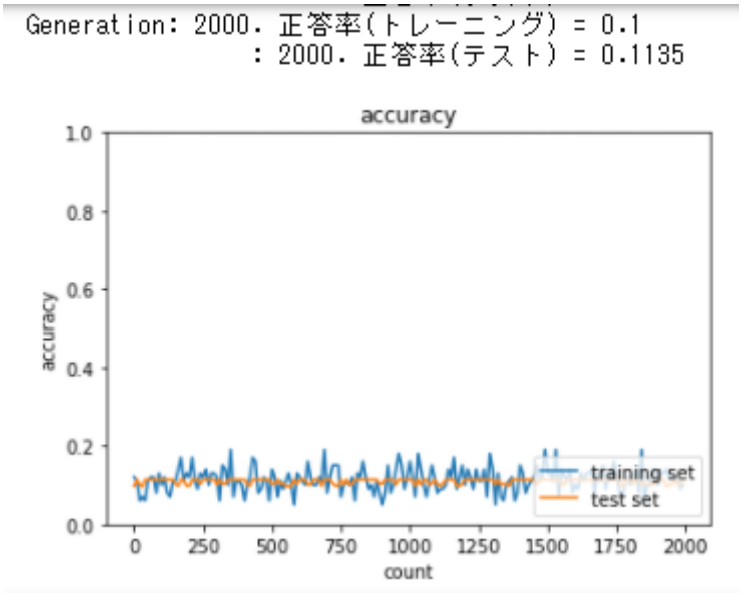
- 勾配消失問題
  - ✓ 要点のまとめ

    勾配消失問題とは、誤差逆伝播法において、下位の層に逆伝播が進むにつれ、重みの更新量が少なくなること。主な原因に活性化関数の導関数の最大値が 1 未満であることがあげられる。

    勾配消失問題の解決方法として、活性化関数に ReLU など、導関数の最大値が 1 未満でないものを採用することがあげられる。また、重みの初期値に He や Xavier を用いることでも勾配消失問題が改善される。活性化関数に値を渡す前にバッチ正規化の処理を施す、バッチ正規化という手法もある。
  - ✓ 実装演習結果キャプチャと考察

    活性化関数：シグモイド

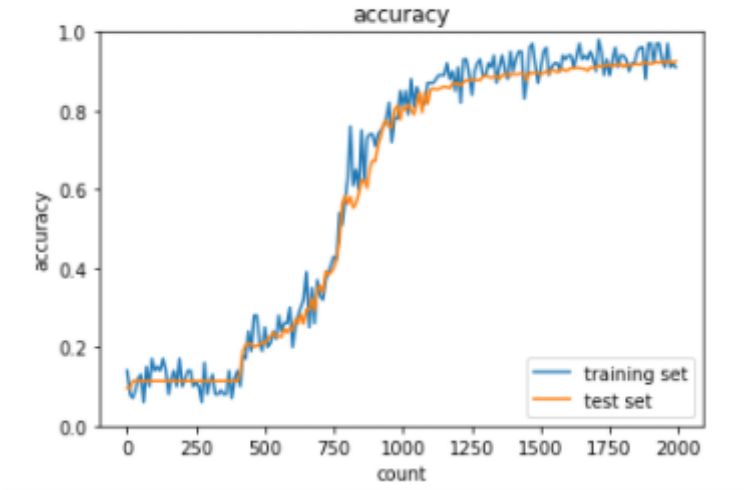
    初期値：ガウス分布



    活性化関数：ReLU

    初期値：ガウス分布

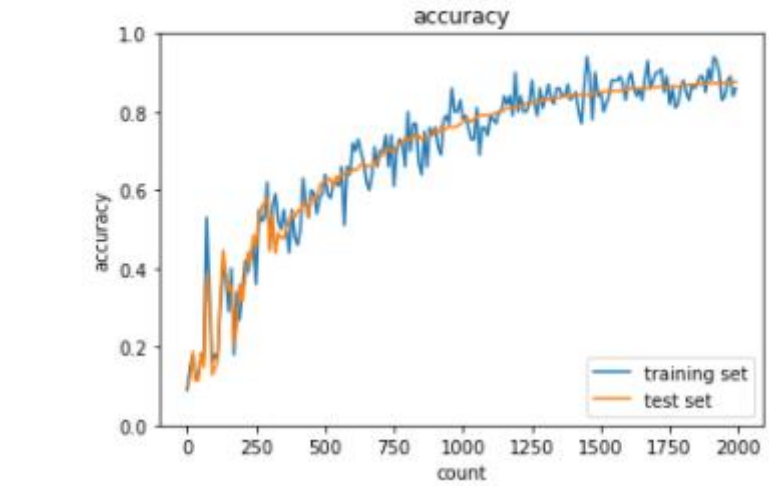
```
Generation: 2000. 正答率(トレーニング) = 0.91
                : 2000. 正答率(テスト) = 0.9253
```



