Loss function

classifier가 얼마나 좋은지 알려주는 function.

- loss function을 통해 가중치의 가치를 평가
- loss function을 최소화하는 W를 찾는 것이 목표.
- classifier와 loss function은 반비례 관계. (objective function, cost function)
- classifier와 - loss function 인 경우도 가끔 사용 (reward function, profit function...)

https://kmiiiaa.tistory.com/6

아래와 같은 데이터셋 인경우

$(x_i, y_i)_i^N = 1$

xi는 이미지

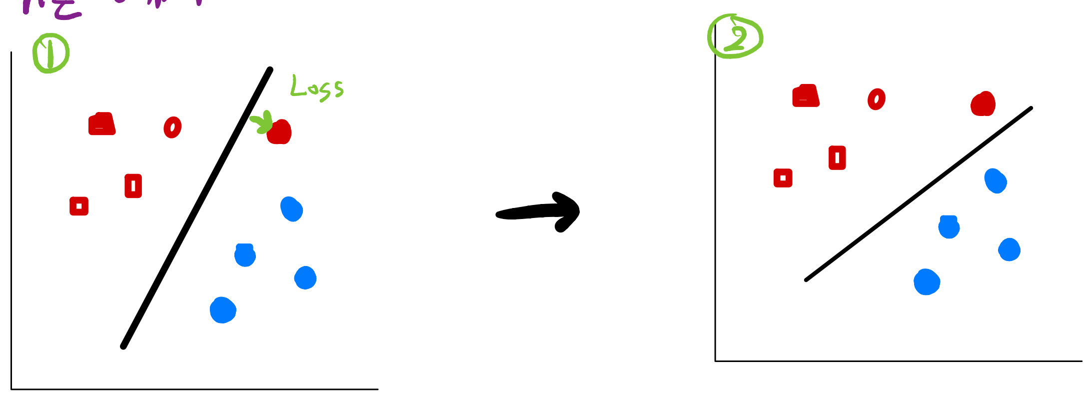
yi는 라벨

loss function

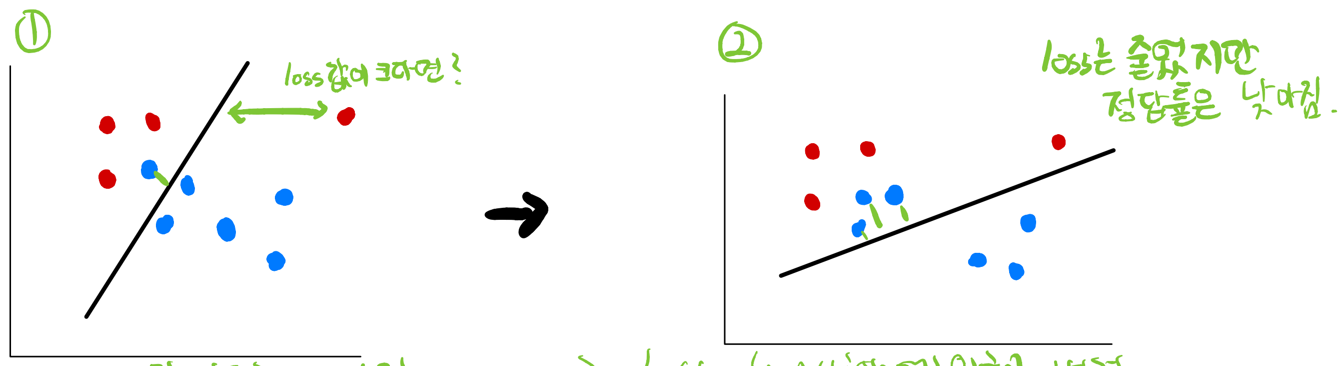
$L_i(f(x_i, W), y_i)$

$L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$ Loss의 평균

기본 예시



극단적 예시



더 잘 분류한다고 가정 ⟶ Loss function에 의해 변경

Regularization에 의해 잡을 수 있다.

여러 W중 W를 선택하기 위한 다른 메커니즘 (오버피팅 방지)

Regularization

모델이 한 쪽으로 지나치게 치우치는 것을 방지..

