PythonとKerasによるディープラーニング

第4章 機械学習の基礎
4.4　過学習と学習不足

19nm715n ZHANGYIHANG

機械学習に過学習はつきものであり、過学習に対処する方法を理解することが不可欠です。機械学習の根本的な課題は最適化と汎化の緊張にあります。

- **最適化**

  訓練データでの性能をできるだけ高めるためにモデルを調整するプロセスのことであり、「機械学習」の「学習」に当たります。

- **汎化**

  学習済みのモデルを全く新しいデータに適用した時の性能がどれくらい良いかを表します。

訓練データでの損失値が小さければ小さいほど、テストデータでの損失値も小さくなるのモデルは、まだ学習が十分ではなく、学習不足と呼ばれます。訓練データを増やすは、学習不足問題の最善策です。過学習をこのようにして解決するプロセスを**正則化**と呼びます。

4.4.1　ネットワークのサイズ削減する

　過学習を回避するための最も単純な方法はモデルのサイズを小さくすることです。モデルの学習可能なパラメータの数を**キャパシティ**と呼びます。パラメータの数が多いほどモデルの記憶容量が増え、訓練サンプルとそれらの目的値との写像をまるで辞書のように学習できることがわかります。
　「キャパシティが多すぎる」と「キャパシティが十分ではない」の間で妥協点を探る必要があります。これをはじき出す方法は実践してみなければなりません。

- 元のモデル

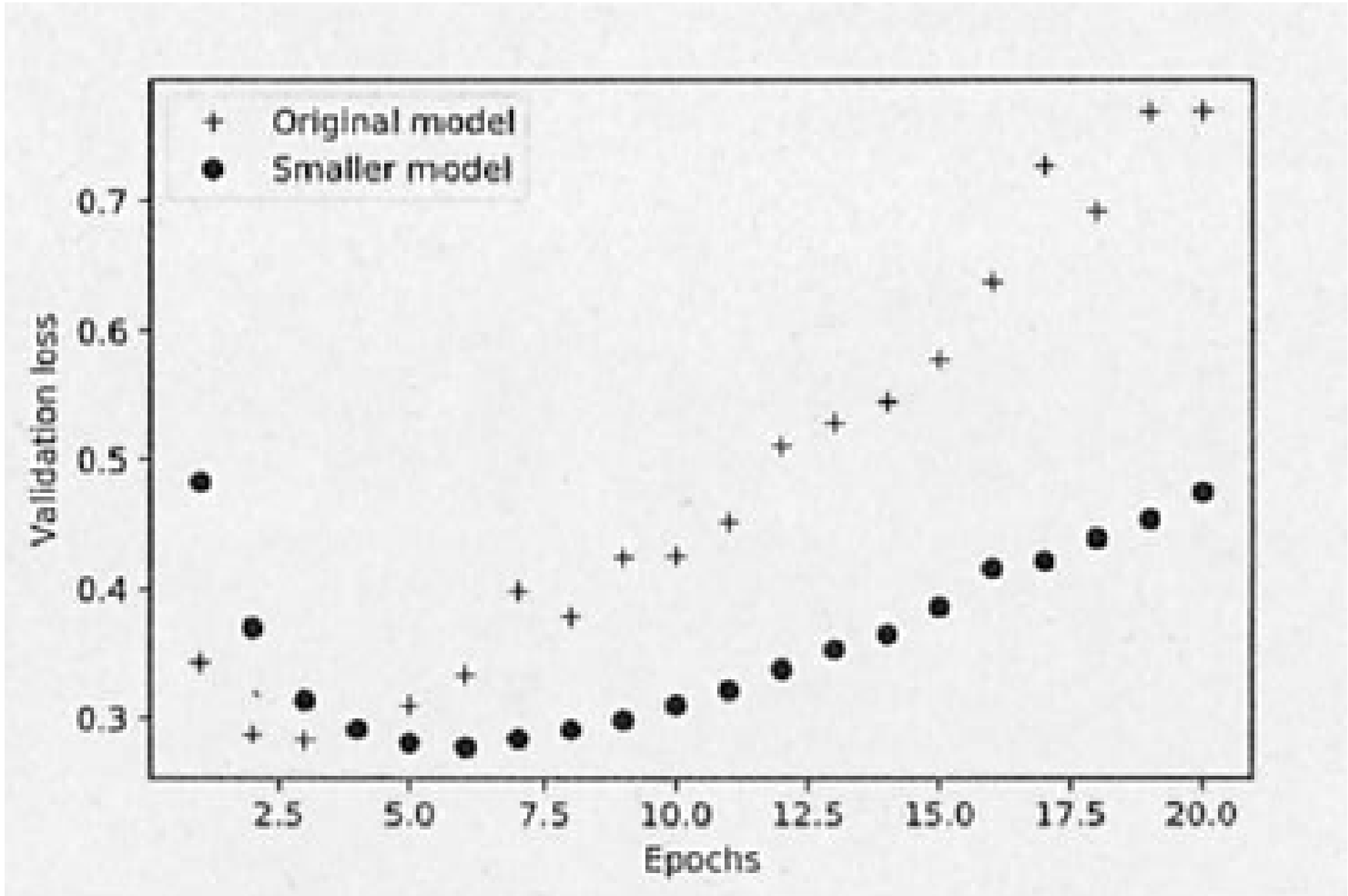
```
from keras import models
from keras import layers

model=models.Sequential()
model.add(layers.Dense(16,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

- キャパシティが小さいモデル

