**확률은** 알려진 모델이 주어져 있고, Data를 예측하는 것이고,

**통계**는 Data가 주어져 있고, 모델을 예측하는 것

* 확률 – 함수다/parameter라고 합니다. 모평균, 모분산, 모표준편차, 모비율, 모상관관계 등
* 통계 - 무작위 표본을 기반으로 인구에 대한 속성을 추론하는 데 도움

사용할 수 있는 자료들을 최대한 정리해서 필요한 정보를 수집하는 것이 ‘**통계’**이고, 그 정보에 기초해서 확률을 계산하고 그것을 판단의 근거로 삼는 것이 **'확률**'

여기서 **확률이란**  어떤 사건이 일어날 수 있는 수학적 기대치이다

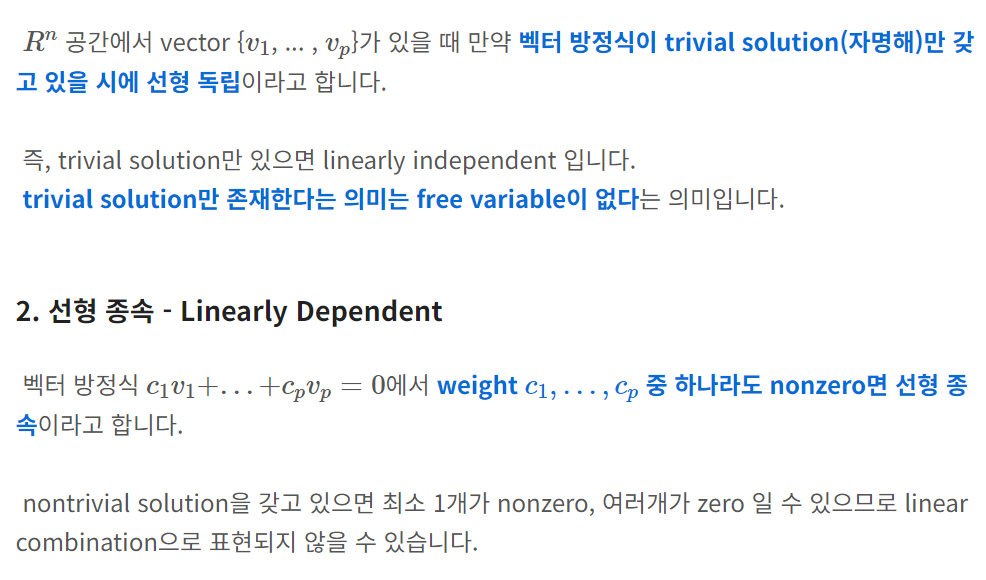
확률  = 특정 사건이 일어날 갯수 / 전체 사건이 일어날 갯수

여기서 **통계**란  이미 발생한 사건이나 앞으로 발생될 사건에 대해서 수준파악, 예측자료로 사용할 데이터분석과정으로 반복횟수가 한번이 아닌 여러번이라는 것이다.

통계학은 가능한 적은 데이타로 전체를 파악하려는 반면

기계학습은 가능한 많은 데이타로 전체를 파악하려고 한다.

AI > ML > neural network ML > DL(layer)