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

simple example

L1 regularization

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

Elastic net(L1+L2)

$$\lambda R(W) = \lambda_1 \sum_k \sum_l W_{k,l}^2 + \lambda_2 \sum_k \sum_l |W_{k,l}|$$

$\lambda_1 = 1,\ \lambda_2 = 0 \Rightarrow L_2$

$\lambda_1 = 0,\ \lambda_2 = 1 \Rightarrow L_1$

traing error를 줄이며 overfitting을 방지하고 곡률을 더하여 최적화 시킨다.
(그래프 설명)

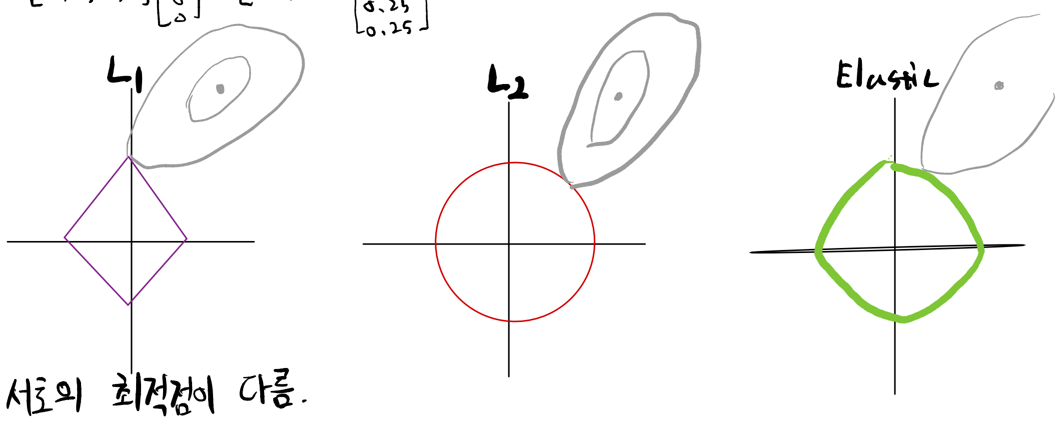
$x = [1,1,1,1]$

$w_1 = [1,0,0,0]$ $\quad w_2 = [0.25, 0.25, 0.25, 0.25]$

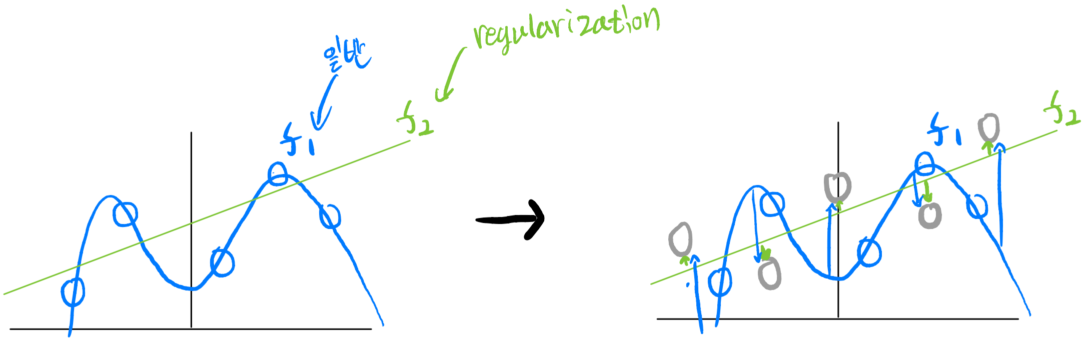
↳ L_2는 가중치 확산을 좋아함.

$w_1^T x = w_2^T x = 1$ 동일한 1 추출

$$[1,1,1,1]\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = [1,1,1,1]\begin{bmatrix}0.25\\0.25\\0.25\\0.25\end{bmatrix} = 1$$



서로의 최적점이 다름.

