

サポートベクターマシン入門

～金貨の真贋を見分けよう～

@salinger001101

さりんじゃー

自己紹介

- 長岡技術科学大学 修士2年
 - 専門分野：機械学習・情報検索・自然言語処理
- 4月からは東京の某社でデータ分析のお仕事
- 主な使用言語
 - Python(メイン)、R(学習中)、Clojure(学習中)
- 趣味に色々と散財してます

注意

- 数式少なめ
 - 結構 about な説明になってる部分あり。
- わかんない！
 - 遠慮なく手を上げて、止めてください。

今回の内容

1. 機械学習とは
2. サポートベクターマシンの理論
3. 最適化のお話
4. 金貨の真贋を見分けよう

1. 機械学習とは

機械学習前夜

1970年 台

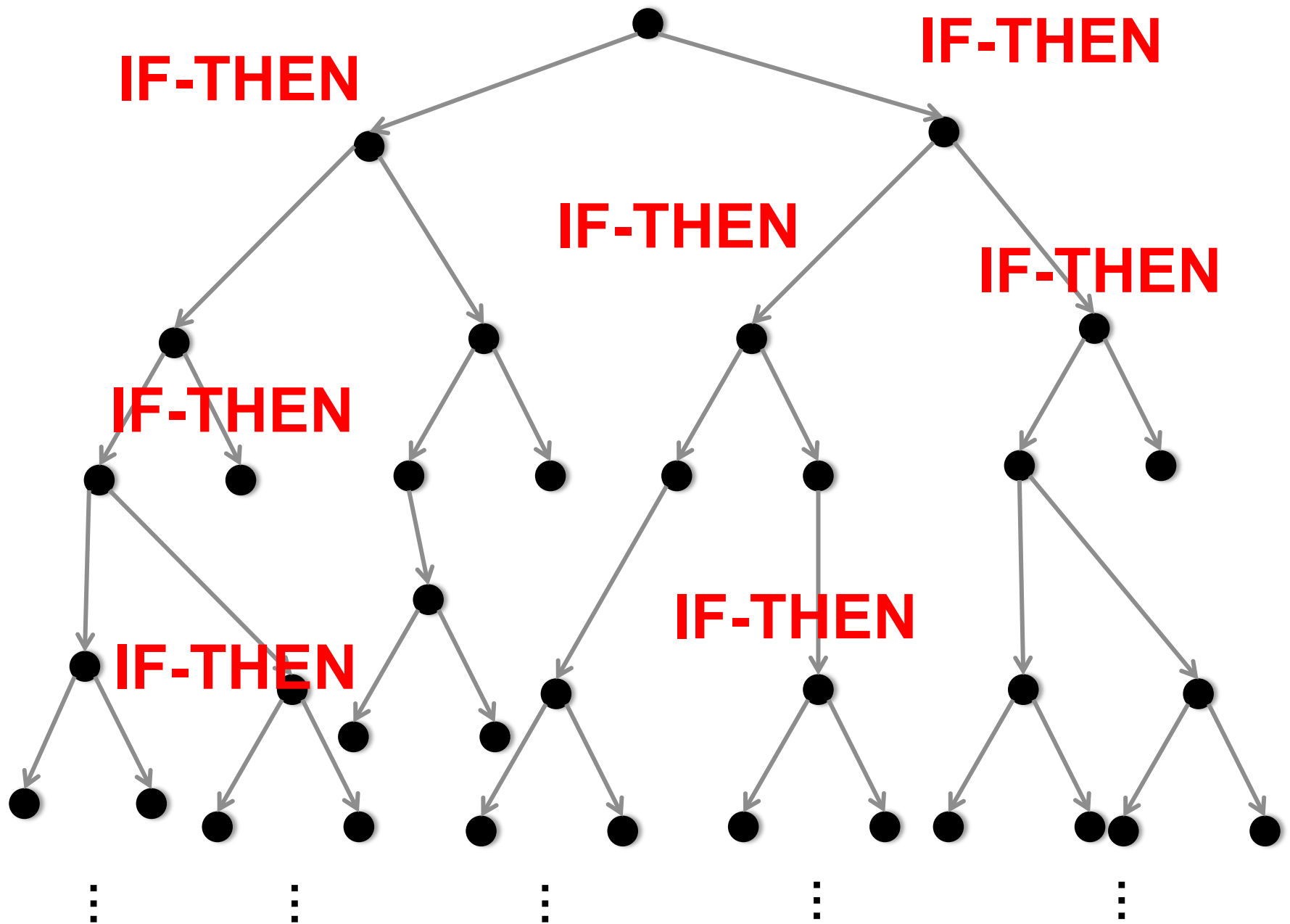
人間の判断を 自動化したい！

医療診断・対話型マニュアル・金融サービス・etc...

