낙서

사서림

2025.05.26.__

Contents

Part I

part

Chapter 1

chapter

1.1 section

1.1.1 subsection

Part II

DBBD 관련

Chapter 2

DBBD 관련 수식

만약 가로 H, 세로 W 인 이미지를 한번에 K 개씩 각각 x,y 축으로 분해한다면 최대 분해 가능한 깊이 M 은 다음과 같다. (참고로 K 는 x 축 분활 갯수와 y 분할 갯수의 곱이다. 즉 $K =$ x 축 분할 갯수 $\times$ y 축 분할 갯수)

$$M = \lceil \log_K HW \rceil$$

이때 a 를 의미 표현 보존률이라고 하면 a 를 구하는 방법은 다음과 같다.

$$M = \lceil \log_K aHW \rceil$$

이때 올림을 제거하고 보면

$$M = \log_K aHW$$

에서 전개하면

$$M = \log_K aHW = \log_K a + \log_K HW \to M - \log_K HW = \log_K a \to a = K^{M - \log_K HW} = \frac{K^M}{K^{\log_K HW}} = \frac{K^M}{HW}$$

임으로 의미 표현 보존률 a 는 다음과 같다.

$$a = \frac{K^M}{HW}$$

이때 a 는 M 의 최대 깊이로 HW 크기의 이미지일 시 최대 깊이로 이미지의 몇% 를 유의미하게 분해할 수 있는지를 나타낸다.
이때 $\sqrt{a}$는 H, W 에 각각 곱하면 100% 유의미하게 분해할 수 있는 비율이다.
이때 M 의 최대 깊이 일시 a 가 1 인 $L \times L$크기인 정사각형이 되는 L 을 찾으면 다음과 같다.

$$1 = \frac{K^M}{L^2} \to L^2 = K^M \to L = \sqrt{K^M}$$

그럼 이제 이것을 일반화 시켜서 $(\mathbb{R}^+)^n$ 에서의 분해를 살펴보자 만약 모든 각 차원을 k 개로 분활한다면 $K = k^n$ 이다. 이때 M 과 a 그리고 L 에 관한 공식은 아래와 같다.

$$\text{shape} = \{L_1, L_2, \ldots, L_n\}\text{일시} \qquad M = \left\lceil \log_K \left(\prod_{i=1}^{n} L_i \right) \right\rceil \text{ 임으로} \qquad M = \left\lceil \log_{k^n} \left(\prod_{i=1}^{n} L_i \right) \right\rceil \Rightarrow M = \left\lceil \frac{1}{n} \log_k \left(\prod_{i=1}^{n} L_i \right) \right\rceil$$

$$a = \frac{K^M}{\prod_{i=1}^{n} L_i} = \frac{(k^n)^M}{\prod_{i=1}^{n} L_i} = \frac{k^{nM}}{\prod_{i=1}^{n} L_i}$$

$$L^n = K^M \to L = \sqrt[n]{K^M} = \sqrt[n]{(k^n)^M} = \sqrt[n]{(k^M)^n} = k^M$$

이제 일반화 된 식들을 보자

$$M = \left\lceil \frac{1}{n} \log_k \left(\prod_{i=1}^{n} L_i \right) \right\rceil \qquad a = \frac{k^{nM}}{\prod_{i=1}^{n} L_i} \qquad L = k^M$$

여기에서 $k = 2$ 라고 해보자

$$M = \left\lceil \frac{1}{n} \log_2 \left(\prod_{i=1}^{n} L_i \right) \right\rceil \qquad a = \frac{2^{nM}}{\prod_{i=1}^{n} L_i} \qquad L = 2^M$$

여기에서 시간축 T 를 추가하여 가로 H, 세로 W 인 동영상이 있다고 하자. 그럼 식은 아래와 같아진다.

$$M = \left\lceil \frac{1}{3} \log_2 (HWT) \right\rceil \qquad a = \frac{2^{3M}}{HWT} = \frac{8^M}{HWT} \qquad L = 2^M$$

