학습이 잘 되는 모델

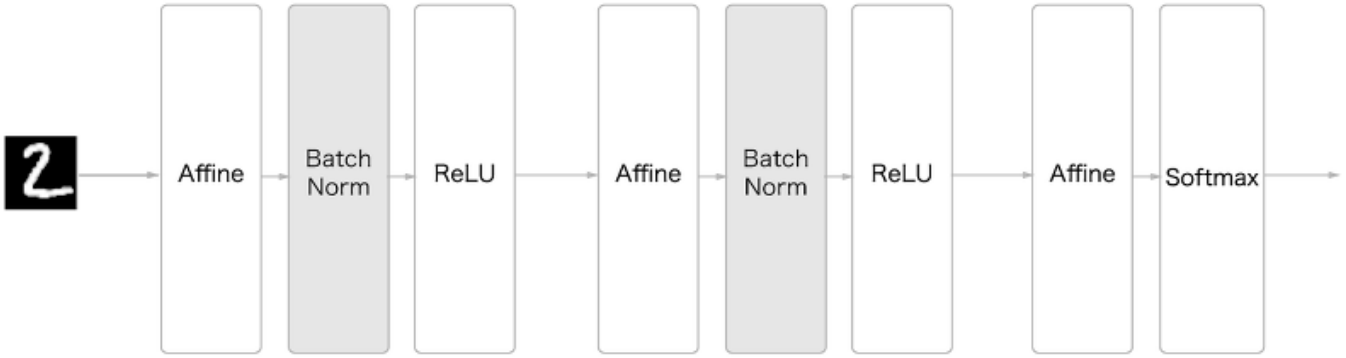
조금더 학습이 잘되는 인공신경망을 만드는 방법을 배우도록 하겠습니다. 간단한 방법이 있습니다. 바로 배치정규화입니다.

배치 정규화는 2015년에 제안된 방법이지만, 많은 연구자와 기술자들이 사용하고 그 효과가 입증된 방법입니다. 배치 정규화를 이용하는 이유들은 다음과 같습니다.

- 학습을 빨리 진행할 수 있다. (학습 속도 개선)
- 초기값 에 크게 의존하지 않는다. (골치 아픈 초깃값 선택 장애를 겪지 않아도 됨)
- 오버피팅을 억제한다. (드롭아웃 등의 필요성 감소)

배치 정규화란?

그럼 배치 정규화의 기본 아이디어를 알아봅시다. 배치 정규화는 각 층에서의 활성화값이 적당히 분포되도록 조정하는 것을 목표로 합니다. 그래서 데이터 분포를 정규화하는 '배치 정규화 Batch Norm 계층'을 신경망에 삽입해서 이용합니다.



학습 시 미니배치를 단위로 정규화하는 방식을 사용합니다.데이터 분포가 평균이 0, 분산이 1이 되도록 정규화하고 수식으로 나타내면 다음과 같습니다.

$$B = \{x_1, x_2, x_3, \ldots, x_m\}$$

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

m개의 입력 데이터의 집합에 대한 평균

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

분산

$$\widehat{x}_i \leftarrow \frac{(x_i - \mu_B)}{\sqrt{\sigma_B^2 + \varepsilon}}$$

평균이 0, 분산이 1이 되게 정규화

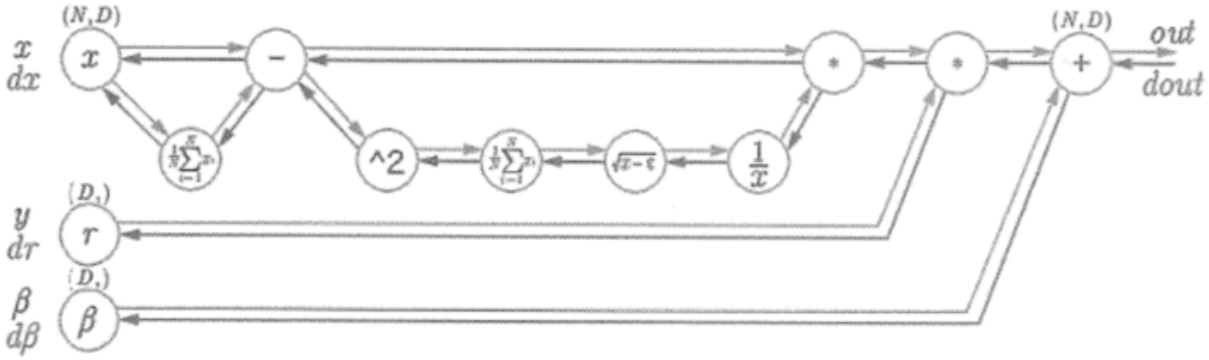
미니배치 B 라는 m 개의 입력 데이터의 집합에 대해
평균 (μ)과 분산 (σ^2)을 구하고,
입력 데이터를 평균이 0, 분산이 1이 되게 정규화합니다.
ε 을 10e-7과 같이 작은 값으로 두어, 0으로 나누는 일이 없도록 합니다.

