in the M4 performed poorly, none of them being more accurate than Comb and only one being more accurate than Naïve2. These results are in agreement with those of a recent study we published in PLOS ONE (Makridakis, et al., 2018).[18]

https://en.wikipedia.org/wiki/Makridakis_Competitions

M5 - 2020

M5 commenced on March 3, and the results were declared on July 1, 2020. It used real-life data from Walmart and was conducted on Kaggle's Platform. It offered substantial prizes totaling US$100,000 to the winners. The data was provided by Walmart and consisted of around 42,000 hierarchical daily time series, starting at the level of SKUs and ending with the total demand of some large geographical area. In addition to the sales data, there was also information about prices, advertising/promotional activity and inventory levels as well as the day of the week the data refe

https://en.wikipedia.org/wiki/Makridakis_Competitions

Findings from M5

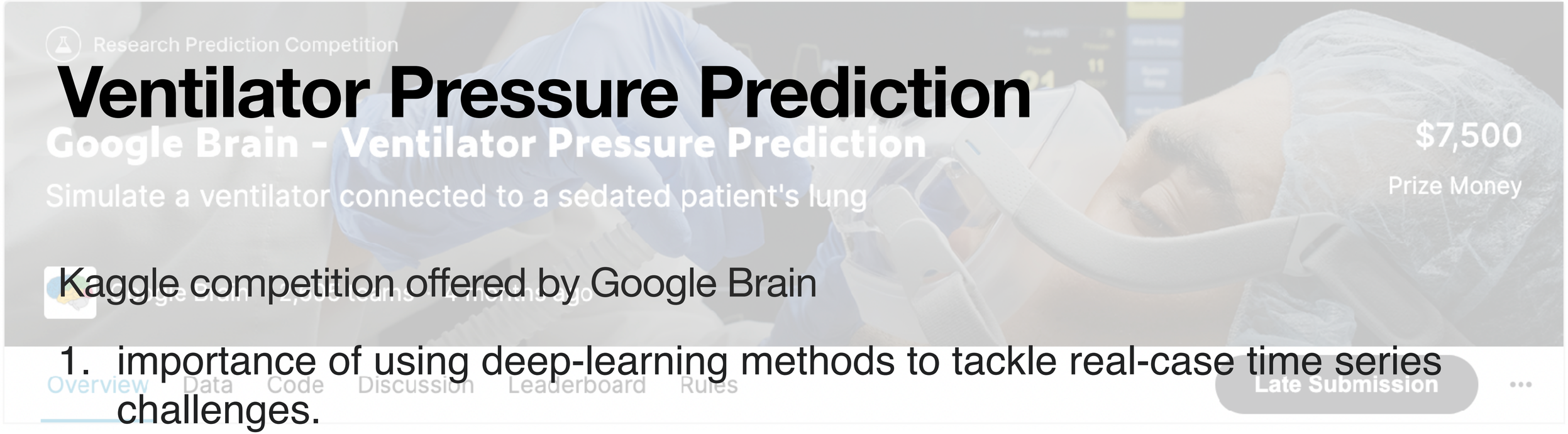
1. With a more creative dataset, the top spot submissions featured only ‘ML’ methods.
2. To be more precise, all 50 top-performing methods were ML-based.
3. This competition saw the rise of the versatile ***LightGBM*** (used for time series forecasting) and the debut of **Amazon’s *DeepAR*** and ***N-BEATS***
4. The *N-BEATS* model, published in 2020, outperformed the winner of the M4 competition by 3%!

https://en.wikipedia.org/wiki/Makridakis_Competitions

Timeseries도 ML의 물결이..

- 최근 2년새 ML의 약진이 두드러짐
- lightGBM이 시계열 예측에서 좋은 성능을 보임
- N-BEATS와 DeepAR의 등장은 최근 2년간 가장 큰 진보라고 생각할 수 있음
- 이후 각종 형태의 self-attention을 활용한 Transformer의 적용이 이루어짐

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

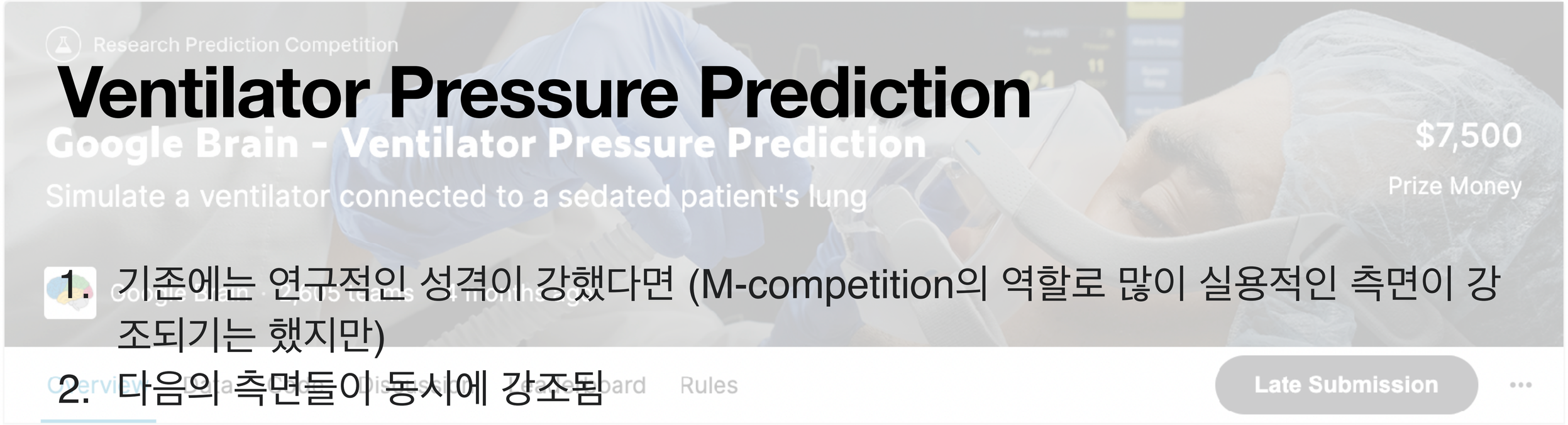


Ventilator Pressure Prediction

Kaggle competition offered by Google Brain

1. importance of using deep-learning methods to tackle real-case time series challenges.
2. Specifically, the goal of the competition was to predict the time sequence of pressure within a mechanical lung, given the time series of control inputs.
3. Each training instance was essentially a time series of its own, thus the task was a multiple time series problem.
4. The winning team submitted a multi-level deep architecture, which included, among others, an LSTM network and a Transformer block.

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0



1. 기존에는 연구적인 성격이 강했다면 (M-competition의 역할로 많이 실용적인 측면이 강조되기는 했지만)
2. 다음의 측면들이 동시에 강조됨

- **Versatility:** The ability to use the model for different tasks.
- **MLOps**: The ability to use the model in production.
- **Interpretability and explainability:** Black-box models are not so popular anymore.

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

ML 모형

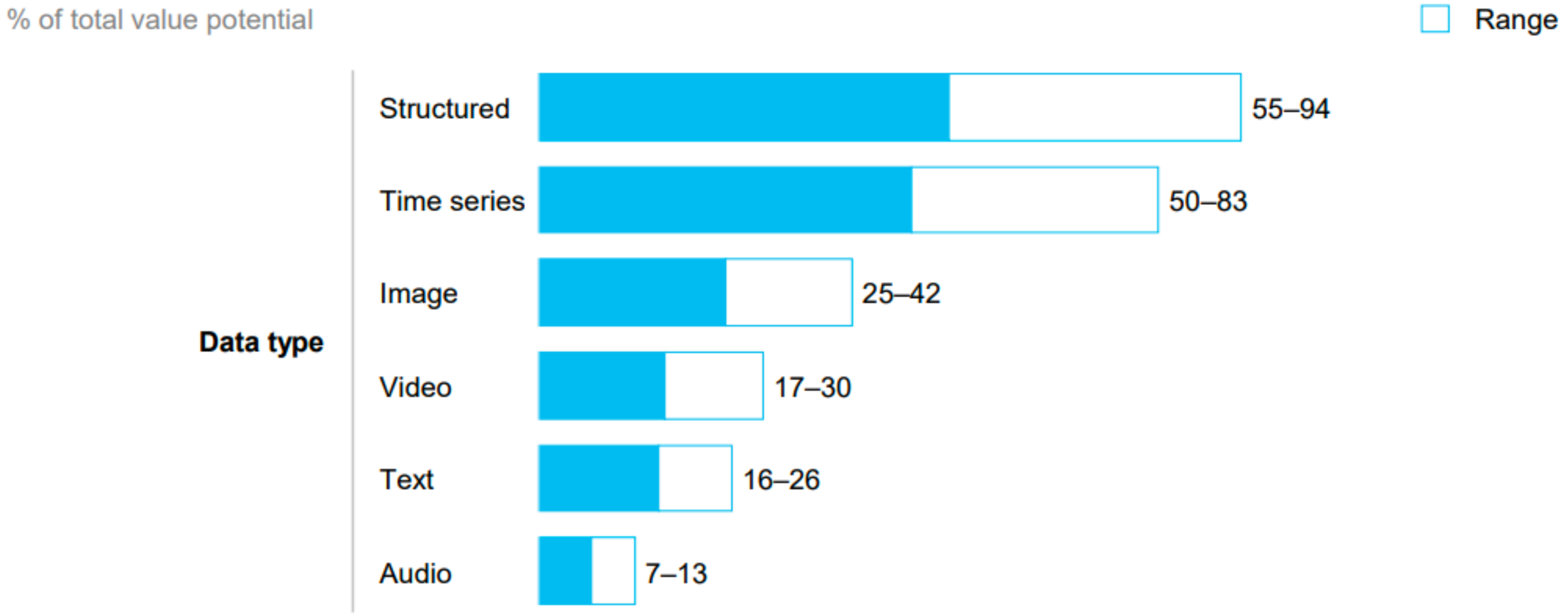
- 결국 선형 모형도 여러가지 오차항의 가정 내에서 Loss를 최소화하는 방식으로 진행하므로, 머신러닝 기법들 역시 적용 가능
- 선형 모형이 관측치와 독립변수의 선형 결합으로 모형을 진행한다면,
- 머신러닝 모형들은 훨씬 많은 interaction을 고려한 비선형 모형으로 볼 수 있음
- SVM (SVR이 시계열 모형에는 더욱 적합)도 있겠지만, 가장 대표적인 방법이 Boosting과 Bagging을 바탕으로 한 ensemble 기법들임

http://kanggc.iptime.org/grad-em/Lec4/Lec-4.pdf

http://elearning.kocw.net/contents4/document/lec/2013/Konkuk/Leegiseong/9.pdf

ML 모형

- ML에서 시계열 예측은 중요한 영역



https://medium.com/daria-blog/machine-learning-for-time-series-forecasting-part-1-6e97661c9773