이때 만약 FHD 급인 영상이 10 분에 60Hz 짜리라고 해보자 그러면 $H = 1920, W = 1080, T = \text{fps} \times 60 \times 10 = 60 \times 60 \times 10 = 36,000$ 이다. 그럼 이때 M 을 구해보자.

$$M = \left\lceil \frac{1}{3} \log_2 (1920 \times 1080 \times 36000) \right\rceil = \left\lceil \frac{1}{3} \log_2 (74,649,600,000) \right\rceil \approx \lceil 12.03980516 \rceil = 13$$

그런대 시스템 상 최대 부하가 $M = 10$ 이라고 하자 이때 이 동영상의 몇% 를 유의미하게 분해할 수 있는지를 나타내는 a 를 구해보자.

$$a = \frac{8^1 0}{74,649,600,000} \approx 0.01438375857 \approx 0.0144 = 1.44[\%]$$

무려 1% 정도만 유의미하다는 충격적인 결과가 나왔다. 그럼 이번에는 $M = 12$ 라고 해보고 다시 계산해보자.

$$a = \frac{8^1 2}{74,649,600,000} \approx 0.9205605487 \approx 0.9206 = 92.06[\%]$$

이때는 90% 가 넘음으로써 정보를 충분히 생략하는 수준에 머문다. 그럼 3 차원 데이터량 $Data_3 = HWT$ 라고 할시 이를 모두 동일한 길이 L 로 나누어 보자.

$$L = \sqrt[3]{Data_3} \approx 4210.585543$$

이때 $M = 13$ 일때의 L 과 비교해보자.

$$L_{\text{stand}} = 2^{13} = 8192 \qquad L \approx 4211 \qquad \frac{L_{\text{stand}}}{L} \approx 1.9453811446212301116124436000 95$$

임을 알 수 있다.

만약 DBBD 를 이용해서 도트화처럼 하기 위하여 $M = \lceil \log_K(HW) \times 2^{-\frac{2}{3}} \rceil$ 로 M 을 구하여 했다고 하자 그럼 $M = \lceil \log_K aHW \rceil$ 으로 바꾸어 a 로 쓸 수 있는지에 대하여 알아보겠다. 간단히 알아보기 위하여 올림은 제거하고 보겠다. 그럼 아래와 같이 된다.

$$M = \log_K(HW) \times 2^{-\frac{2}{3}} \quad M = \log_K aHW \text{이므로} \quad \log_K(HW) \times 2^{-\frac{2}{3}} = \log_K aHW$$

여기에서 a 로 정리하면 다음과 같다.

$$\log_K(HW) \times 2^{-\frac{2}{3}} = \log_K aHW = \log_K a + \log_K HW \Rightarrow \log_K(HW) \times 2^{-\frac{2}{3}} - \log_K(HW) = \log_K a$$

$$\left(2^{-\frac{2}{3}} - 1\right) \log_K HW = \log_K a$$

이때 $2^{-\frac{2}{3}}$ 를 도트 비율 조절 상수 τ 라고 하자 그럼 아래와 같이 바뀐다.

$$(\tau - 1) \log_K HW = \log_K a$$

여기에서 정리하면

$$a = K^{(\tau-1)\log_K HW} = \left(K^{\log_K HW}\right)^{(\tau-1)} = HW^{(\tau-1)}$$

그러므로 τ 에 대하여 a 는 다음과 같다.

$$a = HW^{(\tau-1)}$$

τ 를 구해보자

$$\tau = 1 + \log_{HW}(a)$$

이번에는 한 깊이에서 최대로 생성 될 수 있는 블록의 수에 대한 공식을 말하겠다. 그 공식은 아래와 같다.

$$\text{최대로 생성 될 수 있는 블록의 수} = K^M$$

이때 하이퍼 트리의 개념을 적용하면 다음과 같다.

$$\text{최대로 생성 될 수 있는 블록의 수} = \sum_{i=1}^{M} K^i$$

그럼 이때 $M = 10$ 일 때 일반 생성과 하이퍼 트리 생성을 비교하면 최대 깊이가 아닌 블록의 수를 구할 수 있다.