コンピュータに
ルールを覚えさせて
判断させよう！

エキスパートシステム 誕生！

しかし、
すぐに限界が！



「ルールを定式化できないよ！」

「ルール間に矛盾があるよ！」

「例外に対応できないよ！」

「ろくな結果が出てこねーよ！」

問題は...？

「人間には複雑すぎてルールを定式化できないよ！」

「人間が考えるルール間に矛盾があるよ！」

「人間の想定を超える例外に対応できないよ！」

...でも人間は普通に判断・行動できるよね？

人間は自然に
規則性・パターン・知識
などを抽出して
アルゴリズムを改良する

コンピュータが

規則性・パターン・知識

などを自動で抽出して

アルゴリズムを改良すれば...


機械学習とは 自己学習するシステム

- **教師あり学習**
(目標とする出力がある)
 - 入力から対応する出力モデルを学習する
- **教師なし学習**
(正解は事前にわからない)
 - 入力のみからモデルを構築する

さまざまな機械学習の手法

- Random Forest
- 階層型クラスタリング
- K平均法
- 単純ベイズ分類器
- ニューラルネットワーク
- サポートベクターマシン

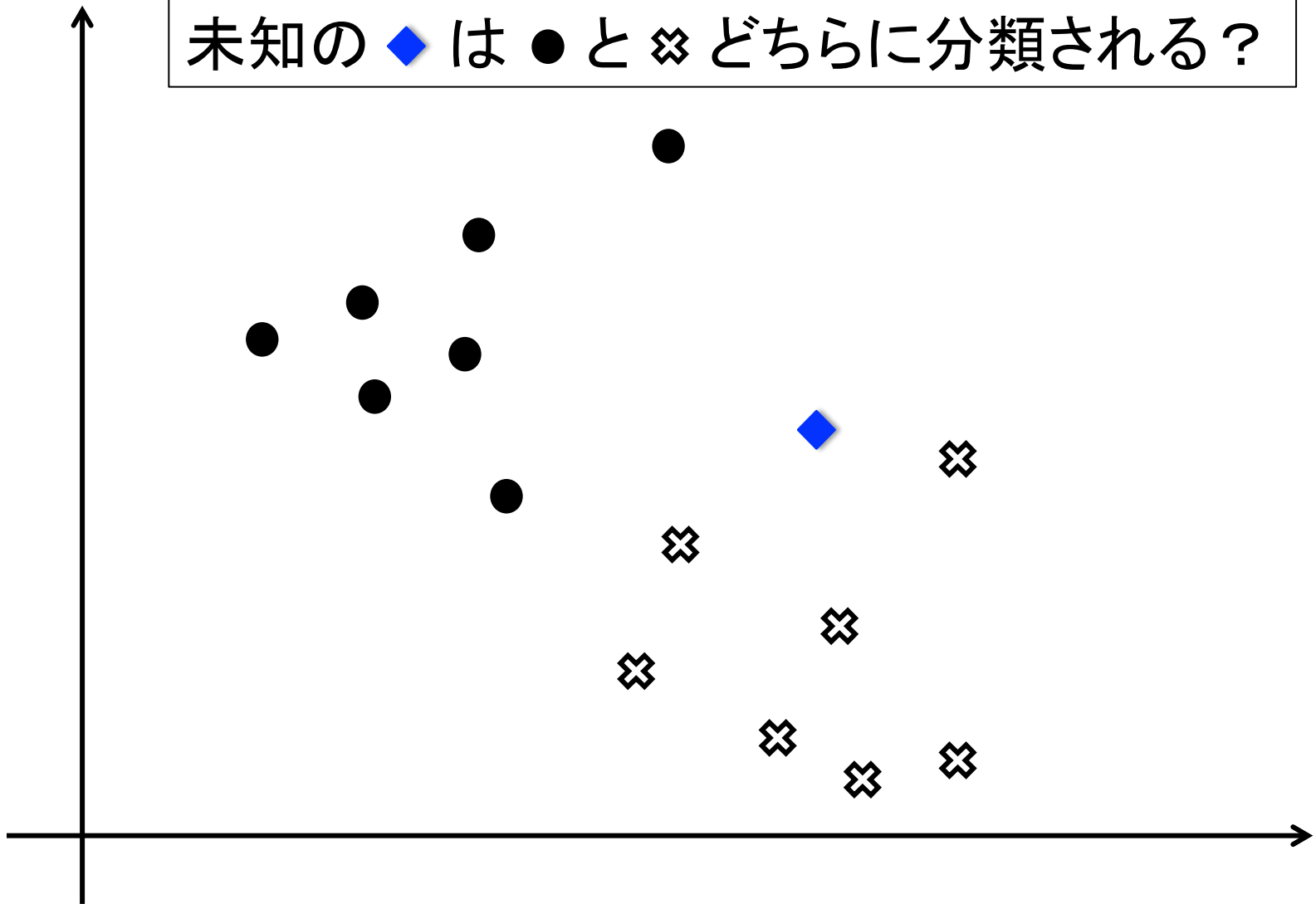
etc.