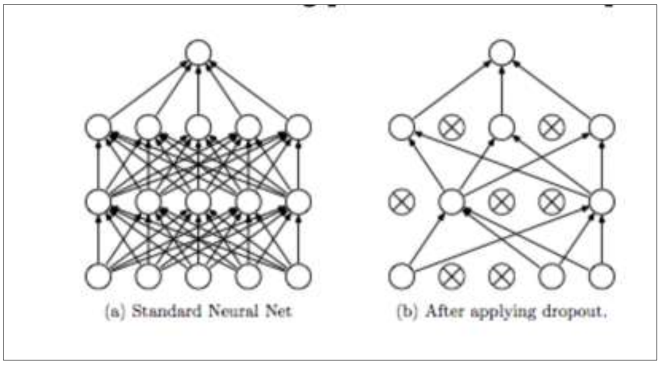


f_2가 새로운 데이터에 대해서 더 강력함.

More complex

Drop out

위 그림에서 왼쪽 그림과 같은 모델에서 몇개의 연결을 끊어서, 즉 몇개의 노드를 죽이고 남은 노드들을 통해서만 훈련을 하는 것입니다.
이때 죽이는, 쉬게하는 노드들을 랜덤하게 선택합니다.
https://doorbw.tistory.com/147



Batch Normalization

각 레이어마다 정규화 하는 레이어를 두어, 변형된 분포가 나오지 않도록 조절하게 하는 것이 배치 정규화이다.

```
# 모델은 스퀀스
model_deep_lstm = Sequential()
#1번 레이어
model_deep_lstm.add(LSTM(64, return_sequences=True,input_shape = (timesteps, input_dim)))
model_deep_lstm.add(Dropout(abs(np.random.normal(0,1))))
model_deep_lstm.add(BatchNormalization())
#2번레이어
model_deep_lstm.add(LSTM(32,return_sequences=True))
model_deep_lstm.add(Dropout(abs(np.random.normal(0,1))))
model_deep_lstm.add(BatchNormalization())
#3번 레이어
model_deep_lstm.add(LSTM(16))
model_deep_lstm.add(Dropout(abs(np.random.normal(0,1))))
model_deep_lstm.add(BatchNormalization())
#4번레이어
model_deep_lstm.add(Dense(n_classes, activation = 'softmax'))
model_deep_lstm.summary()
```

- 단순 문제에서는 오히려 안좋은 성능을 보일 수 있음.

cutout, Mixup, Stochastic depth...

Cross-Entropy Loss(Multinomial Logistic Regression)
두 확률 분포의 차를 구하는 함수.

1. Entropy
불확실성의 척도

$$H(x) = -\sum_{i=1}^{n} p(x_i)\log p(x_i)$$

1.1 예시 동전 던지기 & 주사위 던지기

- 동전 던졌을 때 앞/뒷면이 나올 확률 모두 1/2

$$H(x) = -\left(\frac{1}{2}log\frac{1}{2} + \frac{1}{2}log\frac{1}{2}\right) = 0.693$$

- 주사위를 던졌을 때, 각 6면이 나올 확률을 모두 1/6

$$H(x) = -\left(\frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6}\right) = 1.79$$

1.2 다른 예시
- 가방 안에 빨간 공만 있을 경우 무조건 빨간 공을 뽑기 때문에 불확실성은 0.
- 가방 안에 빨간 공과 검은 공이 50:50으로 있다면 entropy는 가장 크다.
ex)
빨간 공 10인 경우 = 0
빨간 공 3 검은 공 7인 경우, 빨간 공 7 검은 공 3인 경우

$$-\left(\frac{3}{10}log\frac{3}{10} + \frac{7}{10}log\frac{7}{10}\right) = 0.61$$

빨간 공 5 검은 공 5인 경우

$$-\left(\frac{1}{2}log\frac{1}{2} + \frac{1}{2}log\frac{1}{2}\right) = 0.69$$

http://melonicedlatte.com/machinelearning/2019/12/20/204900.html
https://3months.tistory.com/436

로그를 사용하는 이유?
- 엔트로피를 정의할 때 곱셈으로 늘어나는 경우의 수를 로그의 성질을 통해 덧셈으로 바꿔주기 위해
- log에 2나 자연로그를 취하는 이유는 상수e가 가장 많은 값을 표현하게 해주기 때문.
[https://ramees.tistory.com/64]

https://wordbe.tistory.com/entry/ML-Cross-entropyCategorical-Binary%EC%9D%98-%EC%9D%B4%ED%95%B4

2. 활성화 함수 (Activation function)

2-1 softmax란?