活性化関数：シグモイド

初期値：Xavier

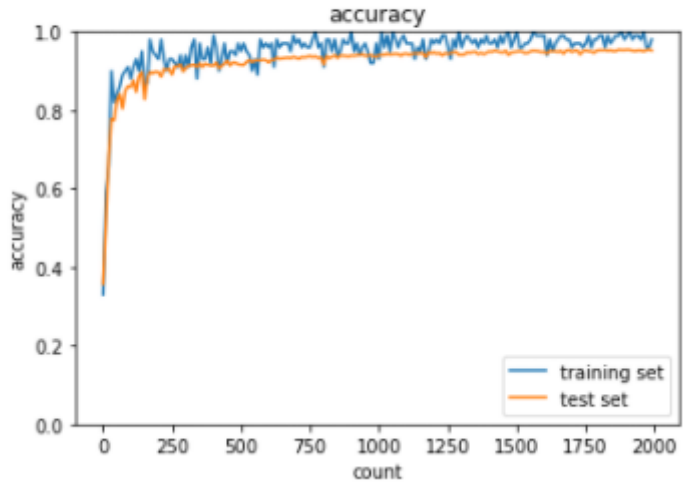
```
Generation: 2000. 正答率(トレーニング) = 0.86
                : 2000. 正答率(テスト) = 0.8762
```



活性化関数：ReLU

初期値：He

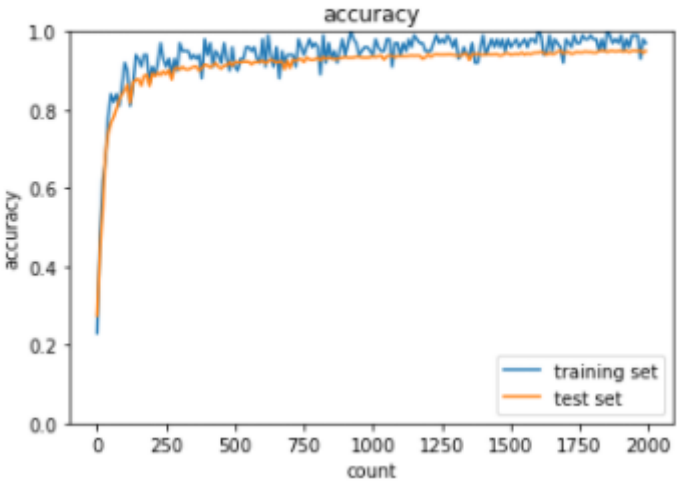
```
Generation: 2000. 正答率(トレーニング) = 0.98
                : 2000. 正答率(テスト) = 0.9523
```



活性化関数：ReLU

初期値：Xavier

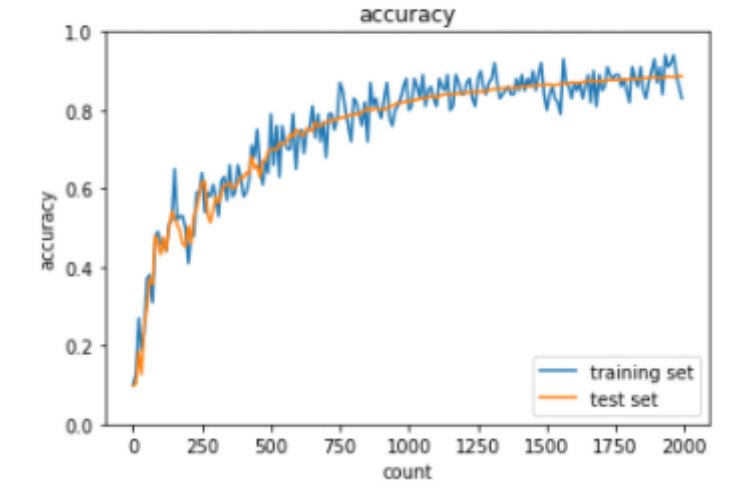
```
Generation: 2000. 正答率(トレーニング) = 0.97
                : 2000. 正答率(テスト) = 0.9502
```



