

SVMの説明書

～実践編～

@salinger01101

BETTERな手順(前回のおさらい)

1. データをSVMで使えるように整形。
2. 素性の選択
3. データのスケーリング。
4. RBFカーネルを利用。
5. 交差検定により、最適なコストパラメータ: C とRBFカーネルの γ パラメータを調べる。
6. 最適なパラメータを用いて、モデルの生成を行う。
7. テストデータで試行。

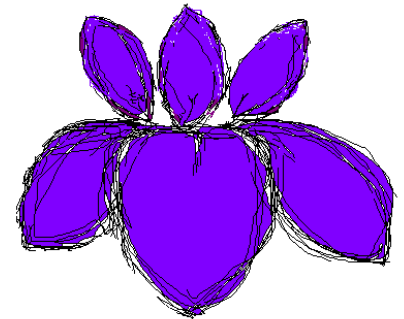
今回の目的

**実際にSVMのパラメータ
を決定してみよう**

テスト用のデータを準備する

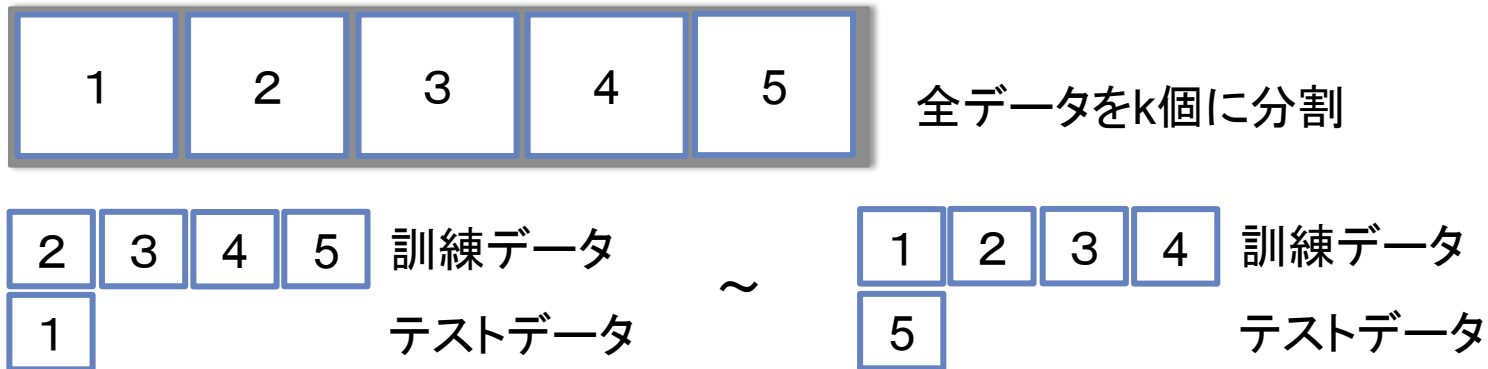
• Iris (アイリス)

- Rに標準で入っているデータセット。
- 3品種(setosa, versicolor, virginica)のあやめの花50本ずつ。
- 花の萼(がく)の長さ、花弁の長さ、幅を測ったもの。
 - あやめの大きな3枚の花びらが、「Sepal (がく片)」で、小さな3枚の花びらが、「Petal (花びら)」である。
- デフォルトでSVMに使いやすい形に整形済み。
- データのオーダーも揃ってるので、スケーリングも必要無し。



交差検定(再掲)

- 訓練データとテストデータの分割方法



このようにk回試行し、その平均を利用する。

こうすることで、テストデータは常に未知のデータになる。
⇒過剰適応(Over fitting)を防げる

BETTERな手順


1. データをSVMで使えるように整形。

2. 素性の選択

3. データのスケーリング。

4. RBFカーネルを利用。

とりあえず、ここまでやってみる。



5. 交差検定により、最適なコストパラメータ: C とRBFカーネルの γ パラメータを調べる。

6. 最適なパラメータを用いて、モデルの生成を行う。

7. テストデータで試行。

サンプルコード(1)

```
library(kernlab)
library(ggplot2)
```

kernlab: SVMのライブラリ (LIBSVMのWrapper)
ggplot2: 2Dグラフ用のライブラリ 事前にインストールしておく

```
# Check iris dataset
head(iris)
nrow(iris)
summary(iris)
```