ML 모형 - Tree 기반 Ensemble

- 약한 여러개의 classifier (regressor)를 여러개 사용하여 예측의 성능을 높임
- Boosting & Bagging으로 크게 나뉨

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- $\mathcal{F}$은 모든 CART 모형의 모임
- 목적함수는 종속변수에 대한 likelihood와 판별기 f_k에 적용하는 regularization으로 구성

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

ML 모형 - Bagging

- 배깅(Bagging; Bootstrap Aggregating)
- 학습 데이터에서 다수의 부트스트랩 자료를 생성하고, 각 자료를 모델링한 후 결합하여 최종 예측 모형 구축
- 부트스트랩(Bootstrap) : 주어진 자료에서 단순 랜덤 복원추출 방법을 활용하여 동일한 크기의 표본을 여러 개 생성하는 샘플링 방법
- 실제 현실에서는 학습 자료의 모집단 분포를 알기가 어려움. 배깅 기법으로 뽑아낸 훈련자료를 모집단으로 생각하고, 평균 예측 모형을 구하여 분산을 줄이고 예측력을 향상
- CART와 같은 decision tree와는 다르게, 개별 tree는 가지치기를 하지 않음으로 해서, overfitting 시킴
- 개별 tree를 서로 다르게 함으로써, 예측치와 예측 에러의 독립성을 확보

인과형 예측기법

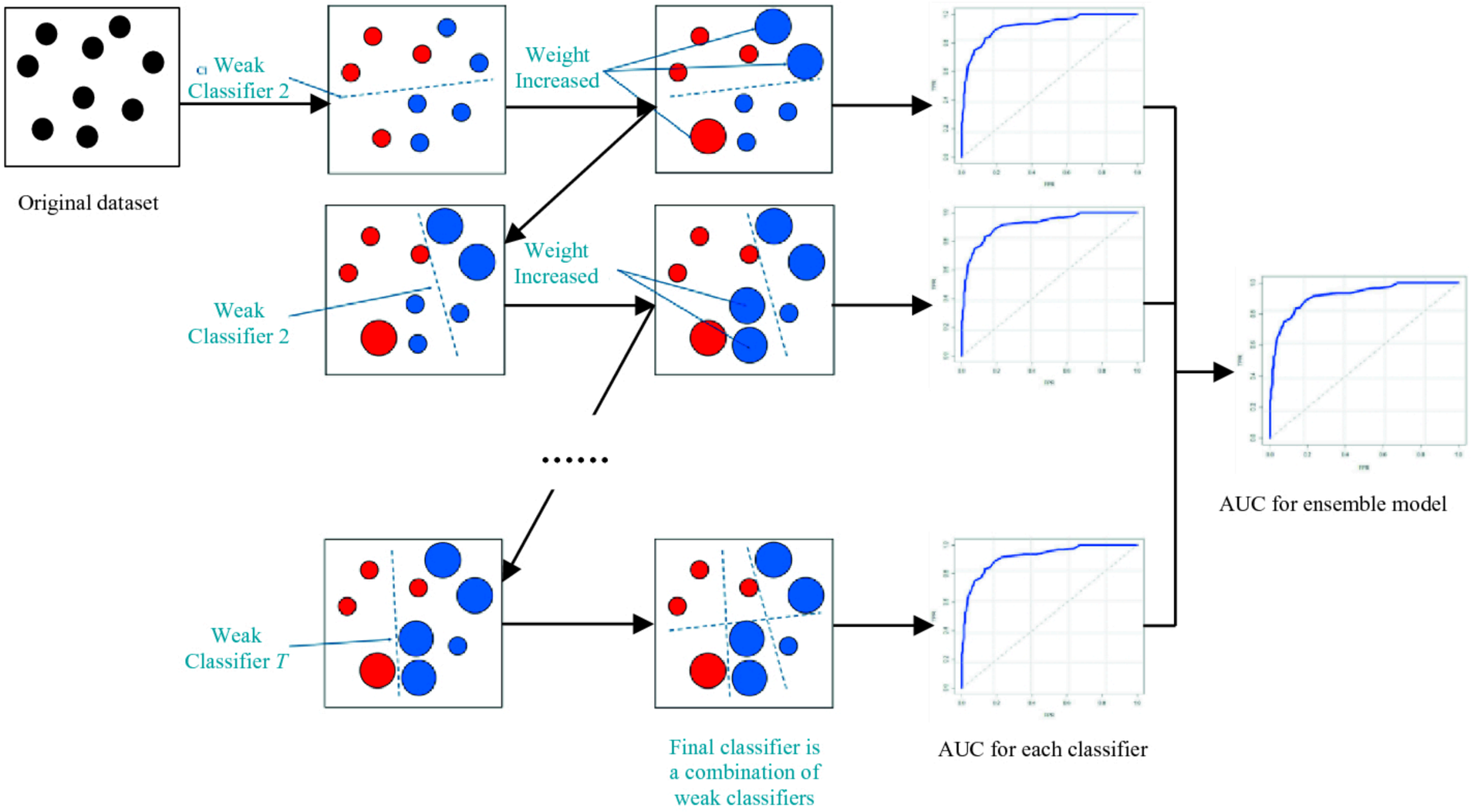
ML 모형 - Boosting

- 잘못 분류된 개체들에 가중치를 적용, 새로운 분류 규칙을 만들고, 이 과정을 반복해 최종 모형을 구축
- 가중치 적용 : 잘못 분류된 데이터를 다음 바구니에 넣는 것
- 잘 분류된 데이터의 가중치를 낮춤으로써 다음 바구니를 구성하여 강한 예측 모형을 만드는 방법
- 부스팅 기법의 특징 : 분류하기 힘든 관측값들에 대해서 정확하게 분류를 잘하도록 유도
- 최적 모델 결정 방법 : 이전 분류에서 정 분류 데이터에는 낮은 가중치 부여 / 이전 분류에서 오 분류 데이터에는 높은 가중치 부여
- 장점 : 특정 케이스의 경우 상당히 높은 성능을 보임 / 일반적으로 과대 적합이 없음

Gradient Boosting

https://datascience.eu/machine-learning/gradient-boosting-what-you-need-to-know/

Gradient Boosting



Gradient Boosting

- "Gradient Boosting”: [Greedy Function Approximation: A Gradient Boosting Machine, by Friedman] 논문에서 처음 나왔음

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i-1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i-1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

XGBoost

XG-Boost의 특성

- [XGBoost: A Scalable Tree Boosting System] -> "Extreme Gradient Boosting"을 의미
- XGBoost는 Gradient Boosting 방법 중 한 가지
- 효율성과 유연성, 휴대성(efficient, flexible, portable)이 뛰어나도록 최적화된 distributed gradient boosting 라이브러리로도 제공
- XGBoost는 많은 데이터 과학 문제를 빠르고 정확하게 해결하는 parallel tree boosting(GBDT, GBM이라고도 함)을 제공
- 동일한 코드가 주요 분산 환경(하둡, SGE, MPI)에서 실행되며 수십억 가지의 문제를 해결할 수 있다.

XG-Boost (장점)

- [Over fitting(과적합)을 방지할 수 있다.
- 신경망에 비해 시각화가 쉽고, 이해하기보다 직관적이다.
- 자원(CPU, 메모리)이 많으면 많을수록 빠르게 학습하고 예측할 수 있다.
- Cross validation을 지원한다.
- 높은 성능을 나타낸다. 실제로 Kaggle에서 XGBoost를 쓴 결과들이 상위권을 쓸어 담기 시작하면서 사람들이 너도 나도 사용하기 시작했다.

XG-Boost 모형 상세

- 데이터 셋 $m \times n$이 주어져 있다고 가정하면, 즉,

$$\mathscr{D} = \{(x_i, y_i) \,|\, |\mathscr{D}| = n, x_i \in \mathbb{R}^n, y_i \in \mathbb{R}\}$$

- Tree Boost 기법에서는 다음과 같은 구조의 모형을 통해 예측을 진행

$$\hat{y} = \phi(x) = \sum_{k=1}^{K} f_k(x), \quad f \in \mathscr{F}$$

$$\mathscr{F} = \{f(x) = w_{q(x)}, q : \mathbb{R}^m \to \{1, \ldots, T\}, w \in \mathbb{R}^T\}$$

XG-Boost 모형 상세

- Model Complexity
  - Gradient boosting에 비해 regularization term이 추가됨
  - 결국 Tree 모형에서 classifier는 다음과 같은 vector로 정의할 수 있음

$$f_t(x) = w_{q(x)}, \quad w \in R^T, \quad q : R^d \rightarrow 1,2,\ldots,T$$

T: t번째 tree의 leaf node 갯수

w:는 Leaf에서의 score가 저장되어 있는 T차원 vector

q: 개별 데이터 포인트를 T개의 leaf node로 나누는 tree classifier

- 위와 같이 classifier를 정의하면 다음과 같이 regularization term을 표현할 수 있음

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

https://kicarussays.tistory.com/28

XG-Boost 모형 상세

- 목적함수는 다음과 같다

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \quad \Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{i=1}^{T} w_i^2$$

- 모형이 작을 수록, 복잡도가 낮을 수록 penalty를 적게 받게 됨
- Regularization term이 없으면 gradient boosting과 같아짐

https://kicarussays.tistory.com/28

XG-Boost 모형 상세

- 목적함수를 최소화 하는 K개의 tree를 찾는 것은 현실적으로 불가능
  - NP Complete 문제로 알려져 있음
- XG-Boost에서는 매 iteration마다 순차적으로 Tree의 가지를 추가해 가는 Additive training 전략을 사용

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

https://kicarussays.tistory.com/28

XG-Boost 모형 상세

- Objective function

$$
\begin{aligned}
L^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{K} \Omega(f_k) \\
&\propto \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \\
&\propto \sum_{i=1}^{n} \left[l(y_i, y_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)
\end{aligned}
$$

(Taylor Expansion, 2nd order approx.)