活性化関数：シグモイド

初期値：He

```
Generation: 2000. 正答率(トレーニング) = 0.83
                : 2000. 正答率(テスト) = 0.8855
```



- ✓ 確認テストなど、自身の考察結果

  バッチ正規化の欠点は、バッチサイズをある程度（一般的には 16 以上）確保しないと使えないという点である。これは、エッジコンピューティングを行う際などに問題となる。この欠点を補うための手法として、レイヤー正規化、インスタンス正規化、グループ正規化などの手法がある。時系列データに対してよく使われる RNN に対してはレイヤーノームが有効である。

  参考：https://qiita.com/omiita/items/01855ff13cc6d3720ea4

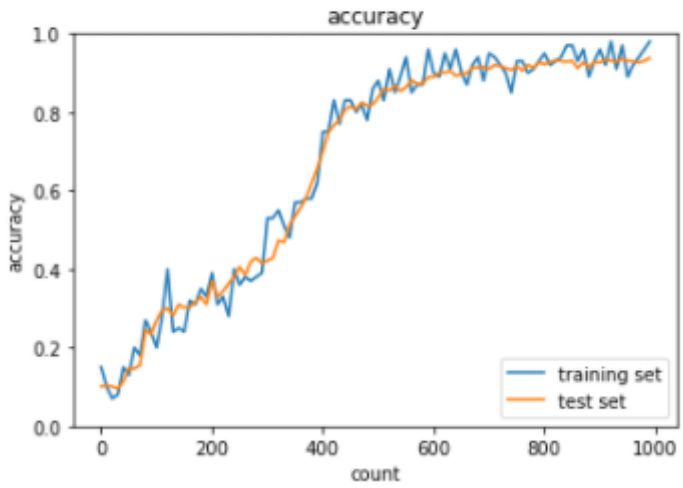
- ➢ 学習率最適化手法
  - ✓ 要点のまとめ

    学習率を適切に設定しない場合、誤差関数の値が収束しない（発散）ことや、収束までのエポック数が大きくなること、局所解に収束してしまうなどの問題が起きることがある。学習率を最適に決定するアルゴリズムとして「モメンタム」「AdaGrad」「RMSProp」「Adam」などがある。
  - ✓ 実装演習結果キャプチャと考察

    MNIST を各種学習率最適化手法で解く演習

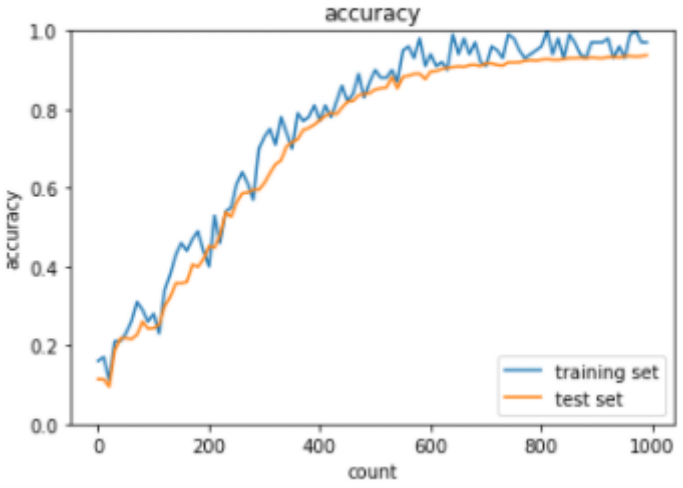
    ・モメンタム

```
Generation: 1000. 正答率(トレーニング) = 0.98
                : 1000. 正答率(テスト) = 0.938
```