```
model=models.Sequential()
model.add(layers.Dense(4,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(4,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

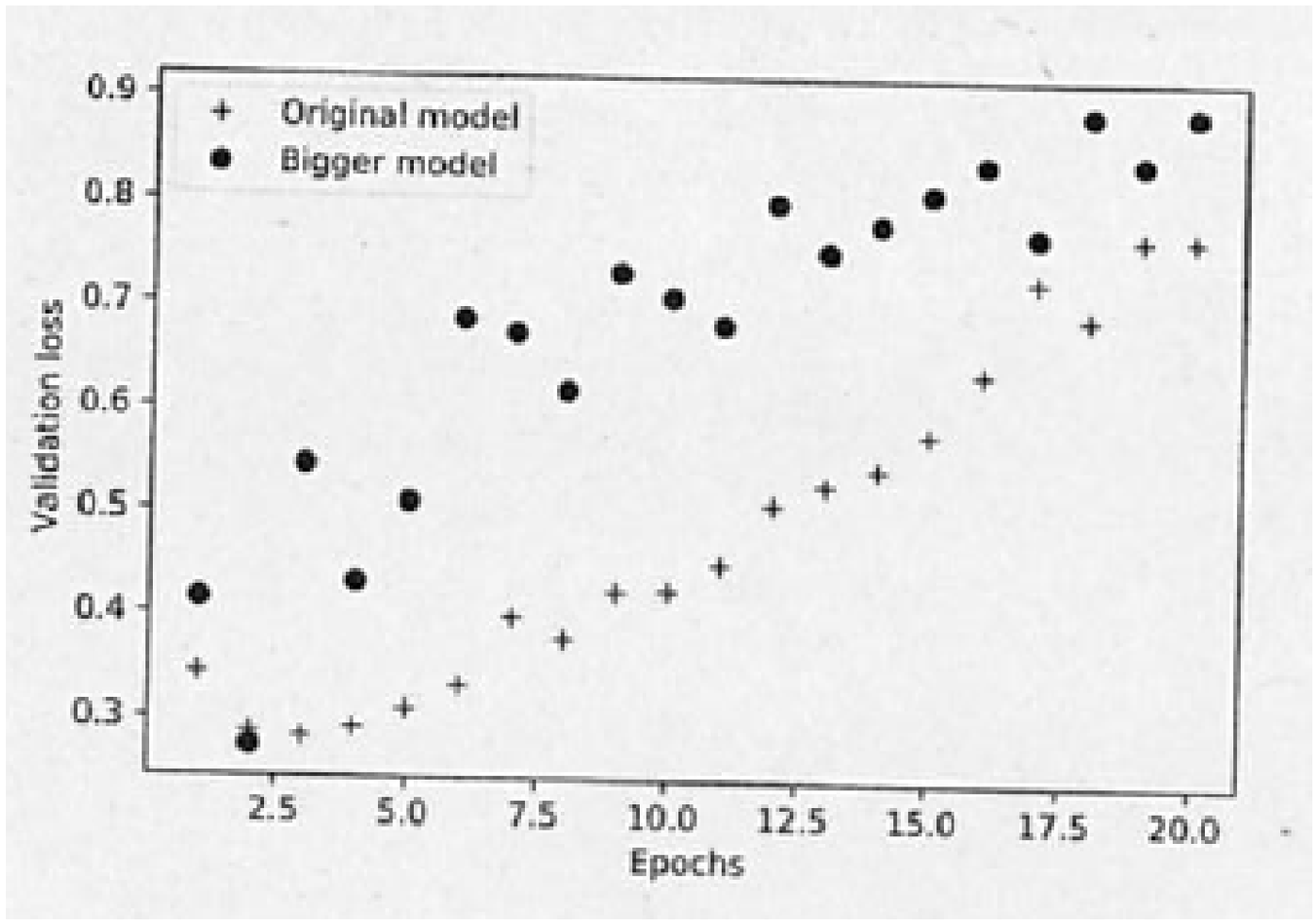
- モデルのキャパシティが検証データセットでの損失値に与える影響



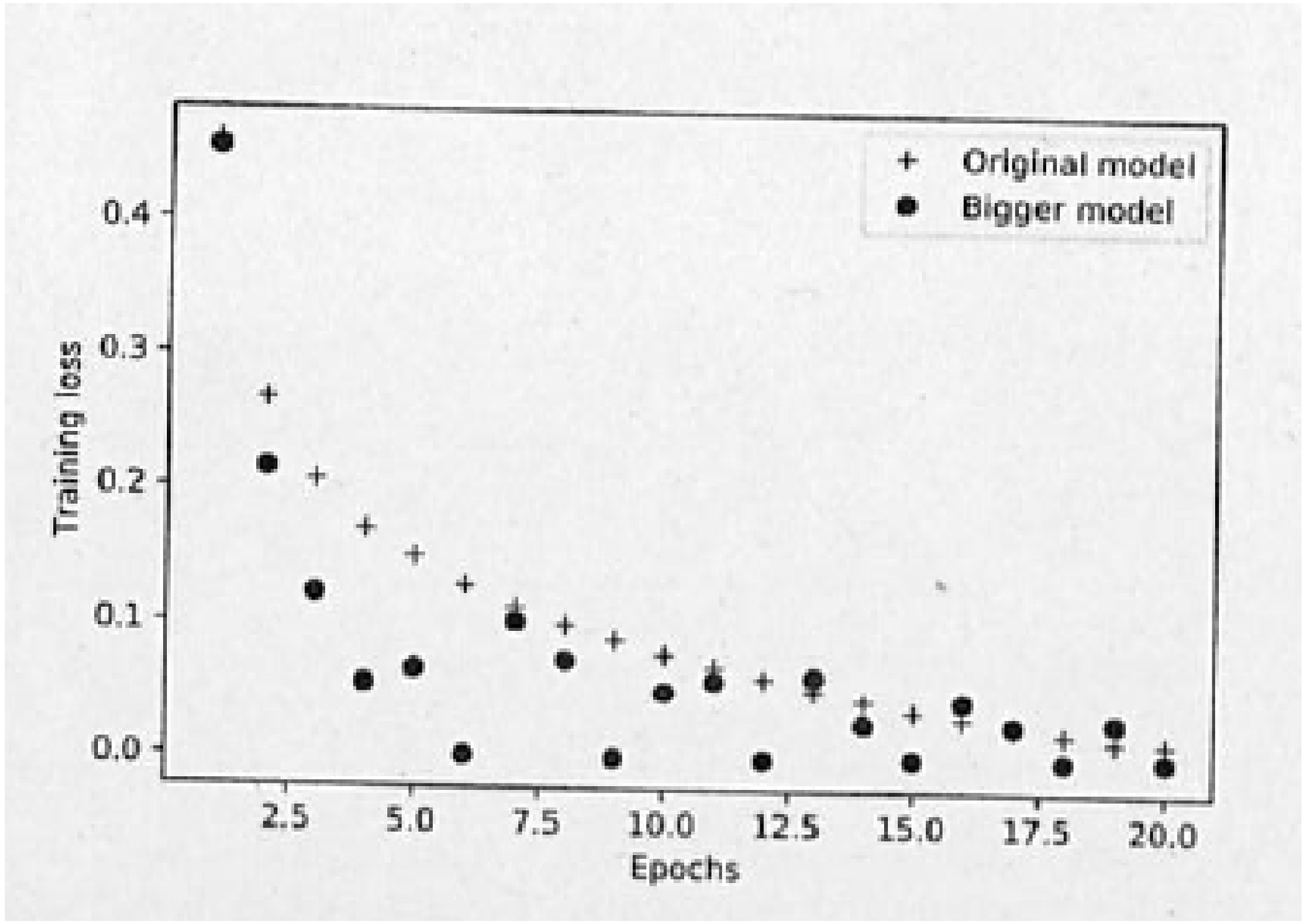