head(): 先頭から6要素分のデータを確認
nrow(): 列数の確認 (今回はデータ件数の確認)
summary(): データの概要を確認

```
# default parameter
classifier.default <- ksvm(
  Species ~., # Define Label
  data=iris,
  type="C-svc",
  kernel="rbfdot",
  cross=nrow(iris) # LOOCV
)
```

ksvm(): kernlab中のSVM Wrapper 関数
Species ~., : ある行をラベルに指定する。残りの行は素性となる。
data: データフレーム(Rのデータ形式の一つ)を指定
type: SVMのタイプを指定。通常は“C-svc”で問題ない。
kernel: RBFカーネルを指定。
cross: 交差検定を行う場合、分割数を指定する。
LOOCV: Leave-one-out 交差検定法のこと。
テストに1件のみを使い、残りをすべて学習用にする。
今回は150施行した平均のAccuracyとなる。
データ件数が少ない場合にオススメな交差検定法。

```
acc.default <- 1 - cross(classifier.default)
print(acc.default)
```

cross(): 作成した分類器の交差検定結果を取得する、
ただし、誤分類率なので、Accuracyは 1 - cross(classifier) で求める。

デフォルトパラメータでの結果

Accuracy(平均値) = 0.9466667

100個中94個はきちっと分類してくれるので、なかなか良い結果。

次は限界への挑戦。

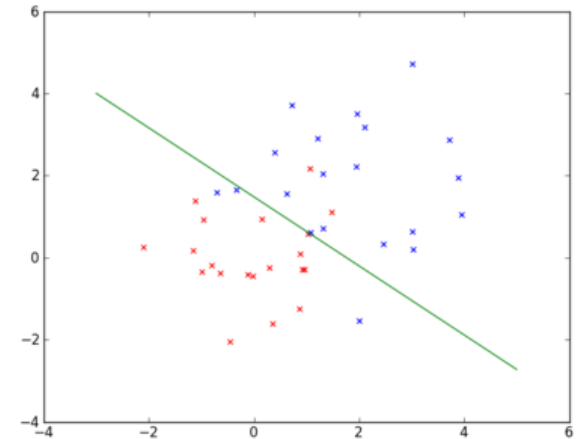
BETTERな手順

1. データをSVMで使えるように整形。
2. 素性の選択
3. データのスケーリング。
4. RBFカーネルを利用。
5. 交差検定により、最適なコストパラメータ: C とRBFカーネルの γ パラメータを調べる。
6. 最適なパラメータを用いて、モデルの生成を行う。
7. テストデータで試行。

調整するパラメータ

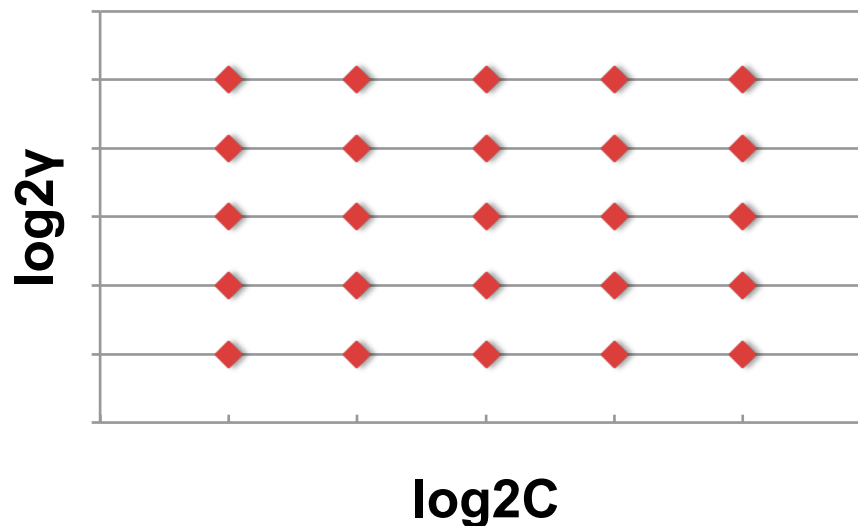
- コストパラメータ: C
 - 分類境界をどの程度まできっちりするか。
 - C が大きい \Rightarrow 誤りを許容しない
 - C が小さい \Rightarrow 誤りを許容する
- RBFカーネルのパラメータ
 - 式中の γ の値。

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i^T - \mathbf{x}_j\|^2) \quad (\gamma > 0)$$



グリッドサーチ(再掲)