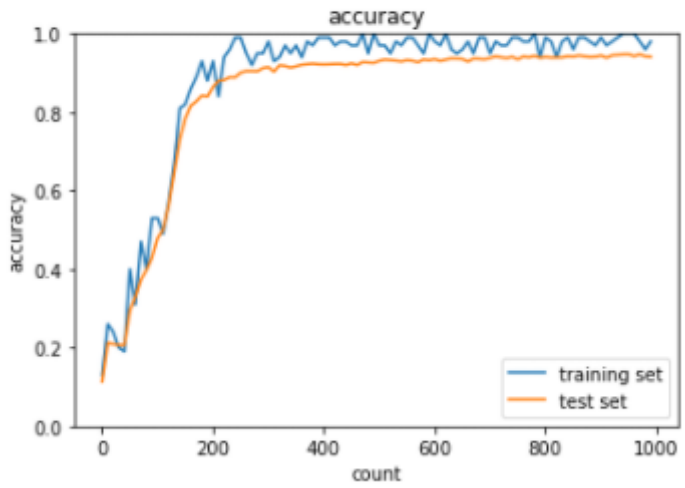
・AdaGrad

```
Generation: 1000. 正答率(トレーニング) = 0.97
                : 1000. 正答率(テスト) = 0.9375
```



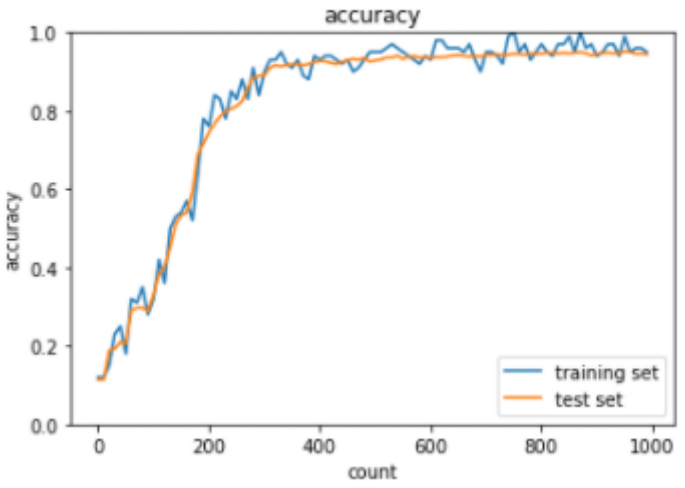
・RMSProp

Generation: 1000. 正答率(トレーニング) = 0.98
: 1000. 正答率(テスト) = 0.9408

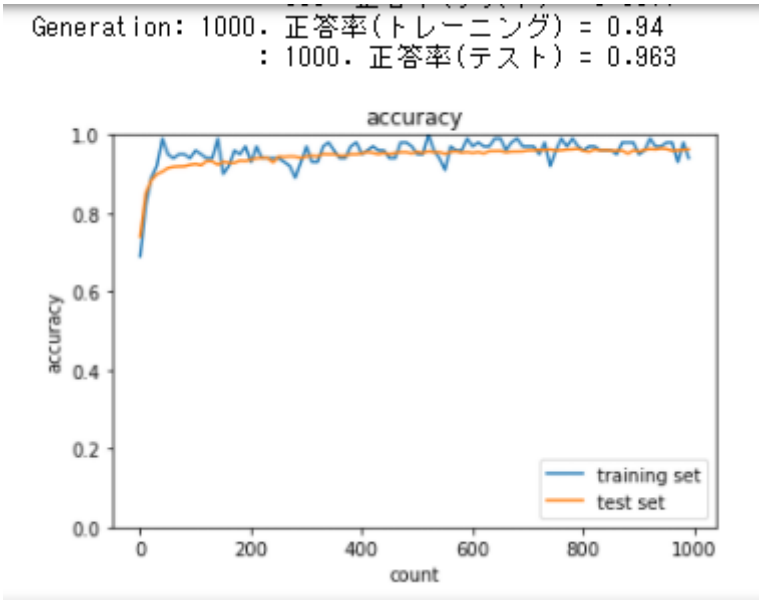


・Adam

Generation: 1000. 正答率(トレーニング) = 0.95
: 1000. 正答率(テスト) = 0.9437



・Adam(バッチ正規化あり,活性化関数：ReLU, 重みの初期値：He)
収束が早まり、精度も向上した。



✓ 確認テストなど、自身の考察結果

・モメンタム

局所最適解になりにくい。更新量の算出に慣性項を導入することで、収束が早まっている。

・AdaGrad

誤差が凹みの極値に近づくほど、重みの更新量を減らすように、誤差関数の偏微分を 2 乗した項を使っている。誤差関数の導関数が 0 となる鞍点で学習が進まなくなる、鞍点問題を起こすことがある。