클래스의 스코어가 각각 0~1 값으로 반환하며, 모든 클래스 스코어의 합은 1이 된다.
가장 큰 출력 값을 받은 클래스가 가장 높은 확률 값을 가진다.

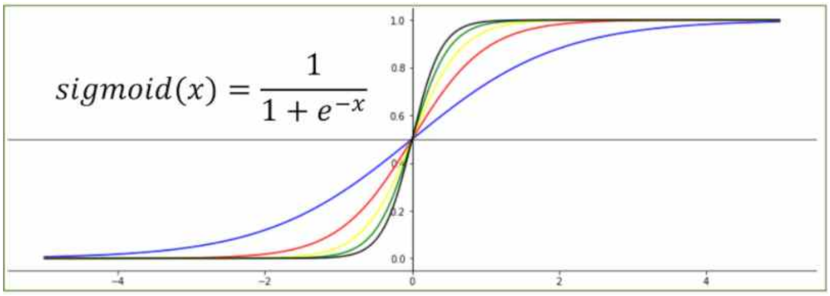
$$f(s_i) = \frac{e^{s_i}}{\sum_{j}^{c} e^{s_j}}$$

https://m.blog.naver.com/wideeyed/221021710286

2-2 Sigmoid란?

- 0~1사이의 값으로 미분가능한 수로 변환
- x가 음수인 경우 0에 가깝게 표현하여 입력값이 최종 계층에서 미치는 영향이 작아짐.

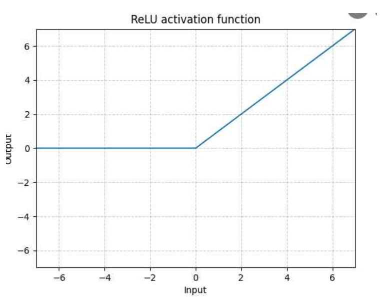
http://taewan.kim/post/sigmoid_diff/



2-3 Relu 란?

- 보통 hidden layer에 사용하며 x값이 0보다 작으면 모두 0을 반환하고 0보다 크면 그대로 출력

https://medium.com/@kmkgabia/ml-sigmoid-%EB%8C%80%EC%8B%A0-relu-%EC%83%81%ED%99%A9%EC%97%90-%EB%A7%9E%EB%8A%94-%ED%99%9C%EC%84%B1%ED%99%94-%ED%95%A8%EC%88%98-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0-c65f620ad6fd



3. Cross Entropy Function

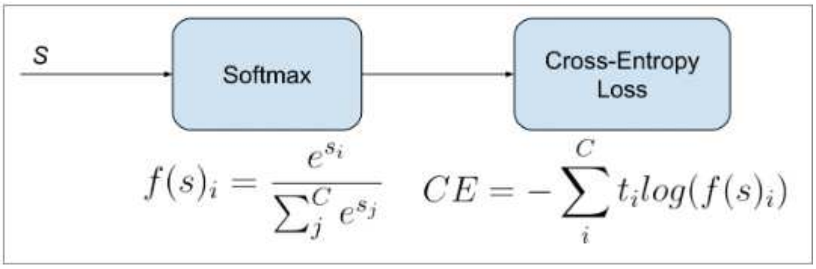
두 확률 분포의 차를 구하는 함수.

$$L_i = -\log P(Y = y_i | X = x_i)$$

$$CE = -\sum_{i=1}^{C} (y_i \log(s_i) + (1-y_i)\log(1-s_i))$$

3-1 Categorical Cross-Entropy Loss

Softmax activation 뒤에 Cross-Entropy loss를 붙인 형태



ex)

cat을 맞추는 문제

		e		normalize	Kullback-Leibler divergence		
cat	3.2	->	24.5	->	0.13	->compare<-	1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

→아래 있음.

one Hot 인코딩.