- 2種類のパラメータを網羅的に探索
 - グラフの赤い点を網羅的に試す。
 - 荒い探索の後、細かい探索。
 - 指数増加列がよい。
 - Ex. $C = 2^n$ ($n = -5 \sim 15$), $\gamma = 2^m$ ($m = -15 \sim 3$)



サンプルコード(2-1)

Get SVM accuracy by args (Cost param, Sigma param)

```
svm.getacc <- function(c,sig){  
  classifier <- ksvm(  
    Species ~., # Define Label  
    data=iris,  
    type="C-svc",  
    kernel="rbfdot",  
    C = c,  
    kpar=list(sigma=sig),  
    cross=nrow(iris) # LOOCV  
  )  
  acc <- 1 - cross(classifier)  
  ret <- c(c,sig,acc)  
  cat(ret,"\n")  
  return(ret)  
}
```

この関数の引数は
コストパラメータとRBFカーネルのパラメータ

パラメータの指定部分

Accuracy, C, RBFカーネルのパラメータ
のベクトルを返す

サンプルコード(2-2)

Grid search

```
svm.gridsearch <- function(seq.c = -5:15, seq.sigma = -15:3){
```

```
  vec <- numeric(0)
```

```
  # Try All C and sigma combination
```

```
  for(c in 2^seq.c){
```

```
    for(sigma in 2^seq.sigma){
```

```
      vec <- c(vec, svm.getacc(c,sigma))
```

```
    }
```

```
  }
```

```
  m <- t(matrix(vec,nrow=3))
```

```
  colnames(m) <- c("c","sigma","acc")
```

```
  # Remove error in acc diff
```

```
  m[m[,3] < 10^-5,3] <- 0
```

```
  return(data.frame(m))
```

```
}
```

グリッドサーチのための関数

2種類のパラメータとその時のAccを返す
(データフレーム形式)。

引数で2種類のパラメータの範囲
 2^n の n を指定。

結果をベクトルに格納

ベクトルを行列に変換

元のベクトル:

[c1, sig1, acc1, c2, sig2, acc2, ...]

行列:

[[c1, sig1, acc1],

[c2, sig2, acc2],

...]

colnames(): 各要素の名前を指定

Accを計算する際に浮動小数点数引き算
時に生じた誤差を除去。

サンプルコード(2-3)

```
grid <- svm.gridsearch()
summary(grid)
write.table(grid,file="gridsearch.csv",row.names=F)
# acc = 0.5 ~ 1
```

グリッドサーチを実行
結果を変数 grid に代入し、
csv形式で出力

```
g = ggplot(grid,aes_string(x="c",y="sigma",z="acc")) + geom_tile(aes(fill=acc)) +
scale_x_continuous(trans="log2") + scale_y_continuous(trans="log2") +
scale_fill_continuous(limits=c(0.5, 1), breaks=seq(0,1,by=0.1))
```

```
print(g)
```

わかりやすく2Dグラフで視覚化。
browser(): ここで一時停止

```
browser()
```

```
# acc = 0.9 ~ 1
```

```
g = ggplot(grid,aes_string(x="c",y="sigma",z="acc")) + geom_tile(aes(fill=acc)) +
scale_x_continuous(trans="log2") + scale_y_continuous(trans="log2") +
scale_fill_continuous(limits=c(0.9, 1), breaks=seq(0,1,by=0.01))
```

```
print(g)
```