$$\text{최대 깊이가 아닌 블록의 수} = \sum_{i=1}^{M} K^i - K^M = \sum_{i=1}^{M-1} K^i$$

그럼 하이퍼 트리시 낭비 되는 메타 데이터 (최대 깊이가 아닌 블록) 의 비율을 구해보자.

$$\text{하이퍼 트리에서 메타 데이터로 낭비 되는 블록의 비율} = \frac{\sum_{i=1}^{M-1} K^i}{\sum_{i=1}^{M} K^i} = \frac{\sum_{i=1}^{M-1} K^i}{K^M + \sum_{i=1}^{M-1} K^i} = 1 - \frac{K^M}{K^M + \sum_{i=1}^{M-1} K^i}$$

이때 $K = 4, M = 13$ 일시 발생하는 메타 데이터로 낭비 되는 비율은 $\frac{\sum_{i=1}^{12} 4^i}{\sum_{i=1}^{13} 4^i} \approx 24.99999888[\%]$ 임을 알 수 있다.

이때 $K = 4, M = 1$ 일시 발생하는 메타 데이터로 낭비 되는 비율은 방금 구한 하이퍼 트리에서 메타 데이터로 낭비 되는 블록의 비율 공식으로 구하지 못한다. 그러나 애초에 낭비되는 블록이 하나도 없음을 알 수 있다.

이때 $K = 4, M = 2$ 일시 발생하는 메타 데이터로 낭비 되는 비율은 $\frac{\sum_{i=1}^{1} 4^i}{\sum_{i=1}^{2} 4^i} = 20[\%]$ 임을 알 수 있다.

이때 $K = 4, M = 100$ 일시 발생하는 메타 데이터로 낭비 되는 비율은 $\frac{\sum_{i=1}^{99} 4^i}{\sum_{i=1}^{100} 4^i} \approx 25[\%]$ 임을 알 수 있다.

즉 일반적으로 전체 블록의 최대 $20[\%] \sim 25[\%]$ 의 블럭이 메타 데이터로 소모될 가능성이 있는 것이다.

그럼 이때 M 을 ∞ 로 보내보자. 그 결과는 아래와 같다.

$$\lim_{M \to \infty} \frac{\sum_{i=1}^{M-1} K^i}{\sum_{i=1}^{M} K^i} = \frac{1}{K} \quad \text{if} \left(\frac{1}{K} \quad K\right) \in \mathbb{R}^2 \wedge \log(K) > 0$$

그럼으로 $K = 4$ 일시 최대 $25[\%]$ 의 블록이 메타 데이터로 소모될 가능성이 있다고 볼 수 있다.

이번에는 이상적인 트리를 만들어보자. 루트 (한번도 쪼게지지 않은 원본) 주소를 0 이라고 할시 규칙과 주소를 추적하는 법을 만들어보겠다.
아래는 $K = 4, M = 3$ 일시 이상적인 트리의 주소들이다.