$x_1 = [1,0,0]$

$x_2 = [0,1,0]$

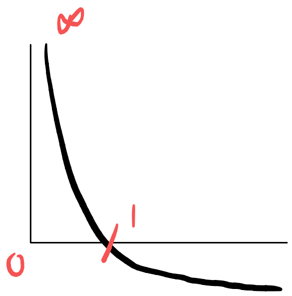
$x_3 = [0,0,1]$

$$\begin{bmatrix} 3.2 \\ 5.1 \\ -1.7 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} e^{3.2} \\ e^{5.1} \\ e^{-1.7} \end{bmatrix} = \begin{bmatrix} 24.5 \\ 164.0 \\ 0.18 \end{bmatrix} \xrightarrow{nor} \begin{bmatrix} \frac{24.5}{188.68} \\ \frac{164}{188.68} \\ \frac{0.18}{188.68} \end{bmatrix} = \begin{bmatrix} 0.13 \\ 0.87 \\ 0.00 \end{bmatrix}$$

sum=0

$$\begin{aligned} CE = & -(1\log 0.13 + (1-1)\log(1-0.13)) \quad \text{cat} \quad \to 0 \\ & -(0\log 0.13 + (1-0)\log(1-0.87)) \quad \text{car} \quad 0 \leftarrow \\ & -(0\log 0 + (1-0)\log(1-0)) \quad \text{frog} \quad 0 \leftarrow \\ = & -(\log 0.13 + \log 0.13) = 4.08 \end{aligned}$$

−log함수



x값이 작을 수록 y값이 매우 커짐.

3-2 Sparse_Categorical Cross-Entropy Loss

class 값이 정수인 경우

ex) 가위바위보 분류 문제

```python
#데이터를 불러와서 라벨 및 train 데이터를 저장해주는 함수
def load_data(img_path, number_of_data=300):  # 가위바위보 이미지 개수 총합에 주의
    # 가위 : 0, 바위 : 1, 보 : 2
    img_size=28
    color=3
    #이미지 데이터와 라벨(가위 : 0, 바위 : 1, 보 : 2) 데이터를 담을 행렬(matrix) 영역을 생성합니다.
    imgs=np.zeros(number_of_data*img_size*img_size*color,dtype=np.int32).reshape(number_of_data,img_size,img_size,color)
    labels=np.zeros(number_of_data,dtype=np.int32)

    idx=0
    for file in glob.iglob(img_path+'/scissor/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=0   # 가위 : 0
        idx=idx+1

    for file in glob.iglob(img_path+'/rock/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=1   # 바위 : 1
        idx=idx+1

    for file in glob.iglob(img_path+'/paper/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=2   # 보 : 2
        idx=idx+1

    print("학습데이터(x_train)의 이미지 개수는", idx,"입니다.")
    return imgs, labels
```

```python
#모델 구축
model=keras.models.Sequential()
model.add(keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,3)))
model.add(keras.layers.MaxPool2D(2,2))
model.add(keras.layers.Conv2D(64, (3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D((2,2)))
model.add(keras.layers.Conv2D(16, (3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D((2,2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dense(3, activation='softmax'))
```

```python
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

→ class가 0, 1, 2로 분류

$x_1=0$
$x_2=1$
$x_3=2$

```python
#모델 적용
#이때 earlystopping과 검증셋이 하이퍼파라미터로 들어간다.
# epochs를 아무리 높게 하더라도 얼리스탑 함수를 통해 가장 좋은 정확도일때 자동으로 멈춰준다.
model.fit(x_split_train, y_split_train, epochs=50, batch_size=10,callbacks=[es], validation_data=(x_split_val,  y_split_val))
```