・RMSProp

勾配の大きなところで学習率を低くするような項が採用されており、学習の際に振動を起こしにくくなっている。

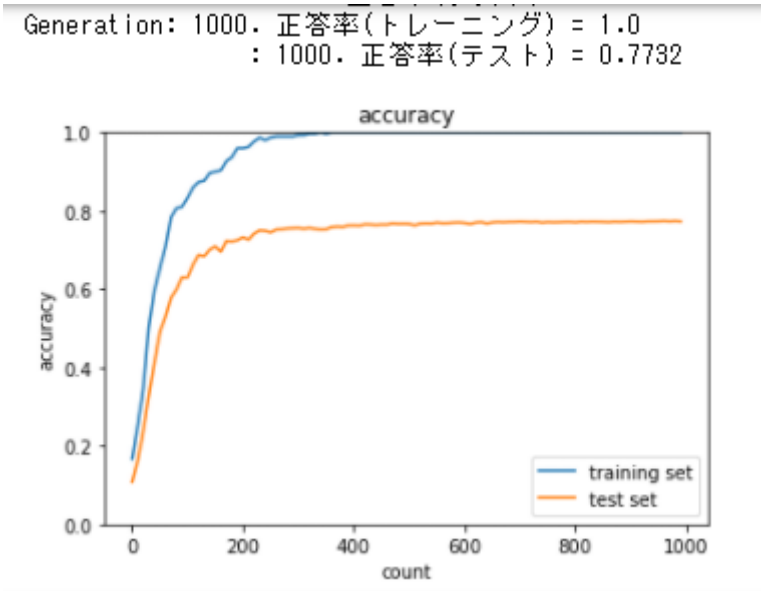
・Adam

モメンタムのメリットと、RMSProp のメリットを併せ持ったアルゴリズム

- 過学習
  - ✓ 要点のまとめ

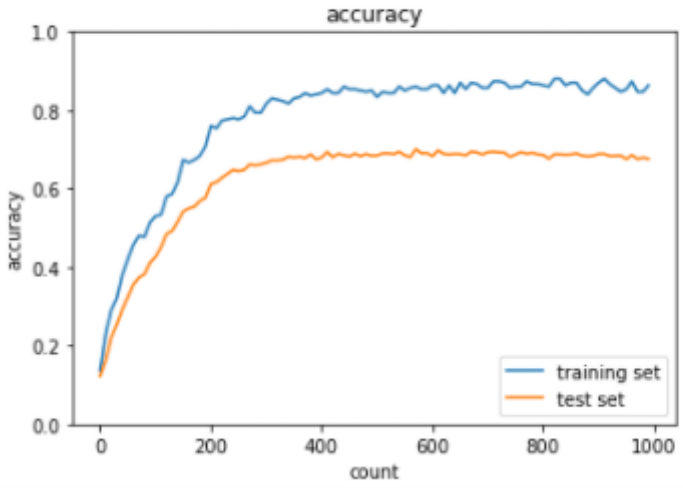
    過学習とは、訓練データに過剰に適合し、検証データやテストデータに適合しない状態のこと。汎化性能が低いとも言い換えることができる。過学習を防ぐ方法として、正則化項の導入がある。重みが大きくなることにペナルティを与えることで、汎化性能を失わない範囲で学習を進めることができる。Lasso 正則化と Ridge 正則化の 2 手法がある。
  - ✓ 実装演習結果キャプチャと考察