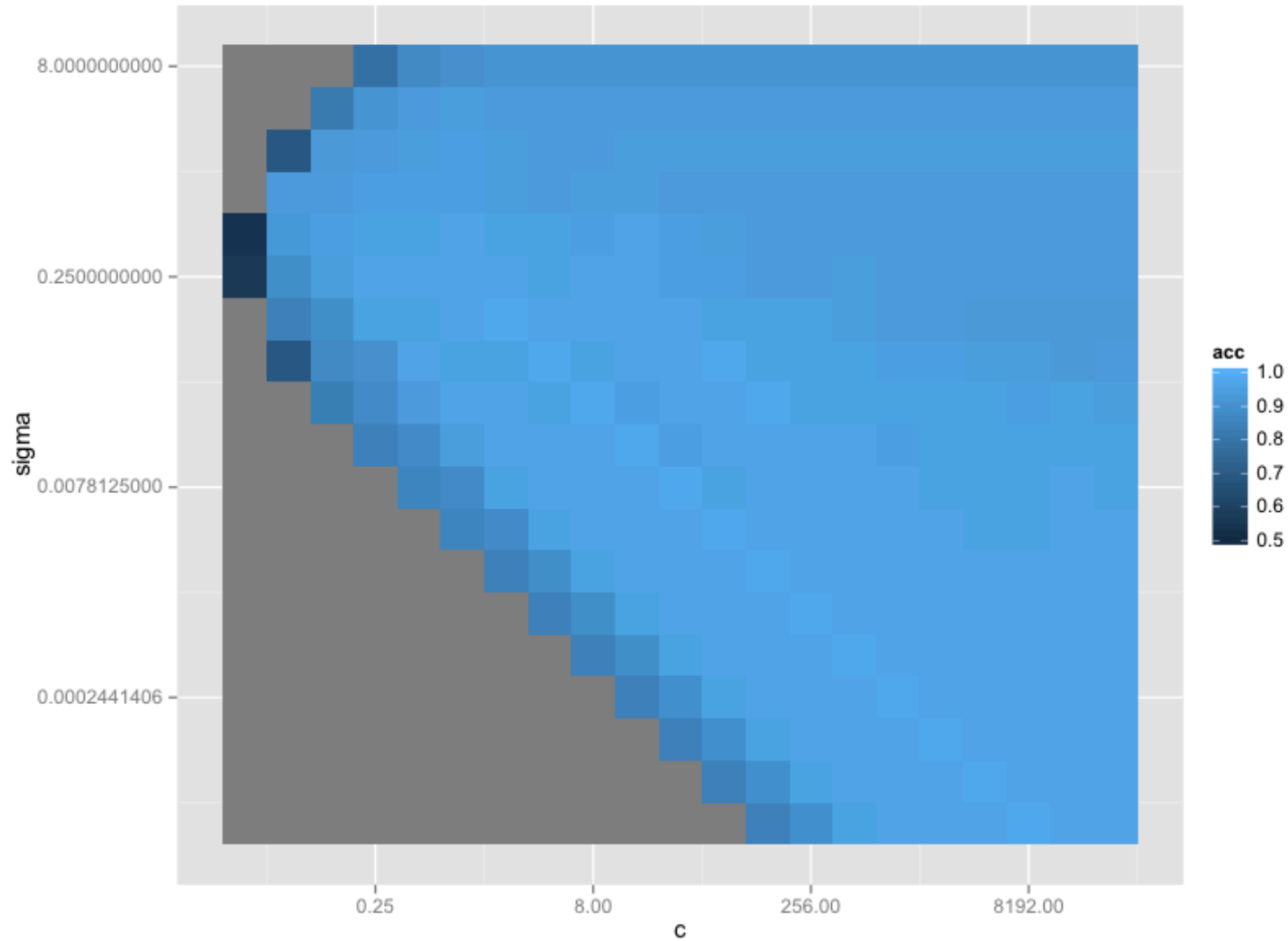
```
browser()
```

```
# acc = 0.95 ~ 1
```