Where, $g_i = \dfrac{\partial}{\partial \hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right), \qquad h_i = \dfrac{\partial^2}{\partial \hat{y}_i^{(t-1)^2}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$

- Objective function은 오로지 g_i, h_i에만 의존함: XGBoost가 custom loss function을 지원하는 이유임 (Gradient boosting에서는 지원하지 않음)

https://kicarussays.tistory.com/28

XG-Boost 모형 상세

- XG Boost에서는 regularization part에 주목해서 식을 전개
  - 이전의 tree 모형에서는 그렇게 중요하게 생각하지 않았던 요소.
  - Impurity를 개선하기 위한 알고리즘 개발에 주력하여서, 상대적으로 그 중요도가 덜 강조되었고,
  - Model complexity는 stopping rule 등을 사용한 pruning을 통해서 주로 해소 => Heuristic 접근법
  - Regularization term을 명확하게 정의하면, 모형에 대한 이해를 높일 수 있고, 결국 보다 성능 좋은 모형을 찾아낼 수 있는 방법임

https://kicarussays.tistory.com/28

Loss + Complexity

- 여기에서는 y_i를 T 차원의 one-hot vector로 간주

$$obj^{(t)} \approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda\sum_{i=1}^{T} w_i^2$$

($g_i w_{q(x_i)}$와 $h_i w_{q(x_i)}^2$ 모두 행렬 곱에 의한 scalar)

$$= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j+\lambda)w_j^2\right] + \gamma T$$

(관측치 관점이 아니라 Leaf node 관점에서의 sum)

여기서. $H_j = \sum_{i\in I_j} h_i, \quad G_j = \sum_{i\in I_j} g_i, \quad I_j = \{i \mid q(x_i) = j\}$

- j번째 node에 속하는 관측치의 결과값 w_j는 모두 동일함

https://kicarussays.tistory.com/28

Derived Goodness of Fit measure

- 앞에서 도출한 목적함수를 최소화 하는 w_j^*는 다음과 같이 표현할 수 있음

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

결국 목적함수의 최소값은

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j}{H_j + \lambda} + \gamma T$$

  - Tree structure를 나타내는 $q(x)$의 우월성을 측정하는 measure 역할

Information gain

- 목표는 information gain을 극대화하는 split point를 찾는 것임
- 순차적으로 Tree를 탐색해 나갈 때, 다음의 information gain을 바탕으로 탐색

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

만약 Gain이 γ보다 작으면, 더이상 가지치기를 하지 않음: 기존의 pruning에 해당

https://kicarussays.tistory.com/28

Parameter

- 일반 parameter: Booster type, n_jobs, verbosity
- Booster parameter: Learning_rate, n_estimators, max_depth, min_child_weight, gamma, lambda, alpha, subsample, colsample_bytree
- Learning Parameter: objective, eval_metric, seed

https://wooono.tistory.com/97

Parameter

- 일반 parameter: Booster type, n_jobs, verbosity
- Booster parameter: Learning_rate, n_estimators, max_depth, min_child_weight, gamma, lambda, alpha, subsample, colsample_bytree
- Learning Parameter: objective, eval_metric, seed

https://wooono.tistory.com/97

일반 Parameter

Parameter	Option	초기값	의미
Booster Type	gbtree	O	높을 수록 과적합하기 쉬움
	gblinear		
	dart		커질 수록 모델의 복잡도가 커짐
n_jobs	1		병렬작업의 개수
Verbosity	0		로그 없음
	1	O	경고 출력
	2		정보 출력
	3		Debug용 error 출력
Valid_parameters	FALSE		Parameter간의 관계 검증
disable_default_eval_metric	0	O	Custom metric 사용시 설정
	1		
num_pbuff			마지막 부스팅 단계에서 결과 저장 공간 크기, 자동설정
num_feature			사용할 feature의 갯수, 자동설정

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Booster Parameter

Parameter	의미	초기값	추천값
Learning rate	Learning rate	0.3	높을 수록 과적합하기 쉬움
n_estimators	Weak learner의 갯수	100	
max_depth	개별 tree의 max depth	6	커질 수록 모델의 복잡도가 커짐
min_child_weight	관측치에 대한 가중치 합의 최소	1	높을 수록 과적합 방지
gamma	Leaf node의 추가분할을 결정할 최소손실값	0	높을 수록 과적합 방지
subsample	Weak learner가 학습에 사용하는 데이터 샘플링 비율	1	낮을 수록 과적합 방지
subsample_method	데이터 샘플링 방법 (uniform, gradient_based)	Uniform	
colsmaple_bytree	각 step tree별로 사용된 feature의 %	1	낮을 수록 과적합 방지
colsample_bylevel	각 tree의 depth level별로 사용될 feature %	1	
colsample_bynode	각 node별로 사용될 feature %	1	
lambda (reg_lambda)	가중치에 대한 L2 regularization 값	1	클수록 과적합 감소
alpha (reg_alpha)	L1 regularization 적용값	0	클수록 과적합 감소
scale_pos_weight	Data의 label 불균형 정도를 알려줌	1	Imbalance 정도를 입력

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Learning Parameter

Parameter	Option	초기값	비고
base_score		0.5	초기편향값
Objective	reg: squared error	O	높을 수록 과적합하기 쉬움
	Binary: logistic error		
	Multi:softmax		num_class를 지정해야 함
	Multi:softprob		
	Count: Poisson		
seed	0		결과 재현을 위한 seed
eval_metric	rmse		
	mae	O	
	logloss		
	Error		
	mlogloss		
	auc		ROC auc를 의미. FPR과 TPR에 민감한 경우 사용
	aucpr		Pre/Recall auc를 의미. f-score나 recall에 민감할 때 사용
	map		
number_boost_round			몇회의 Step을 반복할지 결정
early_stopping_rounds			early_stopping_rounds 동안 metric이 개선되지 않을 때 조기종료

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

LightGBM

LightGBM

- GBM은 Decision Tree를 만드는 과정에 Gradient Boosting을 활용하는 것
- 트리가 가지를 늘려가는 과정에서 Gradient(MSE loss에서는 단순히 residual)를 계산하여 손실함수를 가장 큰 폭으로 줄일 수 있는 부분에서 leaf를 분리
- 따른 말로는, 매 단계마다 이전 단계에서 잘 학습하지 못했던 데이터를 보다 집중적으로 학습하는 트리를 만든다는 것
- GBDT를 만드는 과정 중에서 가장 많은 시간과 자원이 소요되는 부분은 ***split* 지점을 찾는 부분**
  - 해당 Feature의 모든 값을 정렬하고, 가능한 모든 split 지점 중에서 어떤 부분이 가장 잘 데이터를 분류하는지 찾아야 하기 때문
- 이렇게 모든 split 지점을 확인하는 것은 매우 비효율적이기 때문에, 효율적으로 GBDT를 구성하기 위해 많은 방법들이 제시되고 있습니다 (e.g. XGBoost, LightGBM, CatBoost).

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Gradient-based One-Side Sampling (GOSS)

- 기본 가정은 instance의 gradient가 클수록 영향력이 크고 덜 학습된 instance라는 것
- instance의 gradient가 크다는 것은, 해당 instance를 구성하는 값(element)들의 변화가 손실함수에 큰 변화를 가져올 수 있다는 것
- LightGBM에서는 이런 instance를 **모델이 충분히 학습하지 못한 *instance*** 라고 보고 있고, 따라서 이러한 instance를 포함하면서 데이터셋의 샘플 수를 줄이려고 시도

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models ← { }, fact ← $\frac{1-a}{b}$
topN ← a × len(I) , randN ← b × len(I)
for $i = 1$ ***to*** d **do**
  preds ← models.predict(I)
  g ← $loss$(I, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Information gain, variance gain

Definition 3.1 *Let O be the training dataset on a fixed node of the decision tree. The variance gain of splitting feature j at point d for this node is defined as*

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in O : x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in O : x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right),$$

where $n_O = \sum I[x_i \in O]$, $n^j_{l|O}(d) = \sum I[x_i \in O : x_{ij} \le d]$ *and* $n^j_{r|O}(d) = \sum I[x_i \in O : x_{ij} > d]$.

- Split을 통해 Variance gain이 얼마나 줄어들었는지를 기준으로 split의 적정성을 평가
- 가장 적절한 split은
- $d_j^* = \arg\max_d V_j(d)$

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

EFB, Exclusive Feature Bundling

- EFB는 **데이터셋의 *Feature* 수를 줄이는 알고리즘**
- 고차원의 데이터는 sparse하다는 가정 하에, 상호배타적인 변수들을 하나의 bucket으로 묶어서 Feature 수를 줄이는 방식
  - "상호배타적인 변수들"은 동시에 0이 아닌 값을 갖지 않는 변수
- 상호배타적인 변수들을 묶어서 최소의 bundle 로 만드는 문제는 NP-complete 문제
  - Graph 색칠 문제와 같은 문제 -> 다항시간 내 해를 찾을 수 없음
- 따라서, Greedy 알고리즘을 활용해야한다고 주장

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Merge Exclusive Features

- bundle 내에서 기준 변수 설정
- 모든 instance에 대해서, conflict 한 경우는 기준 변수 값 그대로 가져감
- 모든 instance에 대해서, conflict가 아닌 (상호 배타적인) 경우 기준 변수의 최대값 + 다른 변수 값을 취함

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges $\leftarrow \{0\}$, totalBin $\leftarrow 0$
for f ***in*** F **do**
    totalBin += f.numBin
    binRanges.append(totalBin)
newBin $\leftarrow$ new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
    newBin[i] $\leftarrow 0$
    for $j = 1$ ***to*** $len(F)$ **do**
        if $F[j].bin[i] \neq 0$ **then**
            newBin[i] $\leftarrow$ F[j].bin[i] + binRanges[j]
Output: $newBin$, $binRanges$

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

특장점

빠른 속도

Table 2: Overall training time cost comparison. LightGBM is lgb_baseline with GOSS and EFB. EFB_only is lgb_baseline with EFB. The values in the table are the average time cost (seconds) for training one iteration.

	xgb_exa	xgb_his	lgb_baseline	EFB_only	**LightGBM**
Allstate	10.85	2.63	6.07	0.71	**0.28**
Flight Delay	5.94	1.05	1.39	0.27	**0.22**
LETOR	5.55	0.63	0.49	0.46	**0.31**
KDD10	108.27	OOM	39.85	6.33	**2.85**
KDD12	191.99	OOM	168.26	20.23	**12.67**

Table 3: Overall accuracy comparison on test datasets. Use AUC for classification task and NDCG@10 for ranking task. SGB is lgb_baseline with Stochastic Gradient Boosting, and its sampling ratio is the same as LightGBM.

