【要点まとめ】 過学習とは、訓練サンプルにだけ適合してしまった状態。 本来の目的である汎化性能が得られないため、モデルとしては失敗。

過学習を抑制する方法について、以下が挙げられる。

- （パラメータ）正則化 →L1、L2正則化、ElasticNet
- 正則化レイヤー →Dropout
- 正規化レイヤー →Batch正規化、Layer正規化、Instance正規化

【実装演習】 Dropoutの比較検証を写経する

```
def create_model(input_shape, class_num, is_use_dropout,
dropout_rate=0.1):
    if is_use_dropout:
        model = tf.keras.models.Sequential([
            tf.keras.layers.Conv2D(32, 3, padding='same',
input_shape=input_shape[1:], activation='relu'),
            tf.keras.layers.Conv2D(32, 3, activation='relu'),
            tf.keras.layers.MaxPooling2D(),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Dense(class_num, activation='softmax'),
        ])
    else:
        model = tf.keras.models.Sequential([
            tf.keras.layers.Conv2D(32, 3, padding='same',
input_shape=input_shape[1:], activation='relu'),
            tf.keras.layers.Conv2D(32, 3, activation='relu'),
            tf.keras.layers.MaxPooling2D(),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dense(class_num, activation='softmax'),
        ])

    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['acc'])

    return model

epochs = 10
batch_size = 256

is_use_dropouts = [
                False,
                True
]

tf.random.set_seed(0) # 再現性を高めるために乱数シードを固定しています。
histories = []
for is_use_dropout in is_use_dropouts:
    model = create_model(x_train.shape, 10, is_use_dropout, 0.3)
```

```
    history = model.fit(x_train, y_train,
                        validation_data=(x_test, y_test),
                        batch_size=batch_size, epochs=epochs)
    histories.append(history)
```

Dropoutの比較検証

```
import pandas as pd
import seaborn as sns

fig = plt.figure(figsize=(10, 20))

for i, history in enumerate(histories):
    ax = fig.add_subplot(4, 1, i + 1)

    plot_df = pd.DataFrame(history.history)[['loss', 'val_loss']]
    sns.lineplot(data=plot_df, ax=ax)
    ax.set_title(f'is_use_dropout: {is_use_dropouts[i]}')
```

【実装演習考察】 Dropoutの効果について、実際にlossの比較を見ることで理解できました。 ライブラリを使用すると今回のように簡単に実装できてしまいますが、内部の動きを理解しつつ学習を進めていきたいと思います。

【自己学習】 参考書籍：ゼロからつくるDeepLearning 機械学習ではアンサンブル学習がよく使われる。アンサンブル学習とは、複数のモデルを個別に学習させ、推論時にその複数の出力の 平均値を取るというもの。 これはDropoutと近い関係がある。 Dropoutは学習時にニューロンをランダムに消去するが、これは毎回異なるモデルを学習させていると解釈できるから。 つまりDropoutは、アンサンブル学習と同じ効果を擬似的に単一のネットワークで実現していると捉えることもできる。