    正則化も dropout も設定しない場合、訓練データ正答率が 100%になってしまい、検証データの正答率と大きく乖離している。



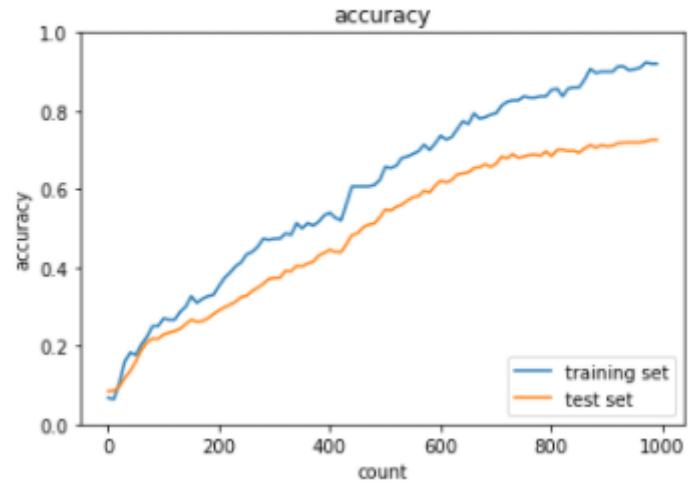
L2 正則化を導入すると、訓練と検証の正答率の乖離はやや小さくなる。600 回目の試行あたりから検証の正答率が下がり始めており、1000 回では試行回数が多すぎるように見受けられる。

```
Generation: 1000. 正答率(トレーニング) = 0.8633333333333333
                : 1000. 正答率(テスト) = 0.6759
```

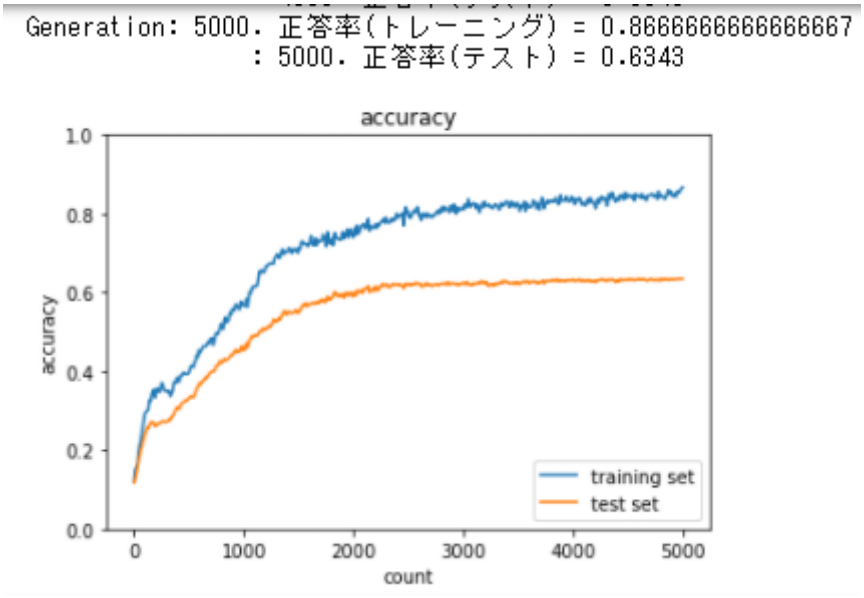


dropout を導入したモデル。訓練と検証の正答率の乖離が小さく、テストの正答率も上記のものと比べて高い。dropout には過学習を抑える効果がある一方、学習に時間がかかるというデメリットがある。1000 回の試行の段階では検証の正答率が上昇している途中であり、学習が完了しているとは言えない。

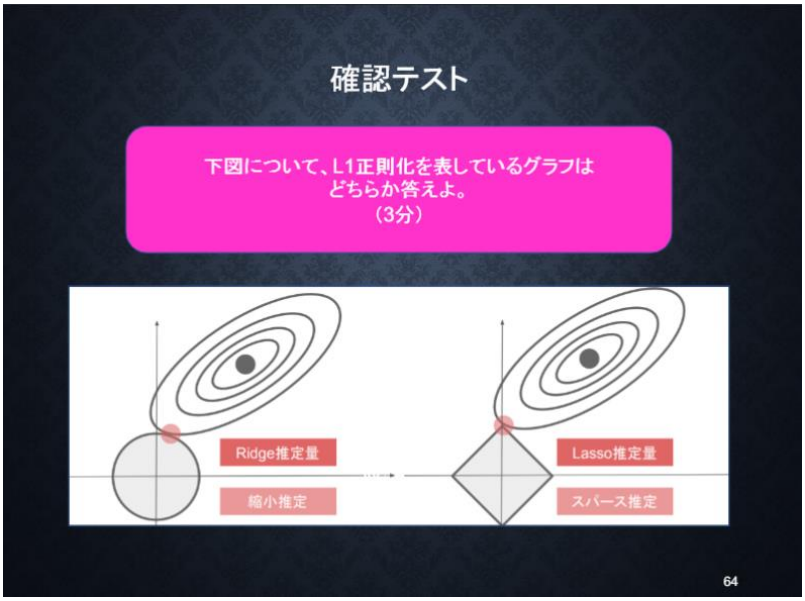
```
Generation: 1000. 正答率(トレーニング) = 0.92
                : 1000. 正答率(テスト) = 0.7261
```