	xgb_exa	xgb_his	lgb_baseline	SGB	**LightGBM**
Allstate	0.6070	0.6089	0.6093	0.6064 ± 7e-4	**0.6093 ± 9e-5**
Flight Delay	0.7601	0.7840	0.7847	0.7780 ± 8e-4	**0.7846 ± 4e-5**
LETOR	0.4977	0.4982	0.5277	0.5239 ± 6e-4	**0.5275 ± 5e-4**
KDD10	0.7796	OOM	0.78735	0.7759 ± 3e-4	**0.78732 ± 1e-4**
KDD12	0.7029	OOM	0.7049	0.6989 ± 8e-4	**0.7051 ± 5e-5**

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

특장점

빠른 수렴 속도

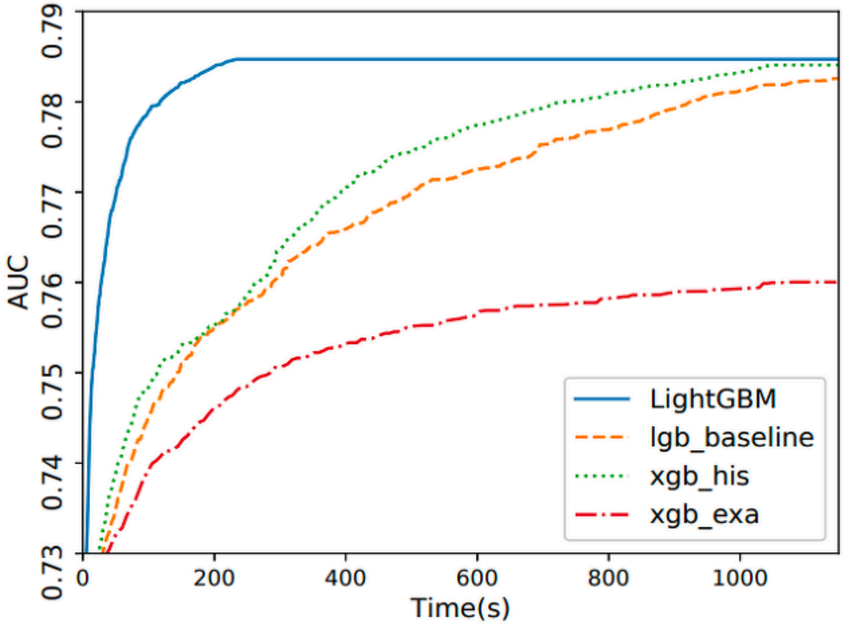


Figure 1: Time-AUC curve on Flight Delay.

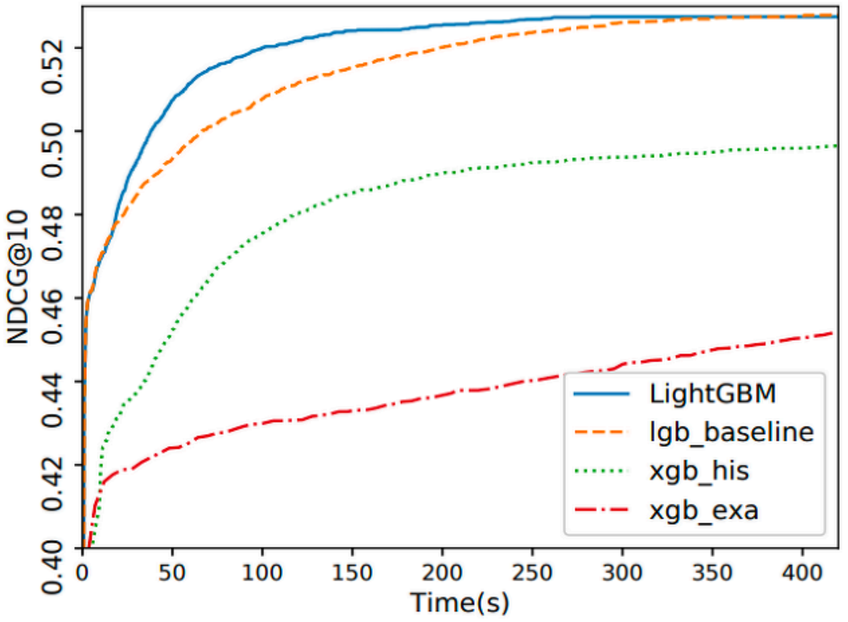


Figure 2: Time-NDCG curve on LETOR.

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

CatBoost

CatBoost (Categorical Boosting) 개요

- CatBoost는 **범주형*(categorical)* 데이터를 처리하는 새로운 방법** 을 제시하고
- 기존의 그래디언트 부스팅 알고리즘을 조작하여 **타겟 누수*(target leakage)*를 개선**
- target leakage는 예측 시점에서 사용할 수 없는 데이터가 데이터셋에 포함 되는 오류
- 모델이 독립변수들인 x만을 활용하여 종속변수인 y를 예측해야 하는데, y에 대한 정보가 x에 포함되어 있는 것이 타겟누수

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

변수 encoding - Target Statistic

- 범주형 데이터는 숫자형 (numeric) 데이터로 바꾸어야 분석이 가능
- 대표적인 방법이 one-hot encoding이나 변수의 범주가 너무 많다면 데이터셋의 크기가 커지게 됨
- 이를 방지하기 위해 one-hot encoding 수행 전에, 범주를 군집화 (clustering) 하여 그 수를 줄이거나, 새로운 숫자형 데이터로 변환할 수 있음
- Catboost에서는 ***target statistics*** 방식을 제안

$$\hat{x}_k^i = \mathbb{E}(y_i \,|\, x_i = x_k^i)$$

- 4개의 target statistics를 구하는 방식이 소개됨
  - Greedy-TS, holdout-TS, One-Leave-Out TS, Ordered TS

https://dining-developer.tistory.com/7?category=929228 https://wooono.tistory.com/97

Random Forest

ML 모형 - Random Forest 개요

- 랜덤 포레스트(Random Forest)는 배깅과 부스팅보다 더 많은 무작위성*을 주어 약한 학습기들을 생성한 후 이를 선형 결합하여 최종 학습기를 만드는 방법
- 배깅을 하되, 개별 tree에서 사용하는 feature를 subset을 활용함으로써, tree의 다양성을 확보하고, 동시에 예측치의 독립성을 확보
- 무작위성(Randomness) : 사건에 패턴이나 예측 가능성이 없는 것, 인위적인 요소가 없는 성질
- 랜덤 포레스트는 정확도와 예측력 측면에서 좋은 성과를 보여줍니다.
- 하지만, 이론적 설명이나 최종 결과에 대한 해석이 어렵다는 단점이 있습니다.

https://blog.naver.com/joseb1234/222240682828

Parameter

포레스트 크기	- 총 포레스트를 몇 개의 트리로 구성할지를 결정하는 매개변수 - 포레스트가 작을 때 : Overfitting - 포레스트가 클 때 : Generalization이 우수
최대 허용 깊이	- 하나의 트리에서 루트 노드부터 종단 노드까지 최대 몇 개의 노드 결정 - 최대 허용 깊이가 작으면 과소 적합(Under-fitting), - 깊으면 과대 적합이 일어나기 때문에 적절한 값 설정 필요
임의성 정도	- 임의성 정도에 따라 비상관화 수준 결정

https://blog.naver.com/joseb1234/222240682828

Ensemble를 이용한 시계열 분석

- 시계열 데이터 예측(Time series forecasting)은 지도 학습(Supervised learning)으로 변형시킬 수 있다.
  - 기본적으로는 ML 기법들은 시점에 대해 extrapolation이 되지 않는다.
- 변환 데이터로 랜덤포레스트를 수행하기 위해서는 벡터값을 반드시 매트릭스로 변환해야 함
- 이러한 개념을 시간 지연 임베딩(time delay embedding)이라고 부릅니다.
- 시간 지연 임베딩은 K개의 차원을 유클리드 공간의 시계열에 데이터를 임베딩하는 과정을 말합니다.
- 공정한 성능 측정을 위해 walk-forward validation을 고려해야 함

https://medium.com/daria-blog/machine-learning-for-time-series-forecasting-part-1-6e97661c9773

Deep Learning 계열

- LSTM, 1D Conv LSTM, CNN LSTM
- ANN, Restricted Bolzman machine
- DeepSD
- Extreme Learning Machine
- deep multi-view spatial-temporal network (DMVST-Net)
- DeepAR
- Gaussian Process Regression
- MCGM(Markov Chain Grey-Markov Model

Deep Learning 계열

- 시계열 모형은 어떤 변수의 현재값과 과거값 간에 안정적인 함수관계가 존재한다는 가정
- 만일 이런 안정적인 함수관계가 존재한다면, ”Universal approximator"의 성질을 지니는 ANN을 활용 가능
- 기계학습 방법론은 다변량 입력, 복잡한 비선형 관계, 결측치 존재라는 불리한 상황에서도 시계열 예측을 효과적으로 수행할 수 있음
- 시계열 예측에 주로 사용되는 딥러닝 모형에는 DNN(Deep Neural Networks), RNN(Recurrent Neural Networks), CNN(Convolutional Neural Networks) 등이 있다.