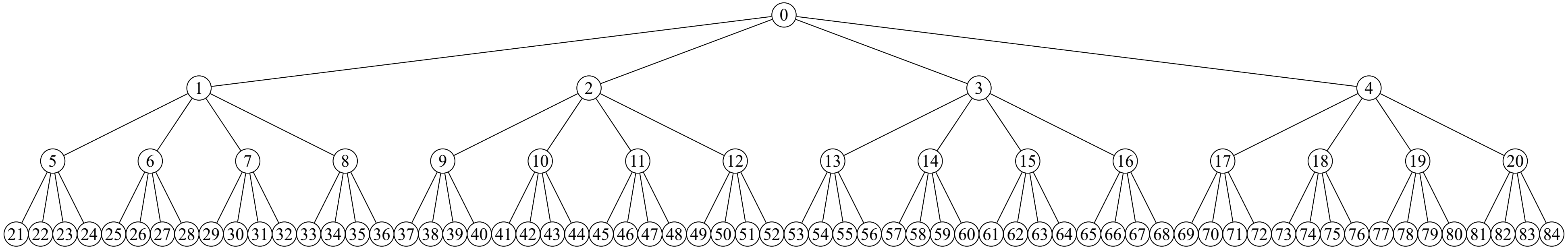

Figure: 트리 기반 블록 인덱스 시각화

이 때 1 번이 생성할 수 있는 주소는 45, 46, 47, 48 번이다. 이를 유도하는 식을 세워 보자. 11 의 부모 주소는 2, 2 의 부모 주소는 0 이다. 이때 0 은 깊이가 0, 2 은 깊이가 1, 11 은 깊이가 2 이다. 이를 이용하여 구할 수 있다. 11 의 위치는 깊이 1 인 부모가 2 번째이며, 깊이가 2 인 본인이 $11 - \sum_{i=1}^{2-1} 4^i = 11 - 4 = 7$ 임으로 7 번째임을 알 수 있다. 그럼 이 때 7 의 자식 주소 중 가장 낮은 주소는 $\sum_{i=1}^{3-1} 4^i + (7-1) \times \sum_{i=1}^{2-1} 4^i + 1 = 4 + 16 + (7-1) \times 4 + 1 = 20 + 24 + 1 = 45$ 임을 알 수 있다. 그리고 자식 주소 중 가장 높은 주소는 $\sum_{i=1}^{3-1} 4^i + 7 \times \sum_{i=1}^{2-1} 4^i = 4 + 16 + 7 \times 4 = 20 + 28 = 48$ 임을 알 수 있다.

그럼으로 D 을 깊이, N 을 현재 주소라고 하면 공식은 아래와 같다.

$$\text{본인 노드가 같은 층에 있는 노드 중 몇 번째인지} = N - \sum_{i=1}^{D-1} 4^i$$

$$\text{본인 노드의 자식 노드 중 가장 낮은 주소} = 1 + \sum_{i=1}^{D} 4^i + 4\left(-1 + N - \sum_{i=1}^{D-1} 4^i\right)$$

$$\text{본인 노드의 자식 노드 중 가장 높은 주소} = \sum_{i=1}^{D} 4^i + 4\left(N - \sum_{i=1}^{D-1} 4^i\right)$$

여기에서 K 를 일반화 하면 아래와 같다.

$$\text{본인 노드가 같은 층에 있는 노드 중 몇 번째인지} = N - \sum_{i=1}^{D-1} K^i$$

$$\text{본인 노드의 자식 노드 중 가장 낮은 주소} = 1 + \sum_{i=1}^{D} K^i + K\left(-1 + N - \sum_{i=1}^{D-1} K^i\right)$$

$$\text{본인 노드의 자식 노드 중 가장 높은 주소} = \sum_{i=1}^{D} K^i + K\left(N - \sum_{i=1}^{D-1} K^i\right)$$

이때 47 번의 부모를 구해보자. 47 은 깊이가 3 인것을 이미 알고 있다는 가정하에 다음과 같이 구할 수 있다. $\left\lceil \frac{N - \sum_{i=1}^{D-1} 4^i}{4} \right\rceil + \sum_{i=1}^{D-2} 4^i = \left\lceil \frac{47-4-16}{4} \right\rceil + 4 = \left\lceil \frac{27}{4} \right\rceil + 4 = \lceil 6.75 \rceil + 4 = 7 + 4 = 11$ 임으로 11 이 47 의 부모이다.

그럼으로 D 를 깊이, N 을 현재 주소라고 하면 공식은 다음과 같다.

$$\text{본인 노드가 같은 층에 있는 노드 중 몇 번째인지} = N - \sum_{i=1}^{D-1} K^i$$

$$\text{본인 노드의 자식 노드 중 가장 낮은 주소} = 1 + \sum_{i=1}^{D} K^i + K\left(-1 + N - \sum_{i=1}^{D-1} K^i\right)$$

$$\text{본인 노드의 자식 노드 중 가장 높은 주소} = \sum_{i=1}^{D} K^i + K\left(N - \sum_{i=1}^{D-1} K^i\right)$$

$$\text{본인 노드의 부모 노드의 주소} = \left\lceil \frac{N - \sum_{i=1}^{D-1} K^i}{K} \right\rceil + \sum_{i=1}^{D-2} K^i$$

$$\text{단 } \sum_{i=0}^{I} K^i, I < 0 \text{일시 그 부분은 } 0 \text{ 으로 처리해야 정합하게 돌아간다.}$$