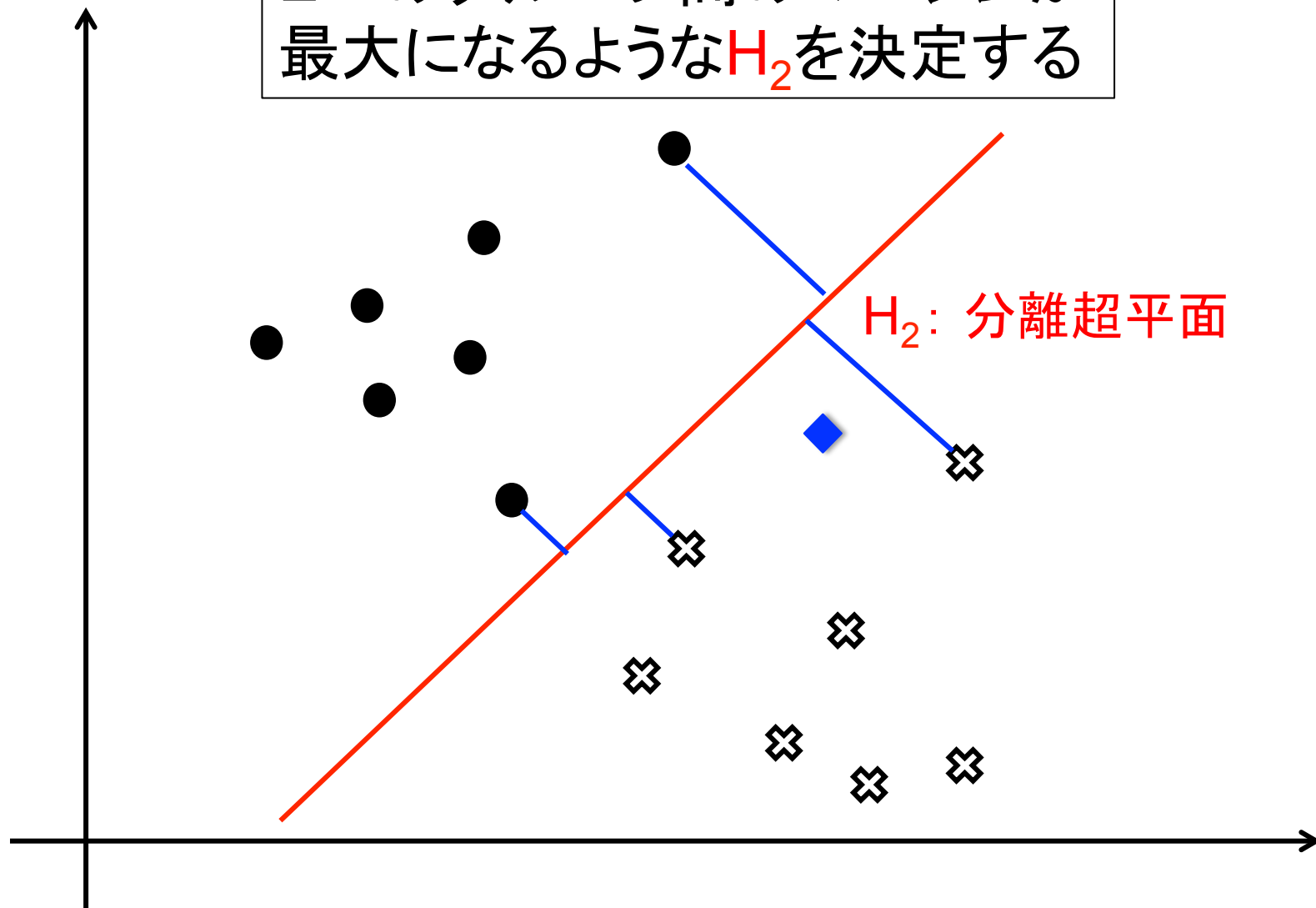
今回は
これを詳しく

2. サポートベクターマシン の理論

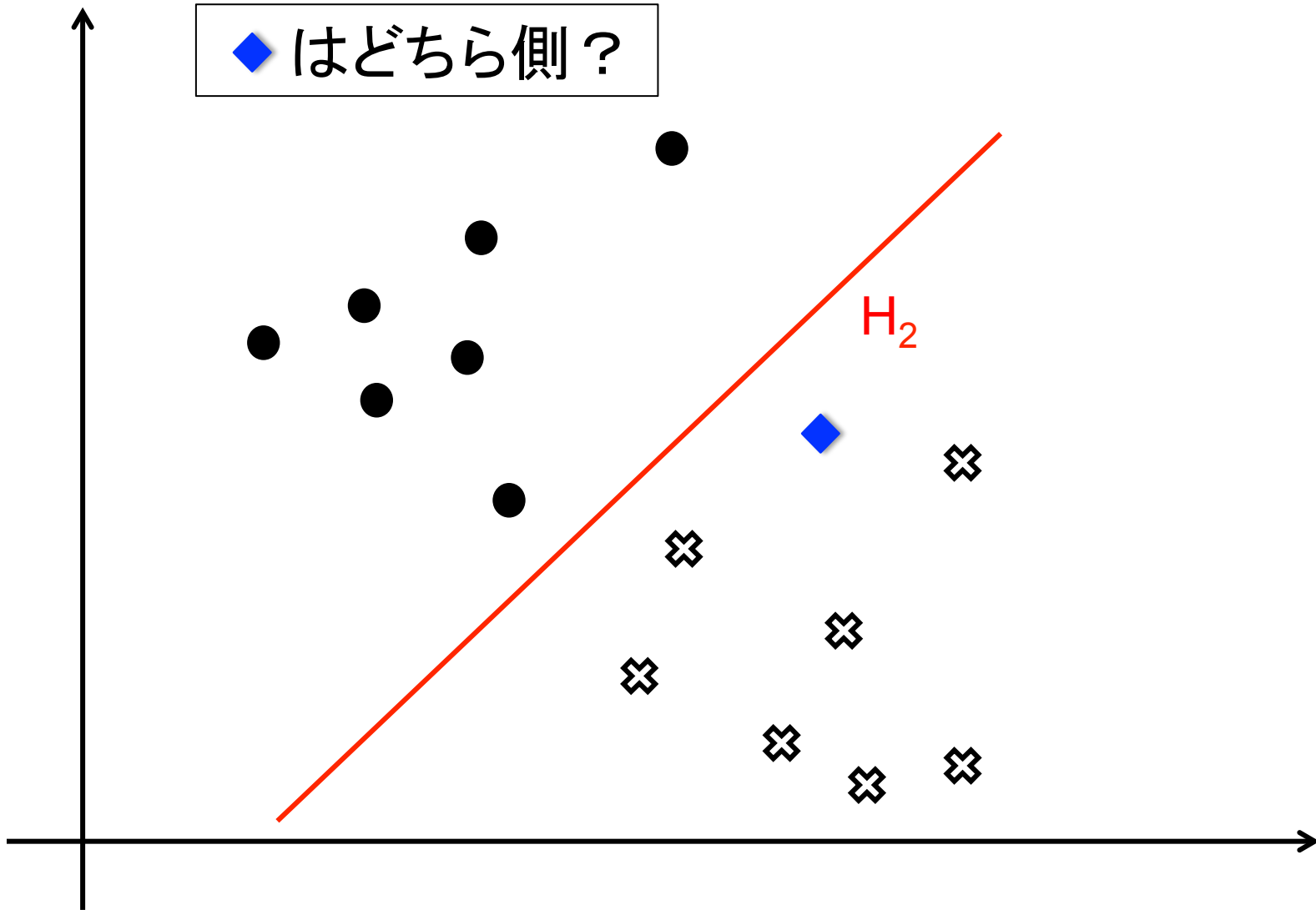
未知の ◆ は ● と ※ どちらに分類される？



