【要点まとめ】 訓練誤差とテスト誤差で、学習曲線が乖離すること。 ※訓練に特化して学習が進んでしまい、本番相当の別データでは全く結果を出せないこと。

対策として以下のようなものがある。 ・実際に学習させる際には、過学習が始まってしまう（誤差グラフが上向いてきた）タイミングで訓練を停止する ・小さな入力には小さなNNを使う ・小さな入力にはノード数も少なくする ・正則化（NNの自由度を削ぐ） →L1（ラッソ）、L2（リッジ）正則化、ドロップアウト

【実装演習】 正則化のサンプルコードを写経する

```
# 正則化（L1、L2）
weight_decay_lambda = 0.05
weight_decay = 0

weight_decay += weight_decay_lambda * np.sum(np.abs(network.params['W' +
str(idx)]))
loss = network.loss(x_batch, d_batch) + weight_decay

# 正則化（ドロップアウト）
class Dropout:
    def _init_(self, dropout_ratio = 0.5):
        self.dropout_ratio = dropout_ratio
        self.mask = NONE

    def forward(self, x, train_flg = True):
        if train_flg:
            self.mask = np.random.rand(*x.shape) > self.dropout_ratio
            return x * self.mask
        else:
            return x * (1.0 - self.dropout_ratio)

    del backward(self, dout):
        return dout * self.mask
```

※ 参考資料：2_5_overfiting.ipynb、ゼロからつくるDeepLearning

【実装演習考察】 L1、L2ノルムの実装ではweight_decay（荷重減衰：大きな重みにはペナルティ）という手法があります。 別の手法…ドロップアウトではランダムにノードを消去していくということを学びました。コード上初見では分かりづらかったのですが、自己学習にて理解を深めることができました。

【自己学習】 ドロップアウトは、モデルが複雑になってきてweight_decayだけでは改善が難しくなってきた場合に検討される。 self.maskに消去するニューロンをFalseで格納（Reluと考え方は同じで、dropout_ratioより大きな要素はそのままの値が入る）される動きとなる。 forwardにて発火した場合、backwardでもそのまま通され、forwardで発火しなかった場合はbackwardでも通されない。