이 공식으로 이상적인 K 진 트리를 만들어서 DBBD 로 인한 유동적으로 자식 노드가 $1 \sim K$ 나올 시 저기에 K 보다 자식 노드가 적을 시 자식 노드의 숫자 만큼 넣고 나머지는 빈공간으로 만드는 방식으로 정규화 시켜서 넣을 수 있게 되었다.

i 깊이에서 상위 깊이로 가는 식은 아래와 같다.

$$s_{i+1}(x) = \left(\frac{W_i}{n} AB_i + b_i \right)$$

$B_i \in \mathbb{R}^{n\times 15}$ n 개의 블록 $A \in \mathbb{R}^{1\times n}$ 평균 벡터 $W_i \in \mathbb{R}$ 강도 조절

$b_i \in \mathbb{R}^{1\times 15}$ 기준선 조절 $S \in \mathbb{R}^{15\times 1}$ 의미 압축 벡터 $s_{i+1}(x) \in \mathbb{R}$ 하나의 스칼라 출력

Part III

생성형 AI 에 관한 나의 생각

Chapter 3

공간에 관한 고찰

ChatGPT 4o 에 한번 랜덤한 프롬프트를 넣는다고 생각해보자. 여러분들은 어떤 프롬프트를 넣을 것인가? 그런대 이것을 자세하게 분석해보면 글, 그림이나 동영상, 파일을 넣을 수 있음을 알 수 있다. 이것을 이용하여 우리는 3 가지 공간을 알 수 있다.

$$T = \text{문자 공간} \qquad M = \text{이미지-동영상 공간} \qquad F = \text{파일 공간}$$

이 때 T, M, F 는 전체 공간 U 에 속한다. 즉 $T, M, F \subset U$ 이다. 다만 U 의 요소가 T, M, F 만 있는 것은 아니며 더 다양하게 있을 수도 있으며 이는 어떻게 각각 하나의 의미 공간으로 묶어서 분류하느냐에 따라서 전체 공간의 구성이 달라질 수도 있음을 알린다.

이때 컴퓨터에서 표현가능한 공간을 $B_R = \{0,1\}^\infty$ 라고 해보자 이때 컴퓨터는 무한을 표현할 수 없음으로 현실적으로 $B = \{0,1\}^n, \ \ n < \infty$ 임을 알 수 있다.
B 와 U 가 서로 손실 없이 상호 사상이 될 때 데이터를 온전히 다룰 수 있다. 즉 $U \Leftrightarrow B$ 이어야 한다. 그러므로 생성형 AI 를 사상으로 표현하면 다음과 같다.

$$\text{생성형 AI} : U \to U$$

그럼 다시 ChatGPT 4o 로 돌아와서 $U = \{T, M, F\}$ 라고 하면 입력 가능한 공간은 일반적으로는 다음과 같다.

$$\mathcal{P}(U) \setminus \phi = \{\{T,M,F\},\{T,M\},\{T,F\},\{M,F\},\{T\},\{M\},\{F\}\}$$

그럼 여기에서 그림 생성형 AI 사상을 만들어보자면 다음과 같다.

$$m_k, m_{k+1} \subset M \text{이며 } m_k \text{다음이 } m_{k+1} \text{공간일 시}$$

$$\text{Transform} : m_k \to \{m_{k+1}, m_{k+1}, \ldots, m_{k+1}\} := m^m \qquad \text{Group} : m^m \times \text{Condition} \to m_{k+1}$$

그럼 이제 T, M, F 공간이 실제로 각각 어떤 공간인지 알아보자

$$T = \text{String 공간} \qquad M = \text{Tensor 공간} \qquad F = B\text{공간의 요소를 File 해더에 따라 정의한 공간}$$

이때 만약 T, M, F 서로 손실 없이 상호 사상이 되는 경우 (또는 손실이 감당 가능할 정도로 적게 발생하면서 사상이 되는 경우) 즉 $T \Leftrightarrow M \Leftrightarrow F$ 일 시에는 T, M, F 공간 중 가장 유리한 공간에서 동작하는 모델로 돌려야 이득이다.
이제 M 공간에서의 생성형 인공지능 정의를 U 공간에서 생성하는 인공지능의 정의로 일반화 하면 다음과 같다.