위에서 마지막 식은 단순히 입력 데이터 x_i를 $\hat{X}$ 로 변환하는 일을 합니다. 이 처리를 활성화 함수의 앞(혹은 뒤)에 삽입해서 데이터 분포가 덜 치우치게 할 수 있습니다. 또, 배치 정규화 계층마다 이 정규화된 데이터에 고유한 확대 scale와 이동 shift 변환을 수행합니다. 수식은 아래와 같습니다.

$$y_i \leftarrow \gamma \hat{x}_i + \beta$$

γ (그리스어: Γάμμα 감마)는 확대를, β 는 이동을 담당합니다. 두 값은 처음에는
γ = 1, β = 0부터 시작하고, 학습하면서 적합한 값으로 조정합니다.
γ = 1은 1배 확대, β = 0은 이동하지 않음을 의미합니다. 즉, 원본 그대로 시작한다는 의미입니다.

여기까지가 배치 정규화의 알고리즘이고, 이 알고리즘이 신경망에서 순 전파 때 적용됩니다. 이를 그래프로 표현하면 다음과 같습니다.



배치 정규화의 계산 그래프

- 사용할 레이어
  - tf.keras.layers.BatchNormalization()
  - tf.keras.layers.Activation('swish')

```python
# -*- coding: utf-8 -*-
import tensorflow as tf
import pandas as pd

파일경로 = './data/boston.csv'
보스턴 = pd.read_csv(파일경로)

독립 = 보스턴[['crim', 'zn', 'indus', 'chas', 'nox',
               'rm', 'age', 'dis', 'rad', 'tax',
               'ptratio', 'b', 'lstat']]
종속 = 보스턴[['medv']]
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[13])
H = tf.keras.layers.Dense(8, activation='swish')(X)
H = tf.keras.layers.Dense(8, activation='swish')(H)
H = tf.keras.layers.Dense(8, activation='swish')(H)
Y = tf.keras.layers.Dense(1)(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='mse')

X = tf.keras.layers.Input(shape=[13])

H = tf.keras.layers.Dense(8)(X)
H = tf.keras.layers.BatchNormalization()(H)
H = tf.keras.layers.Activation('swish')(H)

H = tf.keras.layers.Dense(8)(H)
H = tf.keras.layers.BatchNormalization()(H)
H = tf.keras.layers.Activation('swish')(H)

H = tf.keras.layers.Dense(8)(H)
H = tf.keras.layers.BatchNormalization()(H)
H = tf.keras.layers.Activation('swish')(H)
```

```
35
36  Y = tf.keras.layers.Dense(1)(H)
37  model = tf.keras.models.Model(X, Y)
38  model.compile(loss='mse')
39
40  model.fit(독립, 종속, epochs=1000)
41
42  파일경로 = './data/iris.csv'
43  아이리스 = pd.read_csv(파일경로)
44
45  아이리스 = pd.get_dummies(아이리스)
46
47  독립 = 아이리스[['꽃잎길이', '꽃잎폭', '꽃받침길이', '꽃받침폭']]
48  종속 = 아이리스[['품종_setosa', '품종_versicolor', '품종_virginica']]
49  print(독립.shape, 종속.shape)
50
51  X = tf.keras.layers.Input(shape=[4])
52  H = tf.keras.layers.Dense(8, activation='swish')(X)
53  H = tf.keras.layers.Dense(8, activation='swish')(H)
54  H = tf.keras.layers.Dense(8, activation='swish')(H)
55  Y = tf.keras.layers.Dense(3, activation='softmax')(H)

56  model = tf.keras.models.Model(X, Y)
57  model.compile(loss='categorical_crossentropy',
58                metrics=['accuracy'])
59
60  X = tf.keras.layers.Input(shape=[4])
61
62  H = tf.keras.layers.Dense(8)(X)
63  H = tf.keras.layers.BatchNormalization()(H)
64  H = tf.keras.layers.Activation('swish')(H)
65
66  H = tf.keras.layers.Dense(8)(H)
67  H = tf.keras.layers.BatchNormalization()(H)
68  H = tf.keras.layers.Activation('swish')(H)
69
70  H = tf.keras.layers.Dense(8)(H)
71  H = tf.keras.layers.BatchNormalization()(H)
72  H = tf.keras.layers.Activation('swish')(H)
73
74  Y = tf.keras.layers.Dense(3, activation='softmax')(H)
75  model = tf.keras.models.Model(X, Y)
76  model.compile(loss='categorical_crossentropy',
77                metrics=['accuracy'])
78
79  model.fit(독립, 종속, epochs=1000)
```