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

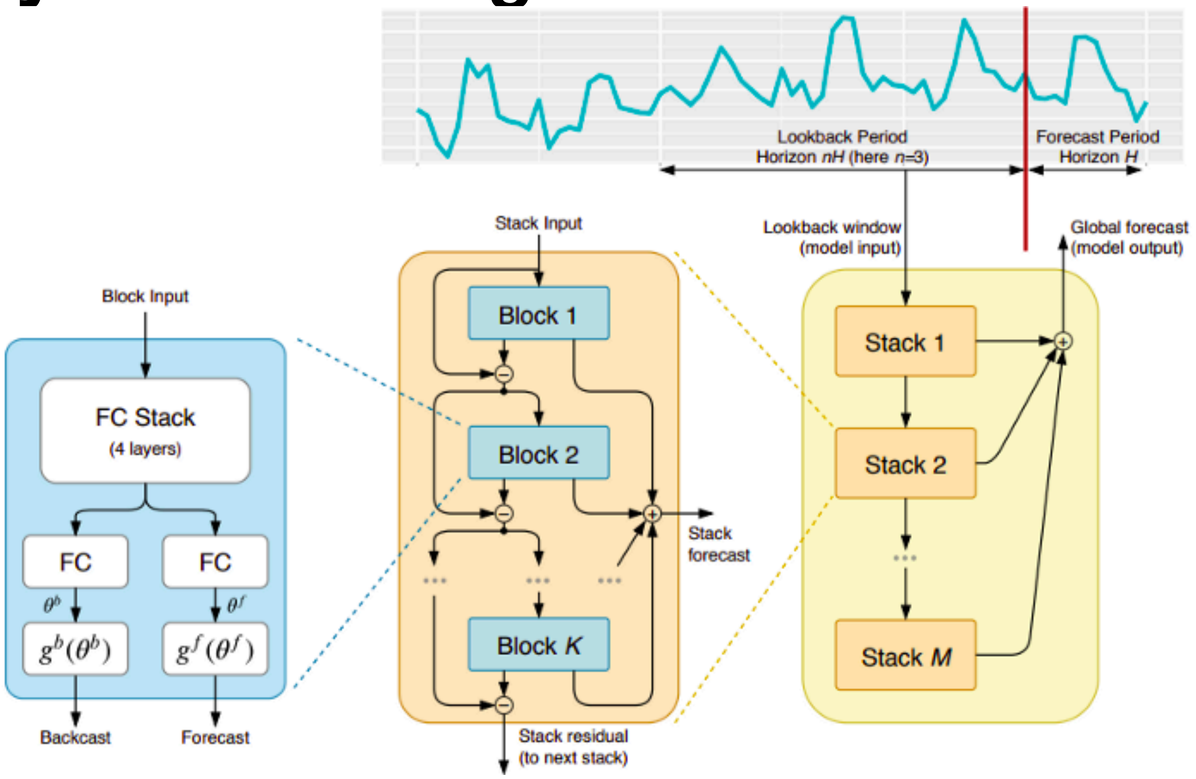
Pure Deeplearning Architecture

N-BEAT

Proposed by ElementAI, founded by Yoshua Bengio

- Each successive block models only the residual error due to the reconstruction of the backcast from the previous block and then updates the forecast based on that error. This process mimics the Box-Jenkins method when fitting ARIMA models.

Note: The original *N-BEATS* implementation only works with univariate time series.

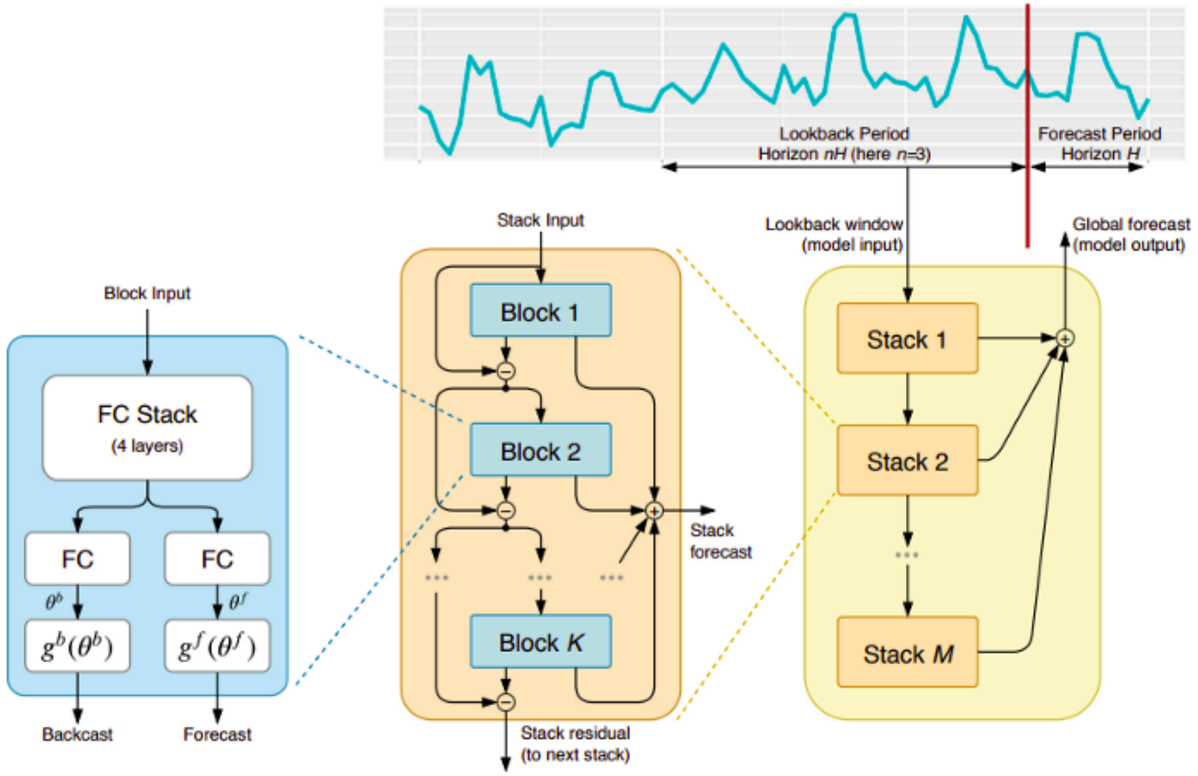


https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

N-BEAT

특장점

- **Expressive and easy to use:**
  - The model is simple to understand and has a modular structure (blocks and stacks).
  - Moreover, it is designed to require minimal time-series feature engineering and no input scaling

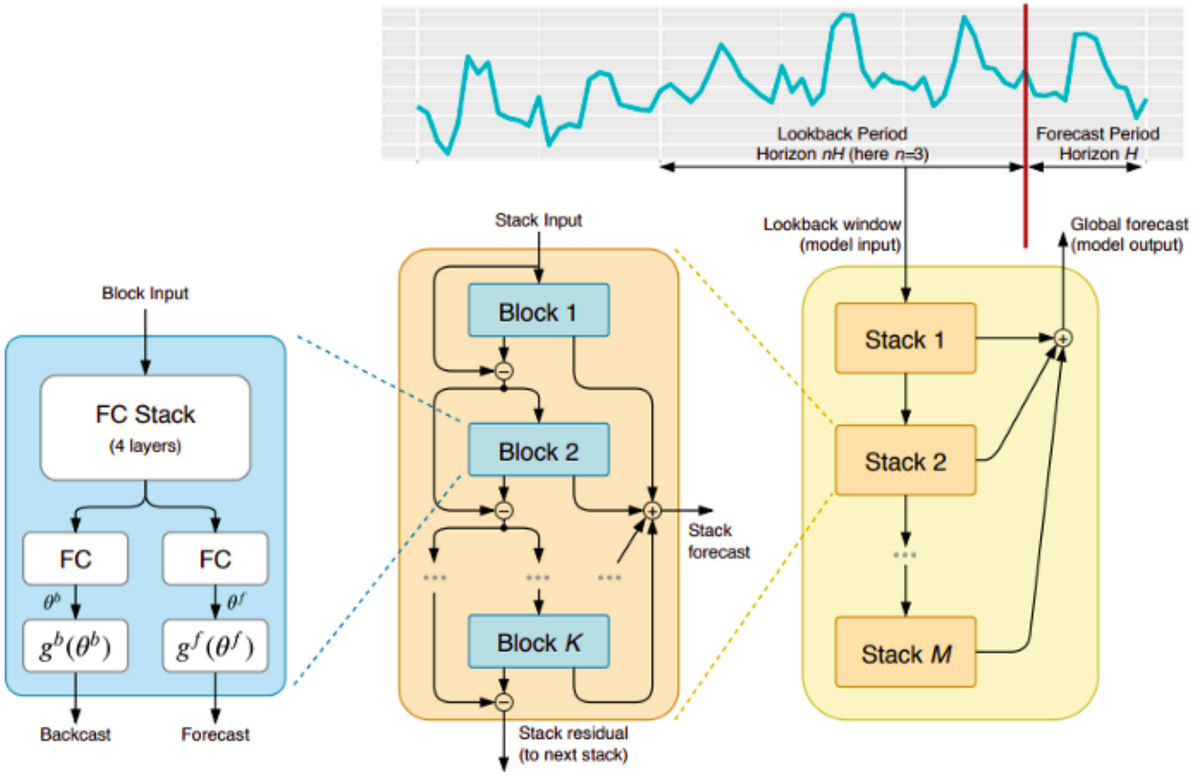


https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

N-BEAT

특장점

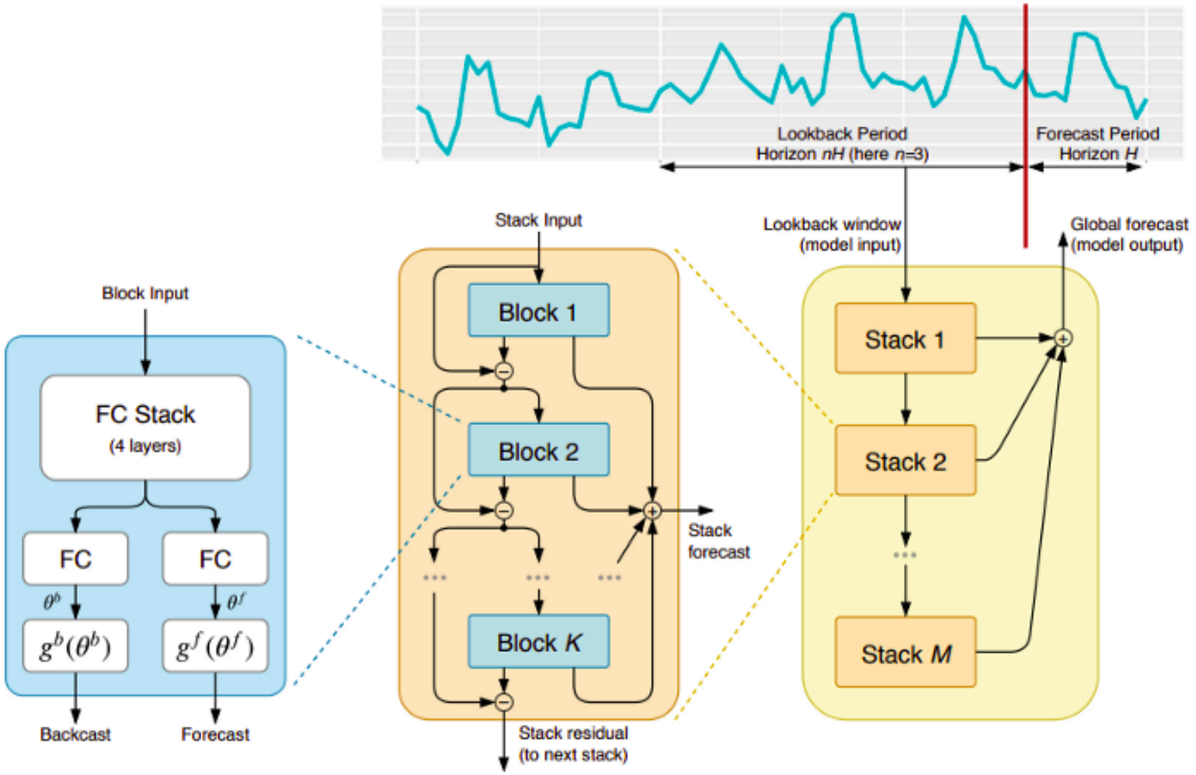
- **Multiple time-series:**
  - The model has the ability to generalize on many time-series. In other words, different time series with slightly different distributions can be used as input.
  - In the *N-BEATS* implementation, this is achieved through *meta-learning*: the inner and the outer learning procedures.
  - The inner learning procedure takes place inside blocks and helps the model capture local temporal characteristics.
  - On the other hand, the outer learning procedure takes place inside stacks and helps the model learn global characteristics across all time-series.