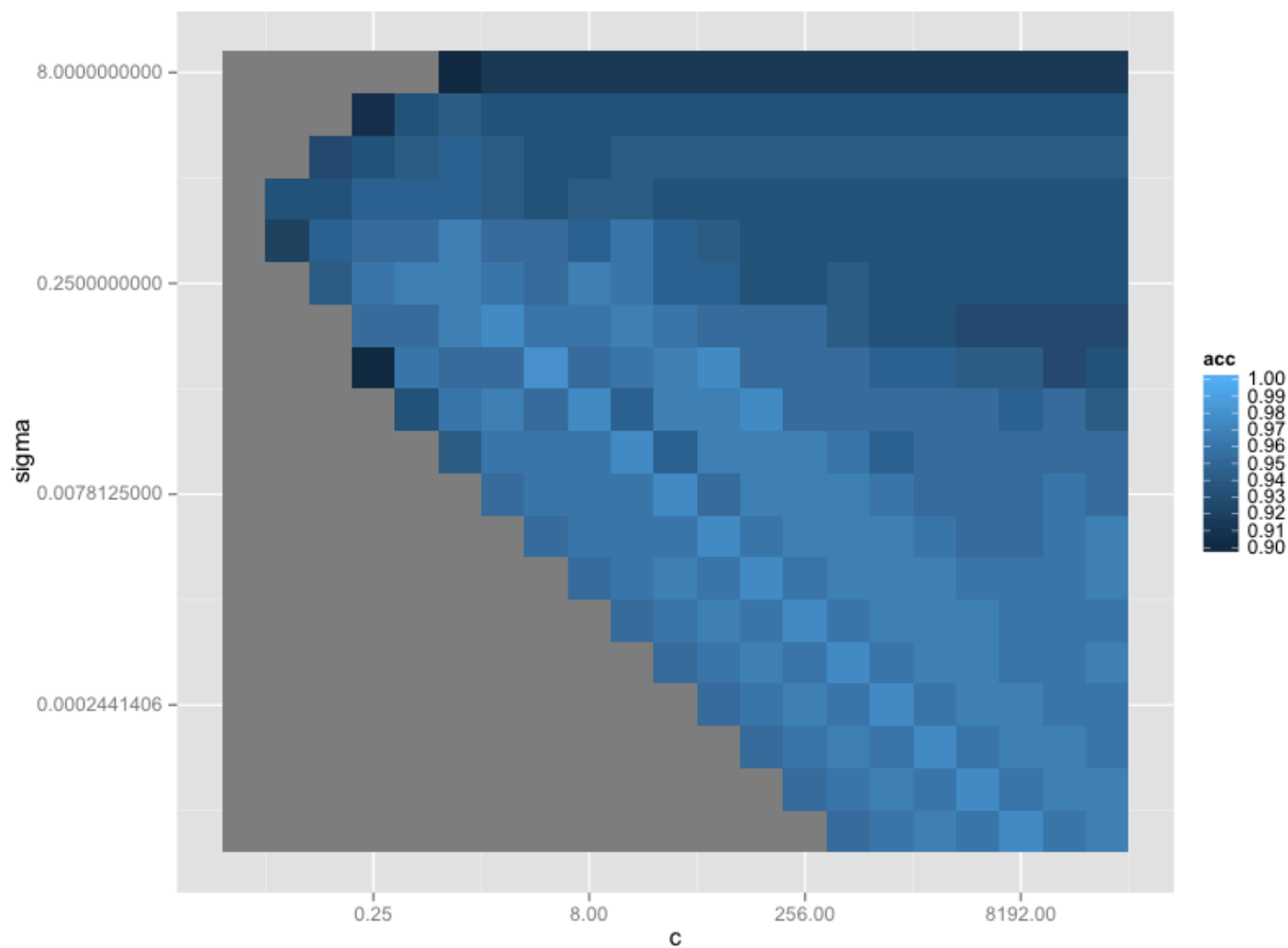
```
g = ggplot(grid,aes_string(x="c",y="sigma",z="acc")) + geom_tile(aes(fill=acc)) +
scale_x_continuous(trans="log2") + scale_y_continuous(trans="log2") +
scale_fill_continuous(limits=c(0.95, 1), breaks=seq(0,1,by=0.01))
```

```
print(g)
```