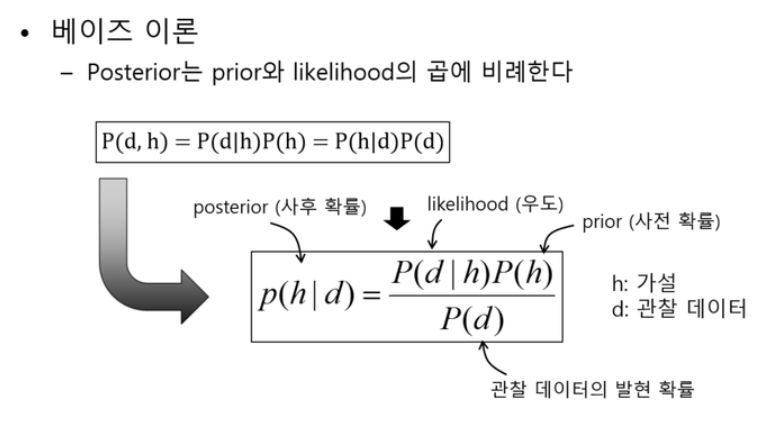


차원 축소 알고리즘인 주성분 분석(Principle Component Analysis, PCA): 분산을 최대로 보존할 수 있는 축을 선택하는 것이 정보를 가장 적게 손실할 수 있다

* **PCA에서는 분산이 최대인 축을 찾고, 이 첫번째 축에 직교하고 남은 분산을 최대한 보존하는 두번째 축을 찾습니다.**
* **혹은**  **SVD**는 매우 많은 피처를 가진 고차원 행렬을 저차원 행렬로 분리하는 행렬 분해 기법 : ∑(digonal matrix)의 0이 아닌 대각 원소값을 특이값(Singular Value)라고 하고, 행렬을 특이값(σ=√𝜆:Singular Value)과 특정한 구조로 분해하는 것

머신러닝 문제는 훈련 샘플이 각각 수천, 혹은 수백만개의 특성을 가지고 있습니다. 이렇게 특성들이 많을 경우, 유의미한 특성들을 찾기가 어려울 뿐더러 훈련을 느리게 해 결과적으로 성능 저하를 일으키는 원인이 됩니다. 이런 문제를 "차원의 저주"(curse of dimensionality)

Bias-Variance Trade-off란 Bias와 Variance는 서로의 변화에 따라 상반되게 움직인다는 것 모델이 복잡해질 수록 편향은 작아지고, 분산은 커집니다. 즉 over-fitting 됩니다. 모델이 단순해질수록 편향은 커지고, 분산은 작아집니다. 즉, under-fitting 됩니다. 무조건 편향만 줄일 수도, 무조건 분산만 줄일 수도 없습니다. 오류를 최소화하려면 편향과 분산의 합이 최소가 되는 적당한 지점을 찾아야 합니다.