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

N-BEAT 모형 상세

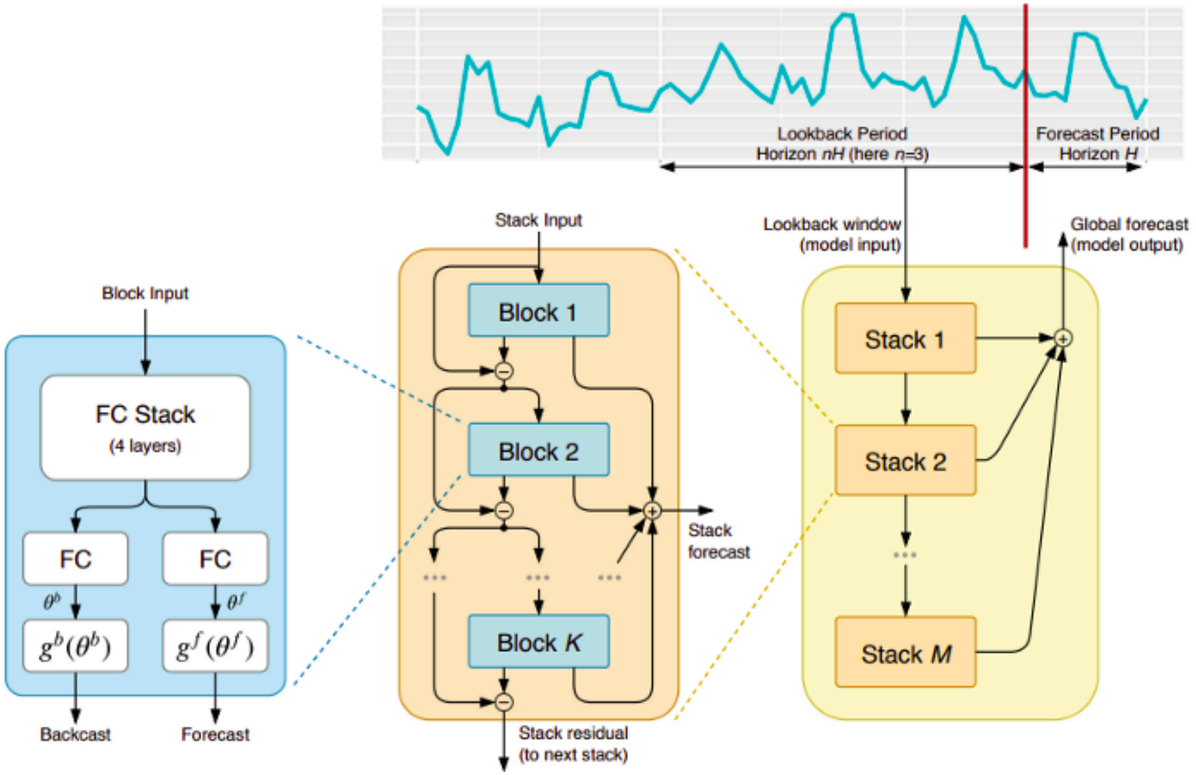
- **Doubly Residual Stacking:**
  - The idea of residual connections and stacking is so brilliant that it is used in almost every type of deep neural networks, such as deep *Convnets* and *Transformers*.
  - The same principle is applied in the *N-BEATS* implementation, but with some extra modifications: Each block has two residual branches, one running through the lookback window (called *backcast*) and the other through the predicted window (called *forecast*).



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

N-BEAT 모형 상세

- **Interpretability:**
  - The model has two variants, *general* and *interpretable*.
  - In the *general* variant, the final weights in the fully-connected layers of each block are learned by the network arbitrarily.
  - In the *interpretable* variant, the last layer of each block is removed.
  - Then, the *backcast* and *forecast* branches are multiplied by specific matrices that mimic **trend** (monotonic function) and **seasonality** (periodic cyclical function).



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

RNN

RNN

- RNN(Recurrent Neural Networks)은 ANN 모형의 입력층에서 출력층으로의 일방향 데이터 전달 구조에 변형을 가하여 은닉층 또는 출력층에서 입력층으로의 피드백 구조를 도입하여 네트워크가 기억력을 가지게 만들어 언어 데이터 처리에 우수한 능력을 보인다.

- 시계열 데이터에 내재되어 있는 temporal structure를 잘 파악해낸다.

- RNN 모형 중에서 시계열 에측에 주로 사용되는 LSTM(Long Short-Term Model)은 관측치간의 순서를 명시적으로 모형에 반영하여 입력과 출력간의 함수관게를 학습하는데, 이런 기능은 DNN이나 CNN에는 존재하지않는다.

- RNN 모형은 일반적으로 데이터에 존재하는 시간 의존성(Temporal Dependence)을 학습하는 기능을 지니며, 특히 LSTM은 데이터에 존재하는 장기간에 걸친 연관관계를 잘 파악해낸다.

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

RNN

- LSTM
- CNN-LSTM
  - A deep cnn-lstm model for particular matter (Huang, C.-J and Kuo, P.-H, 2018)
  - Predicting residual energy consumption using CNN-LSTM neural networks (Kim, T. et al., 2019)

- DeepAR (Salinas, D. et al., 2019)
  - DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks

- MQRNN (Multi-Horizon Quantile Recurrent Forecaster)

- DSSM (Deep Space State Models)

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

DeepAR

DeepAR 개요

- **수없이 많은 시계열을 동시에 예측해야 하는 상황이 더욱 빈번해짐**
  - 기존에도 여러 시계열의 정보를 활용하는 multivariate timeseries 기법들이 있었지만, scalable하지 않음
  - 이를 위해 주로 가장 중요한 품목들 위주로 분석을 진행하거나 상위 카테고리로 그룹화 한후 분석 진행
- **기존의 전통적 시계열방법은 판단과 수작업이 많이 들어감**
  - 시계열의 정상성, 단위근 파악 등
  - 특성 파악후 많은 전처리 작업이 필요
    - Ex) 추세 제거, 차분, 지시변수 생성 등
  - 이러한 작업들 역시 scalability에 영향을 줌

Figure 1: Log-log histogram of the number of items versus number of sales for the 500K time series of ec, showing the scale-free nature (approximately straight line) present in the ec dataset (axis labels omitted due to the non-public nature of the data).

https://www.youtube.com/watch?v=okyo61ZZivA

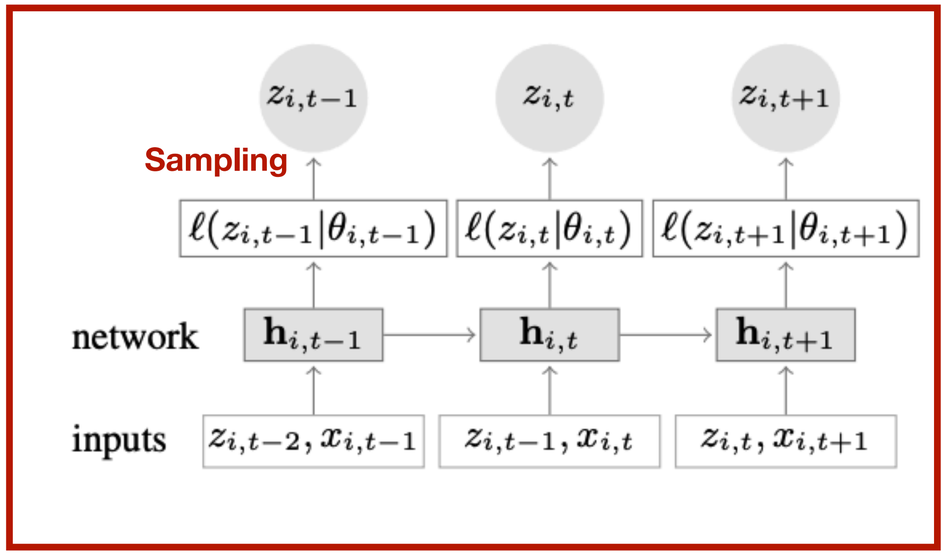
Key advantages

- **Feature engineering의 수고가 줄어듬**
  - 원래 시계열을 거의 그대로 입력 (내부적으로 autoscaling 적용)
- **확률적인 예측이 가능**
  - 신뢰구간 제공이 가능
  - loss의 종류에 따라서, 기대값이 아니라, 기대 quantile을 구할 수 있음
- **적은 데이터에도 효과가 있음**
  - 수많은 시계열의 정보를 빌려올 수 있음
  - 특히 서로 다른 scale(혹은 분포)의 시계열을 동시 예측 가능
- **다양한 정보를 모델링 할 수 있음**
  - Static한 variable (독립변수)
  - 기타 시계열 혹은 미래의 알려진 값까지 모델링이 가능
- **다양한 likelihood를 고려하여, 다양한 형태의 데이터를 모델링할 수 있음**
  - Binary response, Poison response

Figure 1: Log-log histogram of the number of items versus number of sales for the 500K time series of ec, showing the scale-free nature (approximately straight line) present in the ec dataset (axis labels omitted due to the non-public nature of the data).