```
Epoch 1/50
216/216 [==============================] - 1s 4ms/step - loss: 1.0994 - accuracy: 0.3421 - val_loss: 1.0966 - val_accuracy: 0.4000
Epoch 2/50
216/216 [==============================] - 1s 4ms/step - loss: 1.0270 - accuracy: 0.4861 - val_loss: 0.8399 - val_accuracy: 0.6704
Epoch 3/50
```

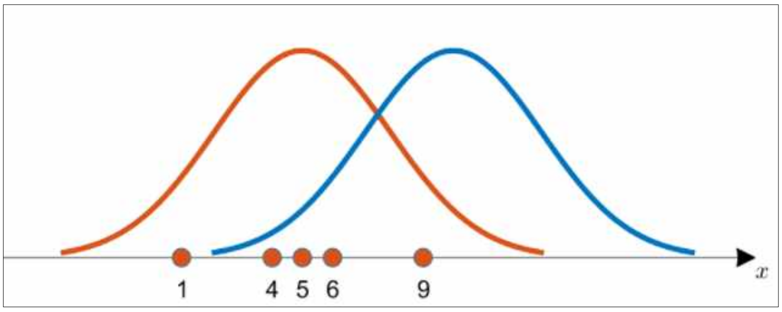
```python
#모델 잘만들었는지 평가해보기
test_loss, test_accuracy = model.evaluate(x_test_norm,y_test, verbose=2)
print("test_loss: {} ".format(test_loss))
print("test_accuracy: {}".format(test_accuracy))
```

```
10/10 - 0s - loss: 1.7787 - accuracy: 0.6367
test_loss: 1.7786505222320557
test_accuracy: 0.6366666555404663
```

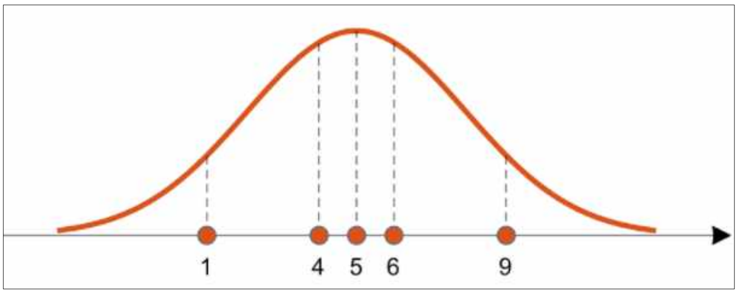
+ 최대우도법(Maximum Likelihood Estimation)

모수적인 데이터 밀도 추정 방법으로써 파라미터 a=(a1, a2,..an)으로 구성된 어떤 확률밀도 함수 P(x|a)에서 관측된 표본 데이터 집합을 x = (x1,x2,..,xn)이라 할 때, 표본들의 파라미터 a=(a1, a2,..an)을 추정하는 방식

- 아래 그림에서 주황선과 파란 선이 있다고 했을 때 우리는 주황색이 중심에 더 일치하다고 판단 할 것이다. 추출된 데이터로부터 분포의 특징을 파악하고 추정할 수 있다.



수학적인 추정방법을 언급하기 위해 likelihood 기여도를 보자



수치적으로 이 가능도를 계산하기 위해서는 각 데이터 샘플에서 후보 분포에 대한 높이(즉, likelihood 기여도)를 계산해서 다 곱한 것을 이용할 수 있을 것이다.

계산된 높이를 더해주지 않고 곱해주는 것은 모든 데이터들의 추출이 독립적으로 연달아 일어나는 사건이기 때문이다.

likelihood function

$$P(x|a) = \Pi_{k=1}^{n} P(x_k|a)$$

보통 로그를 취하여 계산을 한다.

로그를 취하는 이유는 곱을 합으로 변환하기도 하며 계산에 편리함 때문

log-likelihood function

$$L(a|x) = \log P(x|a) = \sum_{i=1}^{n} \log P(x_i|a)$$

likelihood function의 최대값을 찾는 방법(편미분...)

https://angeloyeo.github.io/2020/07/17/MLE.html

Kullback-Leibler divergence

두 확률분포의 차이를 계산하는 데에 사용하는 함수로, 어떤 이상적인 분포에 대해, 그 분포를 근사하는 다른 분포를 사용해 샘플링을 한다면 발생할 수 있는 정보 엔트로피 차이를 계산한다.