1. unsupervised learning – 클러스터링/차원축소 등 **패턴/구조파악하는 것**
2. supervised – 랜덤포레스트/DT/regression/SVM
3. self-supervised models including GPT3 and BERT : 기존 데이터의 가공을 통하여 학습이 가능한 새로운 형태로 만들어 지도 학습을 진행하는 것을 의미합니다. 예를 들어, NLP에서 원래 문장의 일부 단어를 masking하고 해당 단어를 예측하는 과정
4. semi-supervised GPT: 기존 데이터의 가공을 거치지는 않으며, 라벨링이 없는 데이터의 예측 라벨 결과를 이용하여모델을 다시 학습시키는 과정
5. softmax – 스무스한 근사치, 최대값 함수에 근사하도록
6. 모델클래스정의/로스함수/옵티마이저(adam)/run gpu
7. 로그 취해서 곱셈 덧셈화 – 확률은 0to1 계속 곱하면 전체 0 가까워짐
8. 로스 fun 최소 되도록 param 최적화; 미분(가파른방향알려줌) 빼고
9. 
10. mle 우도 최대회, 즉 원하는 데이터셋을 확률을 최대화 한 파라미터 찾는법
11. nll 로그 취해서 마이너스 후 최솟값
12. 로지스틱 회귀 – 이진분류 0/1 시그모이드
13. l1 reg – 가중치절대값 더함, l2 – 가중치 제곱 더함
14. 앙상블로 오버핏막기
15. 하이퍼파라미터 튜닝은 언더피팅(에폭,러닝레이트)
16. cross entropy 확률 불확실성
17. entropy랑 다른점 – 로그 취해주는 값은 실제값 확률이 아니라 모델 예측 확률값임
18. regression – continuous variable에 대해 distribution을 볼 것(classification은 prob, 이산값)
19. regression – 즉 input(독립)에 대해 continuous 값(y, 종속)으로 매핑하는 예측, 예측하고 싶은 종속변수가 숫자일 때 보통 회귀라는 머신러닝의 방법을 사용
20. regularization – 베이지안에선 prior추가, MAP로 연결
21. regul – nll 적용해서 정리하면 l1(릿지) 2라쏘
22. val(hyper,under) loss > train loss = overfitting
23. glove – 통계정보 count based model + neural network loss 줄이는 장점
24. glove- co-occurance probability로 loss fun 정의
25. glove – 특정중심단어가 주어졌을 때 임베딩된 두 벡터의 내적이 두 단어의 동시 등장확률의 비율이 되게끔 임베딩
26. glove – 유사한 단어일수록 동시등장확률이 크단 것
27. glove – center context 서로 자유롭게 바뀔수 있어야함 “homomorphism”
28. “homomorphism” – 덧셈의 항등원 관계가 곱셈의 항등원이 됨
29. “homomorphism” – exp함수 씀
30. glove - least squered objective function & 최대값 고정(복잡성)
31. wordvector 평가 : NER 개체명인식- real task에 적용해서
32. 다의어 – 같은 벡터공간 내 다른 클러스 형성하면 그걸로 분류
33. 다의어 – 한 단어에 서로 가중치 부여(빈도기반) 후 선형 결합 통해 새로운 워드벡터생성 이를 가지고 클러스터링 분류
34. 자코비안 – 다변수함수에(<->스칼라함수) 대한 일차 미분
35. 자코비안 – 입력변수 한 개에 대해 출력변수의 가능한 모든 편미분
36. 자코비안 – 비선형을 선형변화로 근사
37. shape convention – weight dimension과 맞아 떨어지고 GD 적용
38. 체인룰 – 합성함수는 미분 곱으로 쪼개짐
39. 체인룰 – 각 차원이 무엇인지 계산해야함
40. 체인룰 – 즉 s h z 미분으로 가중치 갱신 가능
41. 왜 활성화함수? – 비선형에 적용하려고, 층 쌓으려고
42. 활성화함수 – relu; 연속변수, 회귀에 적합
43. 렐루 – 음수는 0
44. 시그모이드s – 입력값 매우 크거나 작으면 기울기0
45. back propogation = 체인룰 써서 파라미터 효율 업데이트
46. input – w – activation – soft – cross entropy Loss
47. loss 의 gradient를 미분해서 작아지는 쪽으로
48. back은 loss/weigth 에서 편미분 한걸로 시작
49. overfitting – 데이터적거나/모델이 복잡
50. regu – batch normalization / weight initialization / 앙상블(배깅/부스팅) / dropout / early stopping
51. **batch normalization** – 층으로 들어가는 입력값이, 데이터분포가 한쪽으로 쏠리지 않도록 평균/표준편차 다시 맞춤
52. batch 크면 local minimum
53. batch normalization - feature scaling – 속도개선, 범위 같도록, local minimum 방지
54. **weight initialization** – 파라미터 최적화 위해서, 기울기 소실 방지
55. weight initialization – 가우시안랜덤weight(0,1) / 사비에르(가우시안랜덤을 입력데이터수로 scale, 1/n)