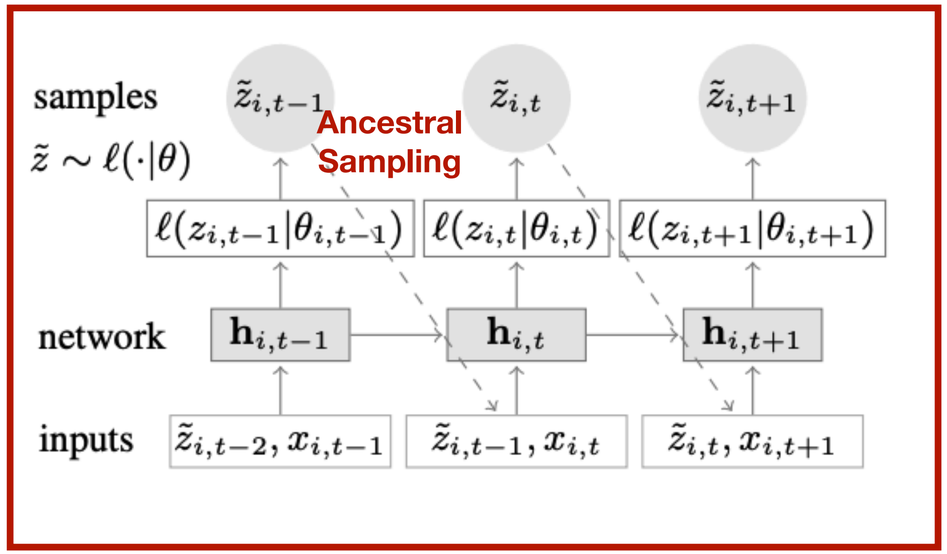
https://www.youtube.com/watch?v=okyo61ZZivA

Sequence to sequence 구조



Encoder

Training Phase

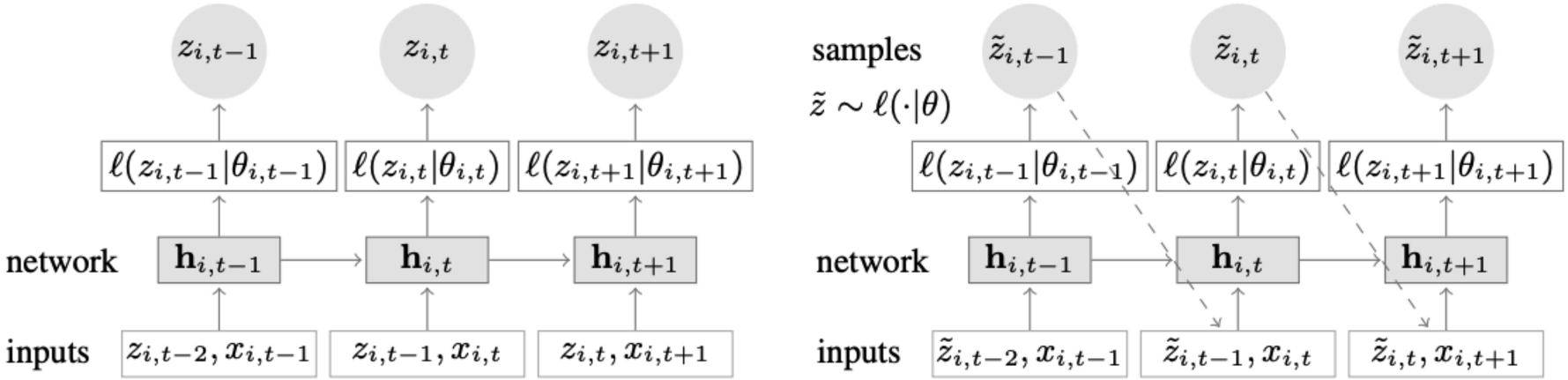


Decoder

Prediction Phase

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Likelihood



- 신경망은 시계열 정보, z,의 likelihood, $l(z\,|\,\theta)$,의 paramter를 예측하는 데 활용
- 논문에서는 두개의 likelihood를 사용
  - Gaussian likelihood

Softplus activation

$$l_G(z\,|\,\mu,\sigma) = (s\pi\sigma^2)^{-1/2}\exp\left(-(z-\mu)^2/(2\sigma^2)\right), \quad \mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^T\mathbf{h}_{i,t}, \quad \sigma(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\sigma^T\mathbf{h}_{i,t}+b\sigma))$$

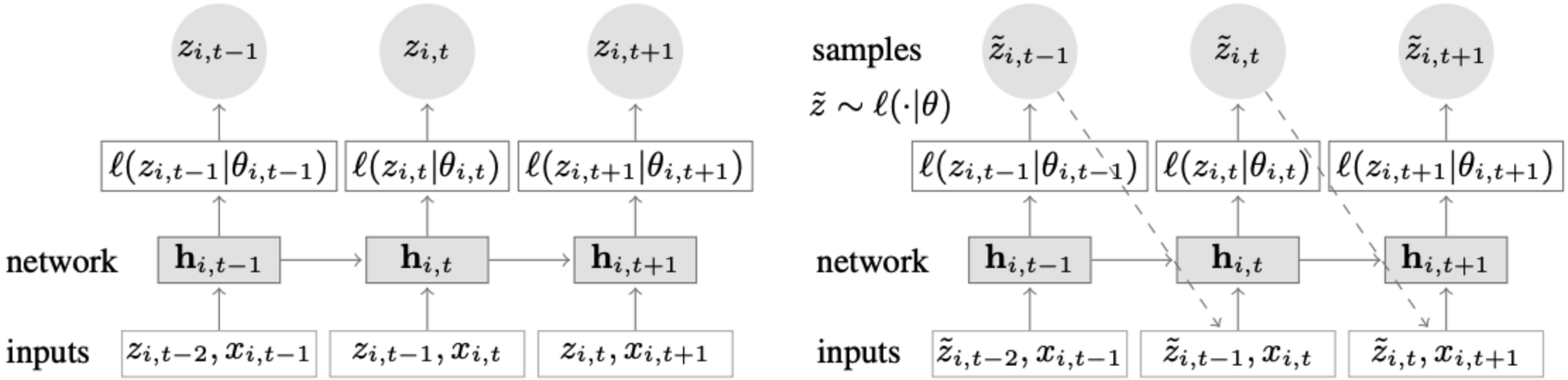
  - Negative binomial likelihood: positive count

$$l_{NB}(z\,|\,\mu,\alpha) = \frac{\Gamma\left(z+\frac{1}{\alpha}\right)}{\Gamma(z+1)\Gamma\left(\frac{1}{\alpha}\right)}\left(\frac{1}{1+\alpha\mu}\right)^{1/\alpha}\left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z}, \quad \mu(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\mu^T\mathbf{h}_{i,t}+b\mu)), \quad \alpha(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\alpha^T\mathbf{h}_{i,t}+b\alpha))$$

α: Shape parameter

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Likelihood

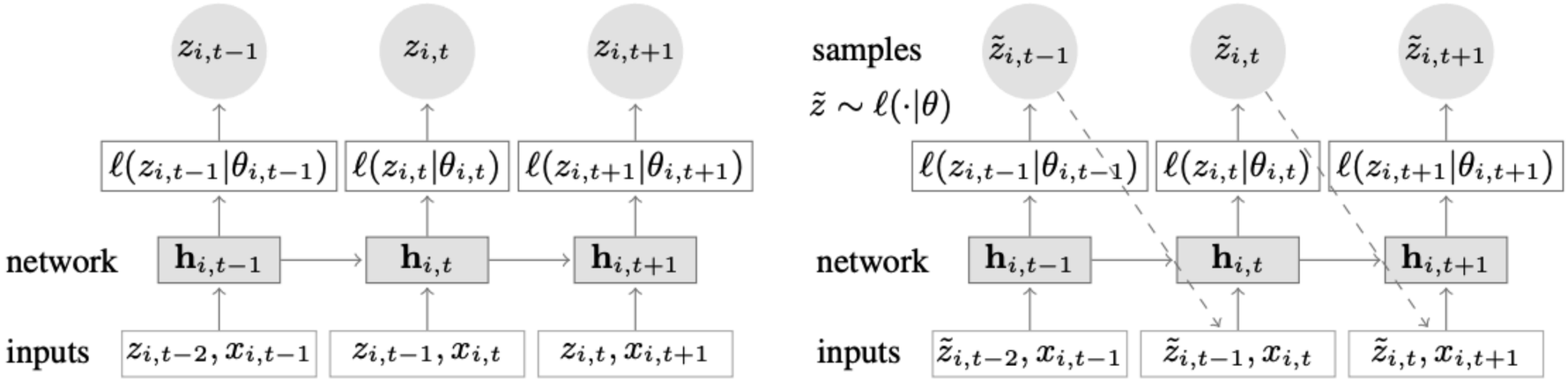


- Input:
  - 과거의 관측치 값: $z_1, \ldots, z_{t_0-1}$
  - 독립변수들: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$
- Output: 추정된 결합확률 분포: $P(Z_{t_0}, Z_{t_0+1}, \ldots, Z_T)$
- 정해진 크기의 임의의 window 내의 값들을 이용하여, MLE를 구하는 방식으로 학습
- 결국 likelihood는 다음과 같음

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \log l(z_{i,t} \,|\, y_t(\mathbf{h}_{i,t}))$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

2 Issues: Scale and Longtail

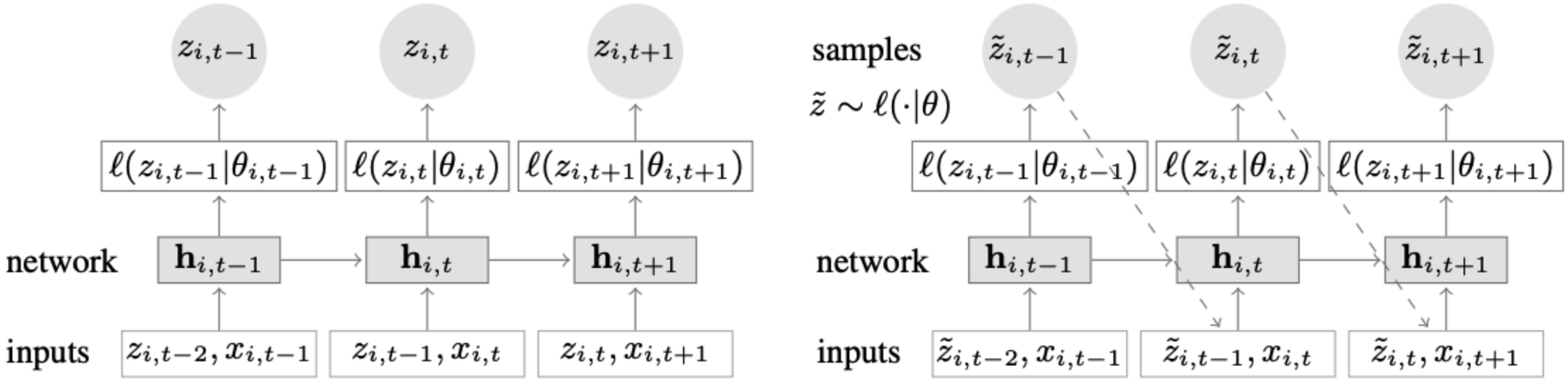


- 두가지 문제가 존재함
  - 시계열의 scale이 크게 다르다. (대다수의 품목이 아주 적게 팔린다)
  - 주로 관심을 가져야 할 시계열의 개수가 크게 적다. (소수의 품목만이 아주 많이 팔린다.)