- キャパシティが大きいモデル

```
model=models.Sequential()
model.add(layers.Dense(512,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(512,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

- モデルのキャパシティが検証データセットでの損失値に与える影響



- モデルのキャパシティが訓練データセットでの損失値に与える影響



4.4.2　重みを正則化する

　オッカムの剃刀とは、「ある事柄を説明するためには、必要以上に多くを仮定するべきでない」とする指針。この考え方はNNに適用すると、より単純なモデルのほうが、より複雑なモデルよりも過学習に陥りにくいです。
　重みの正則化するには、大きな重みを使用する場合のコストをネットワークの損失関数に追加します。このコストには、次の2種類があります。

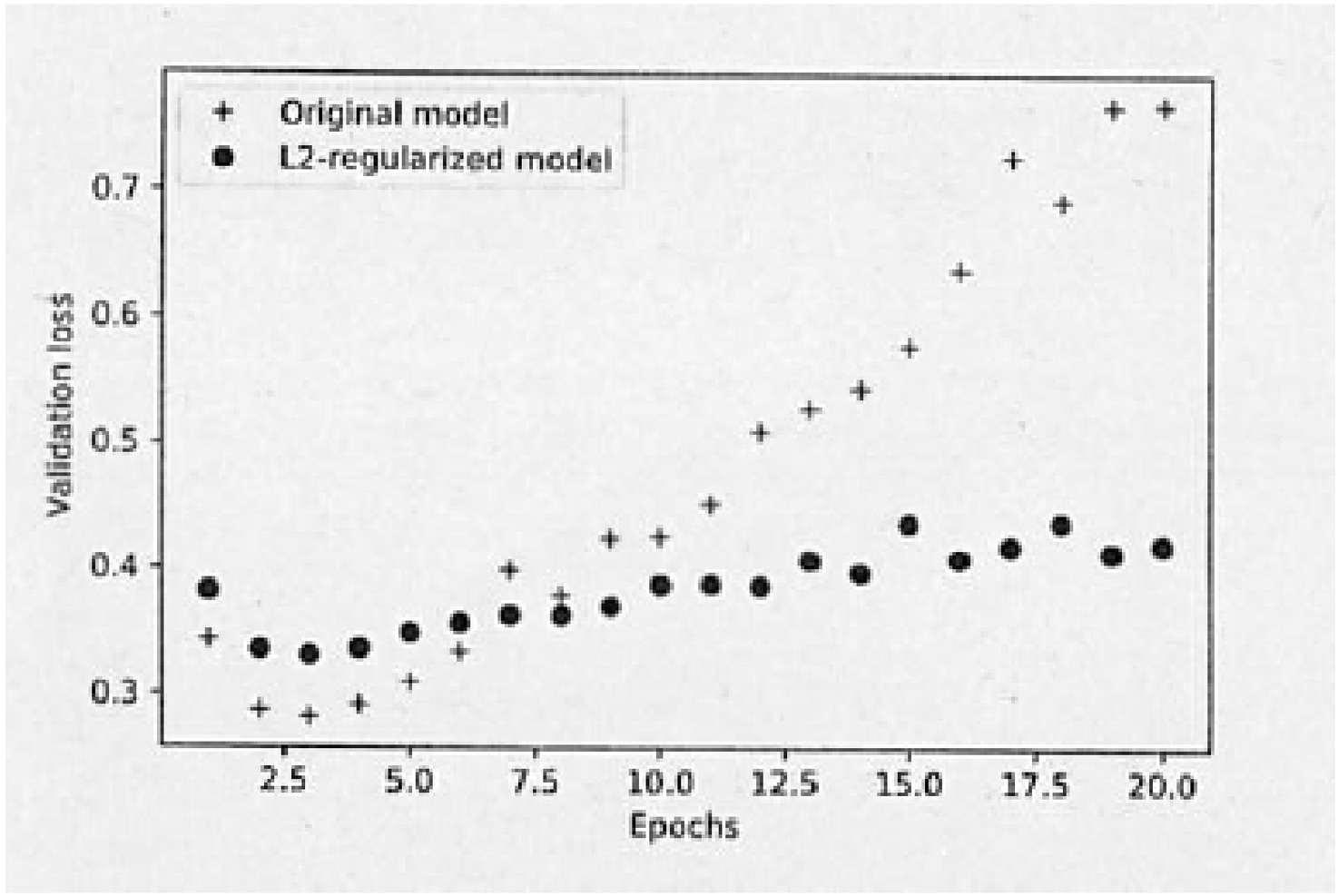
- **L1正則化**
  追加されるコストは重み係数の絶対値に比例します。
- **L2正則化**
  追加されるコストは重み係数の値の二乗に比例します。

- 映画レビュー分類ネットワークにL2正則化を追加