56. weight initialization – 처음부터 분산 두배로 설정(위에 껀 linear함수가정) -> “HE”
57. **gradient clipping**  - SGD전에 norm 크기(gradient크기) 제한
58. **앙상블** **bagging** – 샘플을 여러 개 뽑아서 학습 후 평균내거나 통합, variance
59. **앙상블** **boosting** – 보다 어려운 문제(오답weaker)에 가중치(greater) 더 부여, bias
60. **dropout** – 뉴런 일부 끊어서 과도하게 의존하는 것 방지
61. dropout – 앙상블 비싸서
62. **early stopping** – 데이터 셋이 많이 없을 때 과적합 방지
63. early stopping – validation loss 증가 전 끊기
64. early stopping – l2-regaularization 같지만 전자는 한번만 학습 후자는 매 훈련마다 하이퍼라미터와 러닝레이트 조정
65. n-gram model -> NNM : 워드임베딩
66. window size 때문에 모든 단어 고려 불가
67. 동일한 가중치 반복사용, recurrent 순환 신경망
68. 모든 timestep에서 동일한 가중치 공유하고
69. 순차적인 input과 output의 반환이 가능
70. RNN 직전시점의 hidden에서 생성된 states를 다음 시점 input으로 전달
71. 코퍼스 – RNN – 예측값 – crossentropy – 평균 – bptt - update
72. manytoone 감정 / manytomany번역 / onetomaany 이미지캡션
73. BPTT Back Propagation Through Time
74. 여기서도 matrix 동일
75. 동일하다? 가중치 적으면 <1 : VANISHING
76. >1 : EXPLODING
77. Long Term Dependency : 다음셀로 갈수록 앞 정보 소실(ff의관점) / 기울기 소실이나 폭팍(bptt 관점)
78. lstm과 gradient clipping이 각각 해결
79. Long Short Term Memory Rnn
80. lstm : hidden + cell state(과거 save해서 gradient 살려)
81. lstm : gate(불필요한거거름)
82. gru : 더 간단, reset + update 2개
83. 혹은 DIRECT CONNECTION을 늘려서 해결!
84. Residual Learning : F(X) + X 나온결과에 해당 input 더해서 기존정보
85. dense net 모든 layer 연결
86. highwaynet : 활성화함수 없고, gate2개
87. bidirectional and multi-layer rnns
88. seq2se1 : 인코더&디코더 rosa 등장
89. sos / eos 토큰
90. seq2seq : teacher forcing : 실제값을 다음셀에 넣음(loss만 반영)
91. 여전히 gradient descent
92. BLUE : ngram정확도, 번역측정, 언어 구애안받음
93. subwords models tokenizing – BPE(unknowntoken, oov문제) : 병합 반복~희망사이즈
94. wordpiece – bert tokenizing : word embedding + character embedding
95. sentencepiece – 어떤 언어에도, 띄어쓰기를 하나의 character, pre-tokenizing 필요없음
96. seq2seq with attention 인코더의 모든 출력을 참고하자
97. 인코더 마지막 hiddenstate가 디코더로 전달 - decoder에서 만들어진 벡터들 가지고 인코더의 각 hidden state랑 내적 - attention score – attention distribution – 가중합 – attention value나오면 concat
98. 그리고 병렬이 안된다는 치명적 단점!!! 빅오
99. 쿼리(Query)인 디코더의 hidden state 벡터
100. 키(Key)인 인코더에서 넘어온 hidden state 벡터
101. 밸류 : 각 단어의 의미를 살려주기 위한 가중치 벡터, 실제값
102. 이 세개가 같으면 self-attention
103. self attention : time step 이용안하고 모든 인코더 input 활용 가능 즉!! 장기의존성문제!
104. 인코더 블록은 2개의 sub-layer(Multi-Head(self) Attention, Feed Forward)
105. 디코더 블록은 3개의 sub-layer(Masked Multi-Head(self) Attention, Multi-Head (Encoder-Decoder) Attention, Feed Forward)
106. 각 층 사이에선 subblock, add&norm 반복
107. mutl head self attention : 내적&scale : atten-score – softmax\*v : atten-value – concat \* weight(차원일치)
108. add&normarlization – redisual connection & layer normalization
109. Feed Forward Network : 은닉층의 차원이 늘어났다가 다시 원래 차원으로 줄어드는 단순한 2층 신경망이다. 활성화 함수로 Relu를 사용한다.