2つのグループ間のマージンが最大になるような H_2 を決定する



◆ はどちら側？



サポートベクターマシンとは？

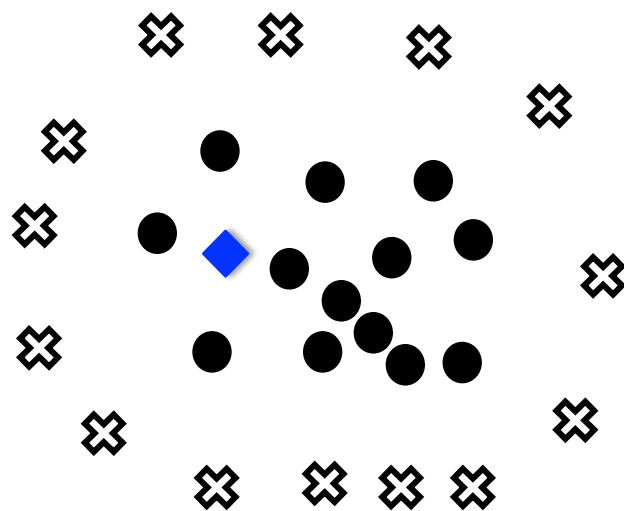
- 教師あり学習

- さっきの例では ● & ※ が学習データ
未知の ◆ を分類する

- マージン最大化学習を行う2値分類器

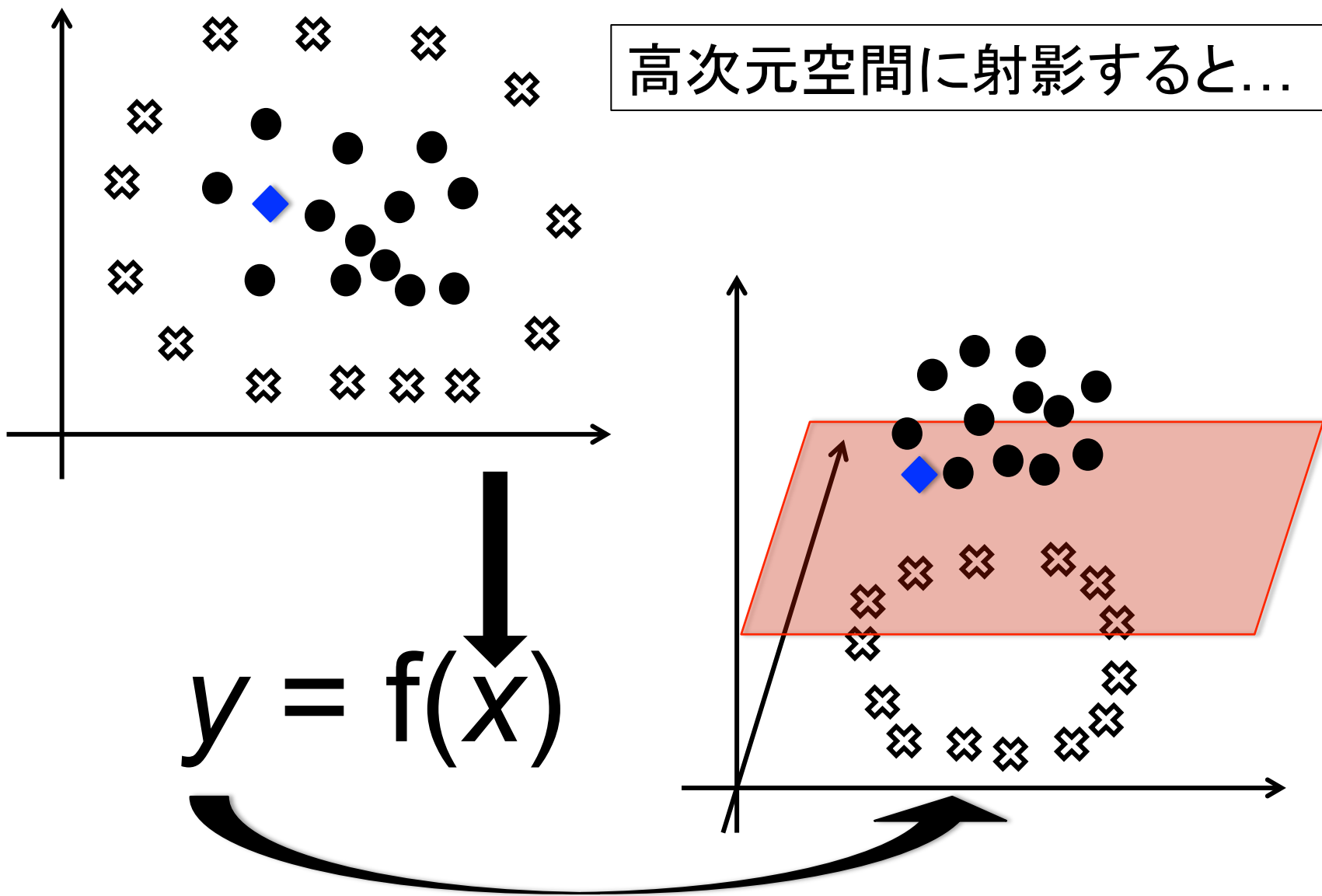
- 分離超平面を決定する
- 線形分類器(真っ直ぐなものしか切れない)
だったのだが...