```
from keras import regularizers

model=models.Sequential()
model.add(layers.Dense(16,kernel_regularizer=regularizers.l2(0.001),
                       activation='relu',input_shape=(10000,)))
model.add(layers.Dense(16,kernel_regularizer=regularizers.l2(0.001),
                       activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

- L2正則化が検証データセットでの損失値に与える影響



- 重みを正則化するためのKerasの様々な正則化項

```
from keras import regularizers

regularizers.l1(0.01)
regularizers.l1_l2(l1=0.001,l2=0.001)
```

```
<keras.regularizers.L1L2 at 0x1812a1ca630>
```

4.4.3ドロップアウトを追加する

　ドロップアウトは、ニューラルネットワークにおいて最も効果的で最もよく使用されている正則化手法の1つです。これを層に適用すると、訓練中にその層の出力特徴量の一部がランダムに取り除かれ、０に設定されます。

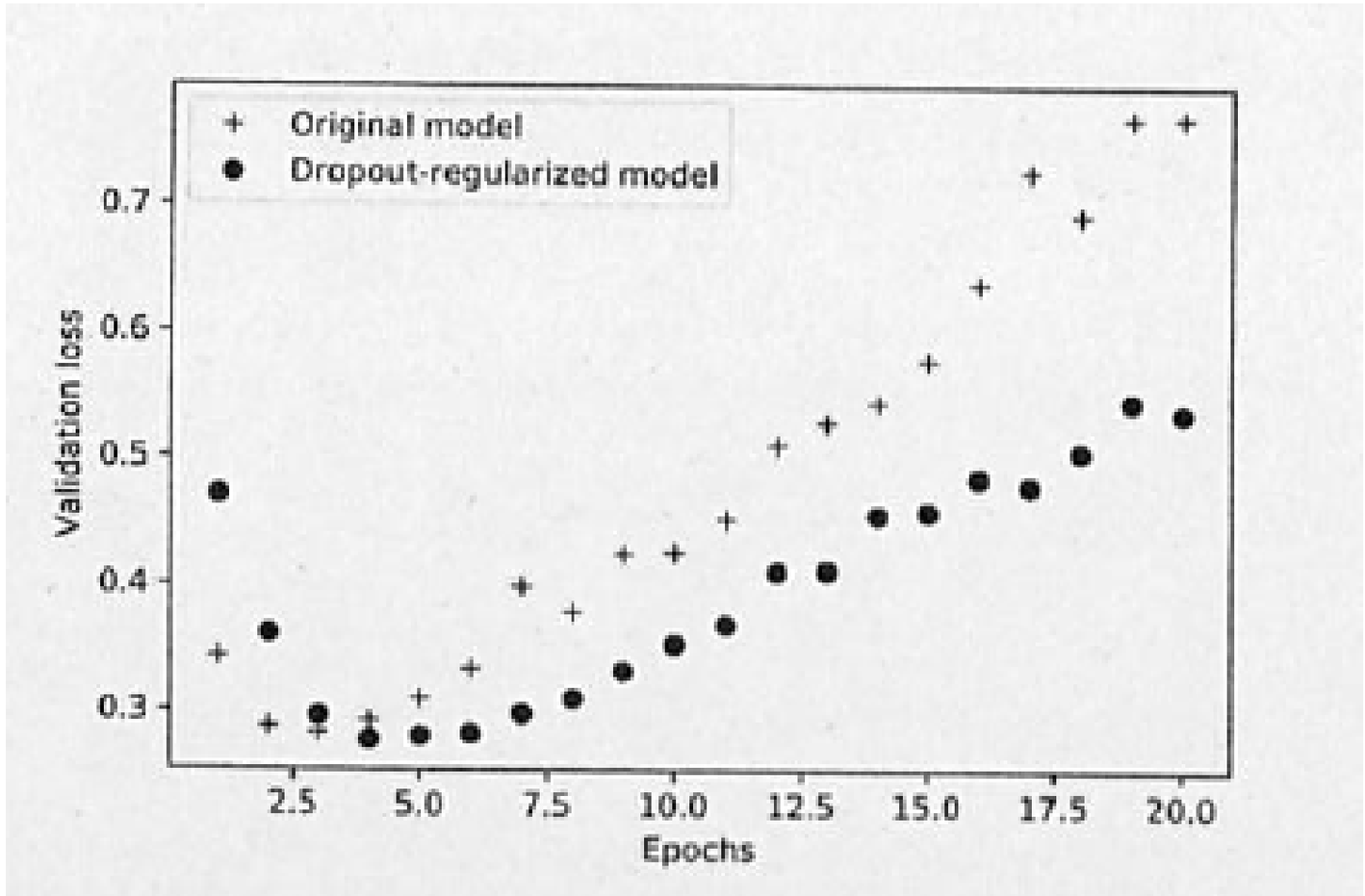
　ドロップアウト率は、ドロップアウトすると特徴量の割合であり、通常は0.2から0.5に設定されます。

　サンプルごとにニューロンの異なるサブセットをランダムに削除すれば、コンスピラシーが阻止され、過学習が抑制されることができます。

- 映画レビュー分類ネットワークにドロップアウトを追加

```
model=models.Sequential()
model.add(layers.Dense(16,activation='relu',input_shape=(10000,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1,activation='sigmoid'))
```

- ドロップアウトが検証データセットでの損失値に与える影響



ニューラルネットワークで過学習を防ぐための最も一般的な方法をまとめます。

1. 訓練データを増やす。
2. ネットワークのキャパシティを減らす。
3. 重みを正則化する。
4. ドロップアウトを追加する。