**Masked Self-Attention(look-ahead mask)**

1. 디코더 블럭에서 사용되는 특수한 Self-Attention이다.  
   디코더는 Auto Regressive하게 단어를 생성하기 때문에 타깃 단어 이후 단어를 보지 않고 단어를 예측해야 한다.  
   따라서 타깃 단어 뒤에 위치한 단어는 Self-Attention에 영향을 주지 않도록 마스킹(Masking)을 해주어야 한다.  
   그렇기 때문에 Softmax를 취해주기 전에 가려주고자 하는 요소에만 −∞에 해당하는 매우 작은 수를 더해준다.

**Encoder-Decoder Attention**

1. 디코더에서 Masked Self-Attention 층을 지난 벡터는 Encoder-Decoder Attention 층으로 들어가게 되는데 좋은 번역을 위해서 번역할 문장과 번역되는 문장의 정보를 엮어주는 역할을 하는 것이 Encoder-Decoder Attention이다.  
   디코더 블록의 Masked Self-Attention으로부터 출력된 벡터를 쿼리(q) 벡터로 사용한다. 키(k)와 밸류(v) 벡터는 최상위(6번째)인코더 블록에서 사용했던 값을 그대로 가져와서 사용하게 된다. 계산 과정은 Self-Attention과 동일하다.

**Linear & Softmax Layer**

1. 디코더의 최상층을 통과한 벡터들은 Linear층을 지난 후 Softmax를 통해 예측할 단어의 확률을 구하게 된다.
2. pretraing – model 전체를 train하기 위해
3. 파라미터 초기화 해서 finetuning 때 조정하면됨
4. labeled data 필요없음
5. encoder decoder : bart/t5
6. RoBERTa 더 긴 시퀀스
7. SpanBERT : 마스킹 연속적
8. NLG을 위해 디코더에서 알고리즘 쓰임 : greedy decoding / beamsearch / random / top-k / top-p
9. beam search : 여러 가설 중 가장 높은 확률
10. 그런데 실제로 사람은 어디로 튈지 모름 단순 확률 큰거 하나만 보기엔 무리
11. top-k : 높은 k개 (flat하거나 특정 token만 high 할수도)
12. top-p : 누적확률 p에 속하는 상위
13. 여전한 문제는 train에선 teacher forcing이 불가능해서 노답될 수도 bias!!
14. meta-learning/in context learning : learn-to-learn
15. learn-to-learn : 학습 과정에서 스킬과 패턴인식 능력 키워서 inference 단계에서
16. meta-learning : no gradient update
17. good prompt : k-nearst neighbors
18. calibration 보정 : 분포가 1:1로 나와야되는거 추정해서 softmax취해서 0으로 uniform하게
19. zero-shot CoT Chain of Thinking : trigger문장을 줘서 복잡한 inference task 따로 설명 안해도 됨
20. zero-shot CoT Chain of Thinking : few shot 없이!!
21. zero-shot CoT : let’s think step by step
22. Instruction Finetuning : Aligning 문제, zeroshot 향상시키기 위헤 FLAN
23. Aligning : 의도대로 안나옴 왜냐면 자연어는 다르니까
24. Instruction Finetuning : 그래서 직접 instruction,ouput 데이터 셋을 모아서 finetuning에 쓰임
25. Instruction Finetuning : unseen task 문제!!
26. ChatGPT, InstructGPT : RLHF
27. RLHF : Alignment 문제 -> human feedback 사용하자
28. RLHF : prompt,response 쌍 가지고 3으로 finetuning -> 그 model ouput을 사람이 평가해서 선호도 측정함. 이것을 가지고 reward model 학습, reward는 scalar -> RM을 function으로 활용하고 강화학습 PPO알고리즘(rewardloss미분위함)으로 fine-tuning(비싸서 파라미터 frozen)
29. RLHF : KL penalty : 원래 모델로부터 크게 벗어나지 않도록 reward function에 더해줌
30. 그니까 지도학습/reward model학습/강화학습
31. 데이터셋불러와서 – 토크나이징 – 모델불러오고 – 필요하면 트레인하고 – config 관리