$$u_k, u_{k+1} \subset U \text{이며 } u_k \text{다음이 } u_{k+1} \text{공간일 시}$$

$$\text{Transform} : u_k \to \{u_{k+1}, u_{k+1}, \ldots, u_{k+1}\} := u^u \qquad \text{Group} : u^u \times \text{Condition} \to u_{k+1}$$

이때 LLM 을 위한 Transformer 를 CNN 으로 구현하는 법을 생각해볼 수 있다. 즉 Transform 사상의 결과인 u^u 를 텐서 공간에서 각각의 유사도나 관계 등에 따라 u^u 의 요소 각각의 텐서 공간안의 요소로 잘 정의하면 Attention 을 정의하지 않아도 자연스럽게 Attention 이 텐서 공간안에서 요소와 요소사이의 거리같은 것으로 구해질 가능성이 있다고 볼 수 있다.

Part IV

객체론

Chapter 4

객체론 기본 정의

Part V

함수공간의 요소 W,b 를 이용한 생성형 MLP 이론

Chapter 5

블록 분해 기법에 관한 고찰

Chapter 6

하이퍼 트리

하이퍼 트리의 구조
1. 모든 노드에 데이터 값을 가진다.
2. 데이터 값은 Object 나 추상 클래스 상속을 받아서 모든 데이터 유형을 리스트로 관리한다.
2.1. 또는 더욱 복잡한 버전으로는 List 의 요소 각각을 하나의 데이터 값 유형을 가지는 하위 List 로 가지게 하여 만든다.
3. 이 구조는 그래프의 업그래이드 버전이다.(자동 미분 처럼 모든 데이터 유형을 List 나 List.List 에 저장할 때 추상클래스로 한번 덮고 요소로 넣어야 할 것 같다.)
4. 쉽게 생각해서 이 자료구조는 이런 것이다. 차트나 전이 함수를 생각해보고 한 다양체에서 한 열린공간에서 다른 열린 공간으로 갈 때에는 그냥 선형적으로 기억해도 되며 그를 이용하여 전개할 수 있다. 한마디로 추상화된 선형공간인 것이다. 그와 같이 하이퍼 트리 자료구조 또한 선형성을 추상적으로 가지고 있다고 볼 수 있다. 그래서 다른 이름으로는' 선형 하이퍼 트리' 라고 한다.

Chapter 7

생성형 MLP 이론

퍼셉트론은 $F : \mathbb{R}^2 \to \mathbb{N}_0$ 에서 $F : \mathbb{R}^n \to \mathbb{N}_0, F : \mathbb{R}^n \to (\mathbb{N}_0)^m, F : \mathbb{R}^n \to \mathbb{R}^m$ 의 형식으로 발전해왔다.

하나의 퍼셉트론 (하나의 노드) 를 $F : \mathbb{R}^n \to \mathbb{R}$ 라고 하자. 그럼 F 들의 결과를 쌓은 것은 선형함수를 $L : \mathbb{R}^n - > \mathbb{R}^m$ 비선형 함수를 $Q : \mathbb{R}^m - > \mathbb{R}^m$ 라고 할때 그렇다면 한 층의 모든 노드에 관한 식은 $Q(L(x))$ 이다. 그리고 순전파는 $Q(L(Q(L(Q(L(\ldots)))))) = M(x)$ 이다. $X \to Y$(모델에서의 입력-> 출력) 으로 가는 정제된 해석은 대부분 벡터장이다.(모든 모델의 본질은 $M : X \to Y$ 이라는 사상이며, 이때 모델은 입력 공간 X 위에 정의된 벡터장처럼 동작한다.)