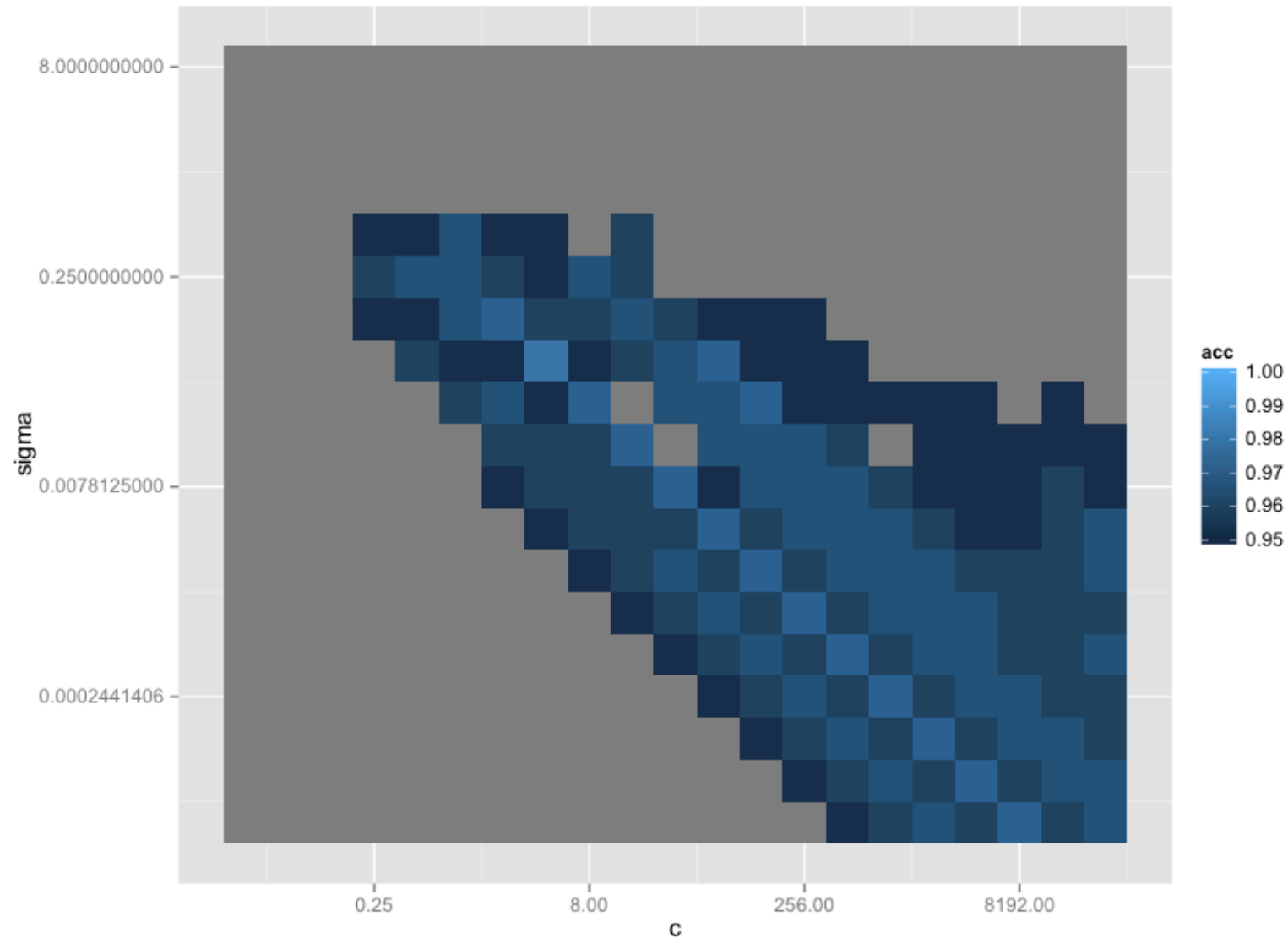
ACC = 0.5 ~ 1 のとき



ACC = 0.9 ~ 1 のとき



ACC = 0.95 ~ 1 のとき



サンプルコード(3)

```
grid.subset = subset(grid, grid$acc == max(grid$acc))  
print(grid.subset)
```

データフレームから、
条件に合う部分のみ切り出す

c	sigma	acc
4	0.0625	0.98

Accuracy(平均値) = 0.98

精度よくなった！完成！

BETTERな手順

1. データをSVMで使えるように整形。
2. 素性の選択
3. データのスケーリング。
4. RBFカーネルを利用。
5. 交差検定により、最適なコストパラメータ: C とRBFカーネルの γ パラメータを調べる。
6. 最適なパラメータを用いて、モデルの生成を行う。
7. テストデータで試行。

まとめ

- きちんと手順を守れば精度向上するよ！
- ただ、通常は素性の選択の方がパラメータチューニングよりもよく効くので、そちらの選定もがんばりましょう！

参考文献

・SVM実践ガイド (A Practical Guide to Support Vector Classification)

http://d.hatena.ne.jp/sleepy_yoshi/20120624/p1

・TAKASHI ISHIDA HomePage SVM

<http://www.bi.a.u-tokyo.ac.jp/~tak/svm.html>

・LIBSVM

<http://www.csie.ntu.edu.tw/%7Ecjlin/libsvm/>

・R統計解析入門: 「iris」の変数を抽出し、変数の順序を並べ替える。

http://monge.tec.fukuoka-u.ac.jp/r_analysis/basic_data_frame15.html

・使用したソースコード等

<https://github.com/Salinger/iris-svm>

SVMのモジュール

- **LIBSVM**

- SVMのモジュール。基本的にはこれで問題ない。
- 各言語用のバインディングあり。
- <http://www.csie.ntu.edu.tw/%7Ecjlin/libsvm/>

- **LIBLINEAR**

- 線形カーネルのみだが、高速。
- 各言語用のバインディングあり。
- <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

- **SVMLight**

- 大きなデータセットを高速に処理可能。
- <http://svmlight.joachims.org>