Q: じゃあ、こういう場合どうするの？

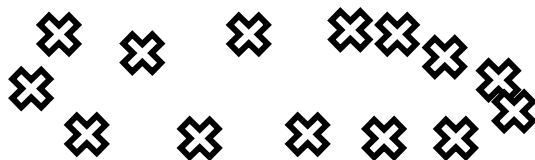
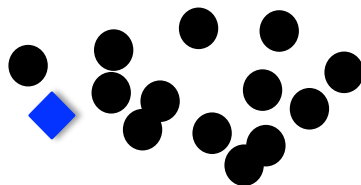


高次元空間に射影すると...

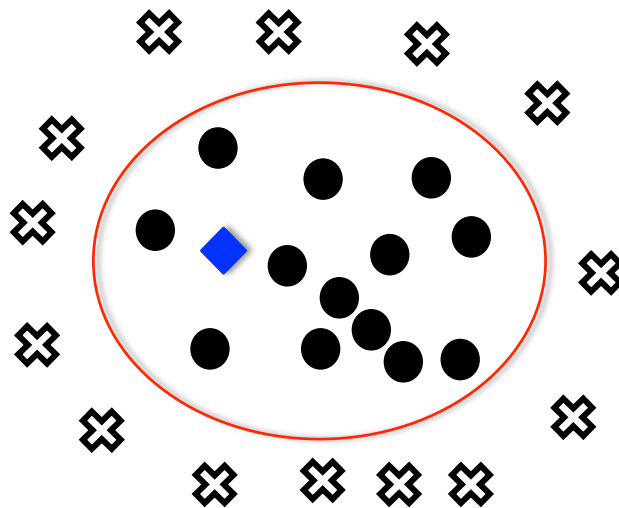
$$y = f(x)$$



A: 直線で切れた！



これと等価



カーネル法の話

- 非線形なカーネル関数により、非線形な識別関数を学習可能。
- カーネル関数を取り入れた一連の手法では、どのような写像が行われるか知らずに計算できる
⇒ カーネルトリック