파라미터 W, b 를 입력공간 X 의 위상적 특징이 있는 상위 공간 K 이며 미분 가능한 함수 공간의 요소라고 하자. 그럼 우리는 이를 아래와 같이 표현 가능하다.

$$W \subset C^k(X, K), \quad X \subseteq K \subseteq \mathbb{R}^{n\times m}$$

$$b \subset C^k(X, K), \quad X \subseteq K \subseteq \mathbb{R}^n$$

여기에서 BDDB 를 이용하여 블럭 포함 여부에 따라 0, 1 을 부여하는 함수 $\Omega_i(x)$ 을 이용한 베이시스 함수를 쓰면 아래와 같다

$$W_{ij}(x) = \sum_{k=1}^{k_{\text{end}}} \Omega_{ij}(x)\alpha_{ij}\phi_k(x;\beta_k) \quad W(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}(x)e_{ij}$$

$$b_i(x) = \sum_{k=1}^{k_{\text{end}}} \Omega_i(x)\alpha_i\phi_k(x;\beta_k) \quad b(x) = \sum_{i=1}^{n} W_i(x)e_i$$

그리고 이때 베이시스 함수가 푸리에 급수라면 아래와 같다.

$$W_{ij}(x) = \sum_{k=1}^{k_{\text{end}}} \Omega_{ij}(x)\left(\beta_{k-\text{sin}}\sin(k\omega t) + \beta_{k-\text{cos}}\cos(k\omega t)\right) \quad W(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}(x)e_{ij}$$

$$b_i(x) = \sum_{k=1}^{k_{\text{end}}} \Omega_i(x)\left(\beta_{k-\text{sin}}\sin(k\omega t) + \beta_{k-\text{cos}}\cos(k\omega t))\right) \quad b(x) = \sum_{i=1}^{n} W_i(x)e_i$$

$\mathcal{L}$ 를 변분 손실함수라고 하자. 오차 하강법을 P 를 W, b 파라미터들을 모은 벡터 공간의 요소, $M(x)$ 를 MLP 순전파라고 하면 $P_{i+1} = P_i - \text{control}(P_i, \mathcal{L}, M, x)\delta P(\mathcal{L}; n(x))$ 라고 하자. 변분 손실함수는 아래와 같다.

$$\mathcal{L}(M) = \int_X \|M(x; W(x), b(x)) - y(x)\| dx$$

여기에서 P 는 $P = [W_1, b_1, W_2, b_2, \ldots, W_i, b_i]$ 인 벡터공간의 요소이다. 즉 우리는 여태 것 W, b 파라미터의 집합으로 본 것을 벡터공간의 한 점 P 로 봄으로서 학습 정도를 벡터 공간에서의 변화로 이해 할 수 있을 것이다. 이때 만약 W, b 가 함수공간의 요소일 시 P 는 W, b 각각의 파라미터 $\alpha_{ij}, \alpha_i, \beta_k$ 묶음이다.

Chapter 8

F 분해 기법

원함수 $F:\mathbb{R}^n \to \mathbb{R}^m$ 이며 정의역의 부분 공간 $D_i \subseteq \mathbb{R}^n$ 이고
근사함수 $f_i : D_i \to \mathbb{R}^m$, $f_{ik} : D_i \to \mathbb{R}$ ($\mathbb{R}^m$에서 k 번째 차원 출력) 이며
스칼라장화 한 벡터장 원함수 $F_k : \mathbb{R}^n \to \mathbb{R}$ ($\mathbb{R}^m$에서 k 번째 차원 출력) 일시

$$f_i\text{의 전체 정확도 } a(f_i) = \left(1 + \sum_{k=1}^{m} \int_{D_i} \|\nabla F_k(x) - \nabla f_{ik}(x)\| d\mathbf{x}\right)^{-1}$$

$$f_i\text{의 j 정의역 차원에서의 1 차원 정확도 } a_j(f_i) = \left(1 + \sum_{k=1}^{m} \int_{D_i \setminus X_j} \int_{X_j} \left\| \frac{\partial F_k(x)}{\partial x_j} - \frac{\partial f_{ik}(x)}{\partial x_j} \right\| dX_j d\mathbf{X}\right)^{-1}$$