- 두가지 해결책을 제시
  - Automatic Scaling
  - Weighted sampling to counter-balance power-law behavior

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

2 Issues: Scale and Longtail



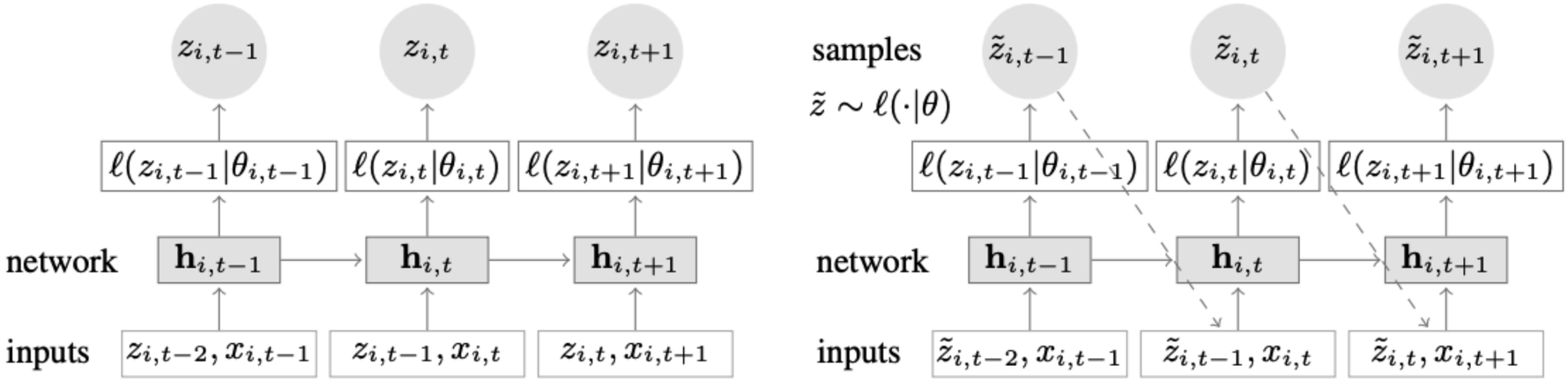
Automatic scaling:

- ***i*번째 시계열의 자기상관 입력인** z 를 v_i라는 scaling factor를 이용해서 scaling함
- v_i는 단순히 시계열의 평균값으로, 논문에서는 다음의 식을 활용 (Heuristic method)

$$v_i = 1 + \frac{1}{t_0} \sum_{t=1}^{t_0} z_{i,t}$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

2 Issues: Scale and Longtail



Handling heavy tail distribution (Weighted sampling):

- Scale에 비례해서 학습셋에 포함될 확률을 높여줌

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Transformers

인과형 예측기법

Transformers

- Informer (Zhou, H. Et al., 2021)
  - Informer: Beyond efficient transformer for long sequence time-series forecasting
- Spacetimeformer (Jake Grigsby, 2019)
  - Long-Range Transformers for Dynamic Spatiotemporal Forecasting
- Temporal Fusion Transformer (Bryan Lim et al., 2020)
  - Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

Spacetimeformer

Spacetimeformer 개요

- **We want the model consider both temporal and spatial relationships.**
- In time series forecasting with transformer-based models, a popular technique to produce time-aware embeddings is to pass the input through a ***Time2Vec*** [6] embedding layer
- While this technique works really well for univariate time sequences, it doesn't make any sense for multivariate time inputs.
  - 다중시계열은 embedding과는 달리 vector로서의 의미가 없음
- In language modeling, each word of a sentence is represented by an embedding, and a word is essentially a **concept**, part of a vocabulary.

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

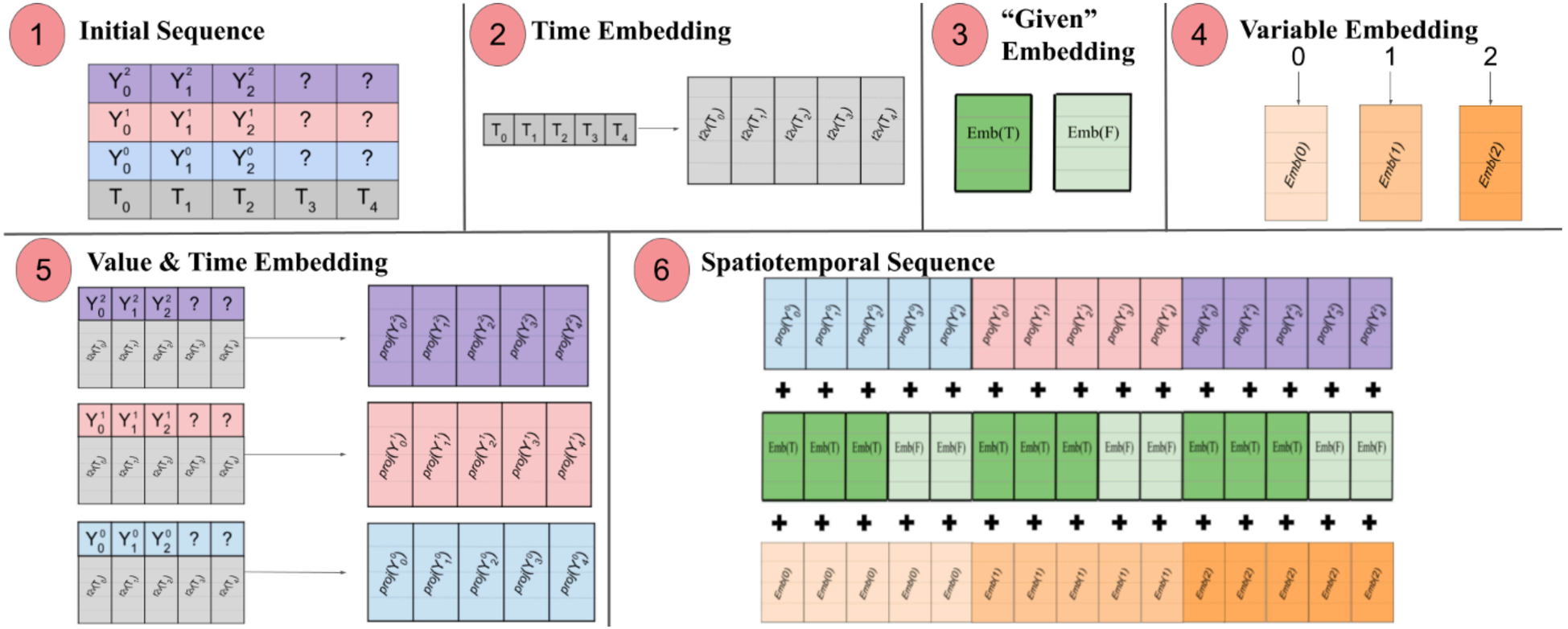
Spacetimeformer 개요

- *Spacetimeformer* addresses this issue by flattening the input into a single large vector, called **spatiotemporal sequence.** If the input consists of N variables, organized into T timesteps, the generated spatiotemporal sequence will have $(N \times T)$ tokens.

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Spacetimeformer 개요

Embedding 가공 방법

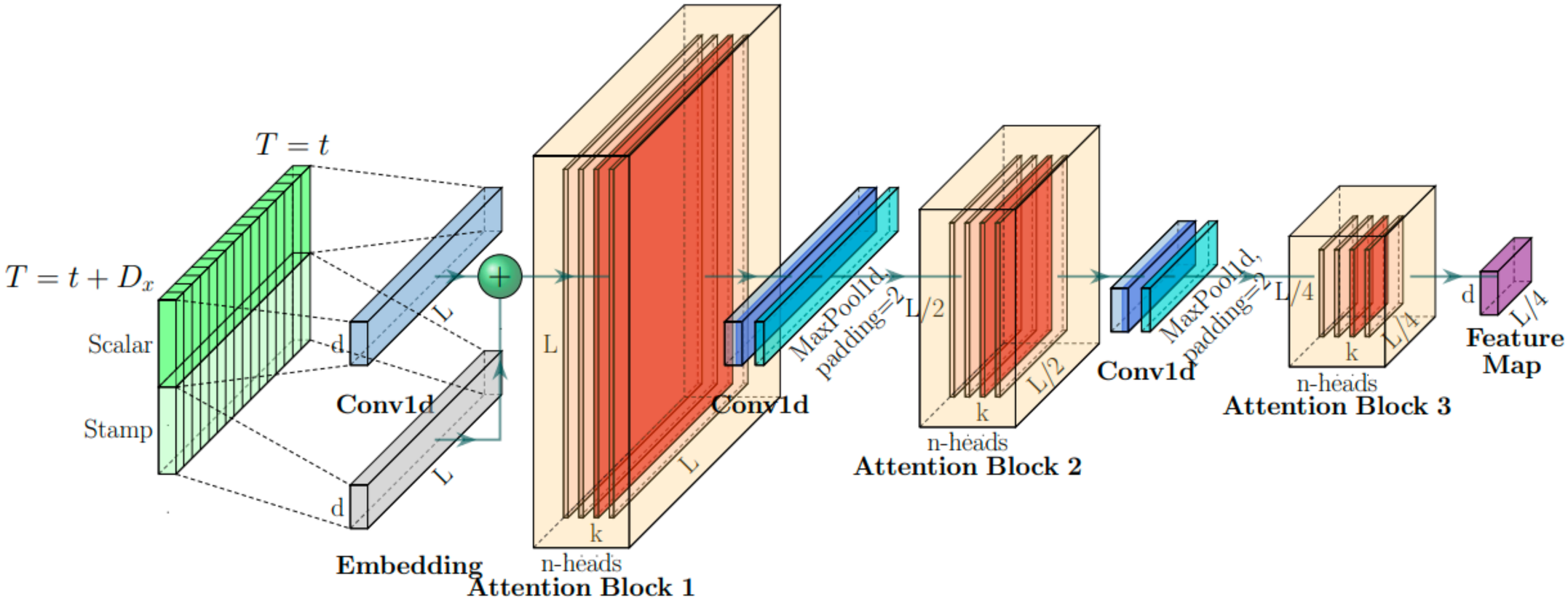


https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Informer

Informer 개요

Architecture



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0