L2 正則化＋dropout(0.15)+イテレーション(5000)のモデル



✓ 確認テストなど、自身の考察結果



L1 正則化を表すのは右の図。L1 正則化はパラメータの大きさをマンハッタン距離で評価する。そのため、パラメータを 2 次元と見立てた場合、等高線はひし形を描く。一方で L2 正則化はパラメータの大きさをユークリッド距離で評価する。そのため、パラメータを 2 次元と見立てた場合、等高線は円を描く。

➢ 畳み込みニューラルネットワークの概念

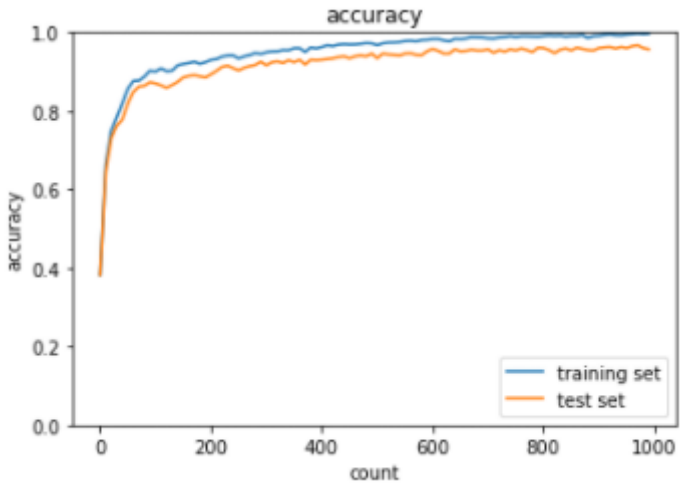
✓ 要点のまとめ

畳み込みニューラルネットワークとは、畳み込み層を含んだニューラルネットワークのことである。畳み込み層は、次元間に関係性のあるようなデータ（Ex.画像データ、時系列データ）に対して有効な手法である。入力データに対してフィルタ

ーをかける。学習の対象となるのはフィルターの重みである。フィルターをかけることで、データのサイズが小さくなってしまう問題は、パディングの手法を用いて解決することができる。代表的なのはデータの周囲を 0 で埋める zero-padding である。過学習を防ぐため、最初の 2 つの全結合層には dropout が設定されている。

✓ 実装演習結果キャプチャと考察

2_6_simple_convolution_network_after の結果



im2col の結果

```
========== input_data ===========
 [[[[48.  2. 74. 50.]
   [60.  0. 56. 22.]
   [62. 44.  0. 95.]
   [70. 60. 21. 76.]]]


 [[[56. 96. 73. 30.]
   [80. 67. 13.  1.]
   [62. 24. 85. 17.]
   [69. 32. 60. 93.]]]]
==============================
============= col ==============
 [[48.  2. 74. 60.  0. 56. 62. 44.  0.]
 [ 2. 74. 50.  0. 56. 22. 44.  0. 95.]
 [60.  0. 56. 62. 44.  0. 70. 60. 21.]
 [ 0. 56. 22. 44.  0. 95. 60. 21. 76.]
 [56. 96. 73. 80. 67. 13. 62. 24. 85.]
 [96. 73. 30. 67. 13.  1. 24. 85. 17.]
 [80. 67. 13. 62. 24. 85. 69. 32. 60.]
 [67. 13.  1. 24. 85. 17. 32. 60. 93.]]
==============================
```