이 정확도들을 이용하여 F 를 정확도에 미달하는 축이나 전체 축에 대하여 분활한다. 즉 해당 함수의 정의역 공간을 내뱉는 함수 (여기에서 분해는 [a,b] 형태로 된다.) 가 $D(F) = (D_x(F), D_y(F)) = [D_{xs}(F), D_{xe}(F)] \times [D_{ys}(F), D_{ye}(F)]$ 일시 $F : \mathbb{R}^2 \to \mathbb{R}^3$ 에 대하여 임계값이 α 이고 미달하는 축에 대하여 2 분할시 다음과 같이 나눠진다.

$$P(F) = \begin{cases} f_i : [D_{xs}(F), \frac{D_{xe}(F)}{2}] \times [D_{ys}(F), \frac{D_{ye}(F)}{2}] \to \mathbb{R}^3 & \\ f_j : [\frac{D_{xe}(F)}{2}, D_{xe}(F)] \times [D_{ys}(F), \frac{D_{ye}(F)}{2}] \to \mathbb{R}^3 & \text{if } a(F) < \alpha \vee (a_x(F) < \alpha \wedge a_y(F) < \alpha) \\ f_k : [D_{xs}(F), \frac{D_{xe}(F)}{2}] \times [\frac{D_{ye}(F)}{2}, D_{ye}(F)] \to \mathbb{R}^3 & \\ f_h : [D_{xs}(F), \frac{D_{xe}(F)}{2}] \times [D_{ys}(F), \frac{D_{ye}(F)}{2}] \to \mathbb{R}^3 & \\ f_i : [D_{xs}(F), \frac{D_{xe}(F)}{2}] \times [D_{ys}(F), D_{ye}(F)] \to \mathbb{R}^3 & \text{if } a_x(F) < \alpha \\ f_j : [\frac{D_{xe}(F)}{2}, D_{xe}(F)] \times [D_{ys}(F), D_{ye}(F)] \to \mathbb{R}^3 & \\ f_i : [D_{xs}(F), D_{xe}(F)] \times [D_{ys}(F), \frac{D_{ye}(F)}{2}] \to \mathbb{R}^3 & \text{if } a_y(F) < \alpha \\ f_j : [D_{xs}(F), D_{xe}(F)] \times [\frac{D_{ye}(F)}{2}, D_{ye}(F)] \to \mathbb{R}^3 & \\ F & \text{otherwise} \end{cases}$$

이를 F 나 f_i 들의 P 에서의 출력이 전부 자기자신이 될 때까지 각각 반복한다.
이 때 각각 D_i 에서 정의된 f_i 들은 다음과 같이 선형 함수로 정의한다.
(이는 대표적인 근사 함수일 뿐 꼭 이걸로 해야한다는 것이 아니다.)

$$f_i(x, y) = f_{ix}(x) + f_{iy}(y)$$

$$f_{ix}(x) = \frac{F(D_{xe}(f_i)) - F(D_{xs}(f_i))}{D_{xe}(f_i) - D_{xs}(f_i)} x + F(D_{xs}(f_i)) \qquad f_{iy}(y) = \frac{F(D_{ye}(f_i)) - F(D_{ys}(f_i))}{D_{ye}(f_i) - D_{ys}(f_i)} y + F(D_{ys}(f_i))$$

그럼으로 다음과 같이 결과가 나온다.

$$F \approx \bigcup_i f_i \qquad \bigcup_{i \neq j} f_i \cap f_j = A \neq \phi$$

이 때 A 공간에서 f_i 에서 f_j 로 이동이 발생할 때 경계에서의 값을 다음과 같이 하도록 하자.

$$T_{i \to j}(x) := T_{ij}(x) = f_j(x) - f_i(x) \qquad V : A \to T_{ij}(A) \to \mathbb{R}^m$$

이렇게 하면 V 는 f_i 들간 이동이 발생할 때의 순간 다차원 공역의 값 변화를 상대적으로 표현할 수 있다.
이 때 V 를 이용하여 보자

$$f_{Vij}(D_i \cap D_j \subseteq A) = V(A)(\mathbf{x} - \min(D(f_i))) + \frac{f_i(A) + f_j(A)}{2}$$

Chapter 9

문양

$z(t) = \left(t^{2.25}\cos(t) + t^2\sin(9t^{1.75})i\right) \times \exp\left(\frac{\pi t^{1.5}}{16} i\right)$