im2col の結果（transpose なし）

```
========== input_data ===========
 [[[[14. 76. 97. 14.]
   [44. 48. 34. 61.]
   [ 2. 88.  9. 53.]
   [35. 91.  1. 31.]]]


 [[[96. 41. 66. 64.]
   [56. 57. 67. 31.]
   [22. 75. 63. 49.]
   [ 9. 30. 23. 17.]]]]
=============================
============= col ==============
 [[14. 76. 44. 48. 76. 97. 48. 34. 97.]
 [14. 34. 61. 44. 48.  2. 88. 48. 34.]
 [88.  9. 34. 61.  9. 53.  2. 88. 35.]
 [91. 88.  9. 91.  1.  9. 53.  1. 31.]
 [96. 41. 56. 57. 41. 66. 57. 67. 66.]
 [64. 67. 31. 56. 57. 22. 75. 57. 67.]
 [75. 63. 67. 31. 63. 49. 22. 75.  9.]
 [30. 75. 63. 30. 23. 63. 49. 23. 17.]]
=============================
```

- ✓ 確認テストなど、自身の考察結果

  講義では 3 次元データ(画像)に対する 2 次元フィルタの例が扱われたが、4 次元データに対して 3 次元フィルタを適用することも可能なのではないかと思い、調べてみた。CT 画像に対して 3 次元フィルタを用い、肺野結節影の検出に応用した新潟大学大学院の事例があった。この例では ResNet を三次元化した 3D-ResNet をを用いて学習を行っている。

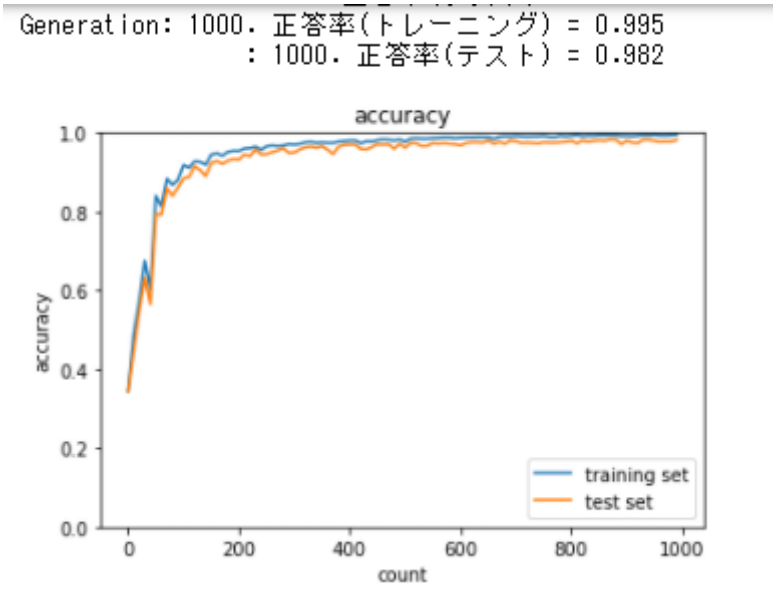
  参考：https://www.jstage.jst.go.jp/article/pjsai/JSAI2020/0/JSAI2020_1C5GS1305/_article/-char/ja/

- 最新の CNN
  - ✓ 要点のまとめ

    AlexNet は、2012 年の ILSVRC において、従来の SVM などを用いた画像認識に大差をつけて優勝し、深層学習の有効性を知らしめたモデル。5 層の畳み込み層と、3 層のプーリング層、3 層の全結合層からなる。活性化関数は ReLU。プーリングの際にフィルタサイズよりも小さなストライドを用いる Overlapping Pooling が用いられている。
  - ✓ 実装演習結果キャプチャと考察

    2_8_deep_convolution_net の実行結果。

    colab ではなく、CPU のみのローカル環境で実行したところ、学習に小一時間を要した。これまでの演習で扱ってきたモデルの中で、圧倒的に高い精度 0.982 を記録した。



✓ 確認テストなど、自身の考察結果

自身の調べものとして 2019 年に発表された EfficientNet について調べた。

EfficientNet は 2019 年に GoogleBrain から発表されたモデル。

この論文の主題は畳み込みニューラルネットワークの深さ・広さ・解像度が精度にどう影響するか調べ、Compound Coefficient という指標を導入することで性能を上げた、というもの。これによって作られた EffientNet-B7 は今間までの SoTA モデルよりも少ないパラメータと計算量で ImageNet における SoTA を達成した。

参考：https://qiita.com/omiita/items/83643f78baabfa210ab1#