さっきの図のようなことを勝手にやってくれる

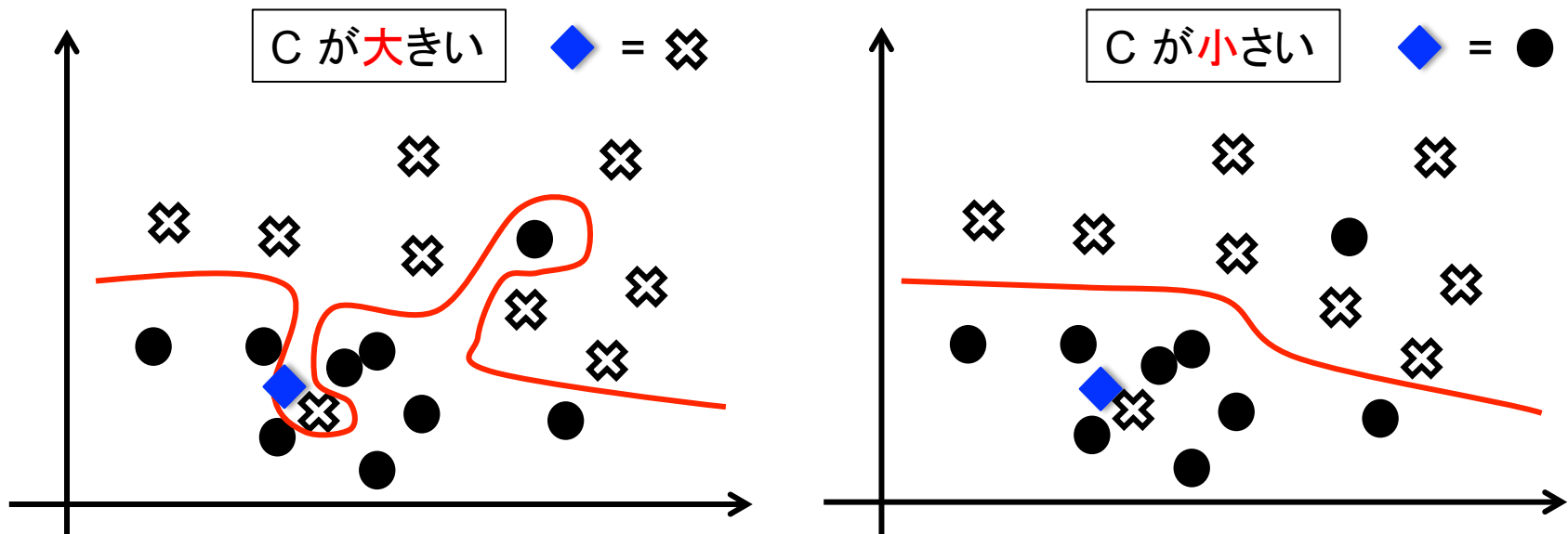
カーネル関数の例

- 線形
(Linear) $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- 多項式
(Polynomial) $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d$
($\gamma > 0$)
- RBF: “Gaussian”
(Radial Basis Function) $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$
($\gamma > 0$)
- シグモイド(Sigmoid) $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$

基本的に使うのは線形カーネルと、
非線形用のRBFカーネルぐらい。RBFの名前だけは覚えておこう。

マージンの話

- 実際に問題を解く際には、どの程度誤りを許容するかが問題となる。
 - コストパラメータ: C で決定する。



分類精度を向上させるためには適当さも必要

実際にSVMを
使いたいときは
これを自力実装？

SVMの実装例

- **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>

3. 最適化のお話

SVMのダメな使い方

1. データをSVMで使えるように整形。
2. デフォルトのパラメータで試行。
3. 「精度悪いなあ...」
4. 「パラメータの調整とやらで精度が上がるらしいぞ？」
5. 適当に選択したカーネルとパラメータで試行。
6. 「精度悪いなあ...別の手法試すか...」

ダメ！絶対！

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. カーネルの決定(基本:RBFカーネル)
5. 交差検定・グリッドサーチにより、
最適なコストやカーネルの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。

データの整形

- SVMに必要なデータ

- 学習用データ(多いほどよい)

- 「素性ベクトル(数値)」と「正解ラベル(数値)」のペア

v1: [[0.50, 0.33, -0.21],
 [0.12, 0.98, 1.34],
 ...]

l: [1,
 -1,
 ...]

- 本番用のデータ(いくつでもよい)

- 学習用に使用した素性ベクトルと同じ要素数からなる素性ベクトル

v2: [[0.23, 0.55, -0.19],
 [0.10, 0.24, 0.78],
 ...]



システムの出力
はこんな感じ

O: [1,
 -1,
 ...]

素性の選択(1)

- 数値データ
 - そのまま使用 [10.0, 2.5, 6.4, -8.2]
 - 範囲ごとに分割 [10, 0, 5, -10]
 - バイナリ化 [1, 1, 1, 0]
- テキストデータ
 - 単語の出現回数 (n-gram)
 - 品詞情報 等を数値化
- 画像・音声データ
 - 元データをそのまま行列からベクトルに
 - フィルタリング
 - 圧縮して単純化
 - フーリエ変換・ウェーブレット変換 等

素性の選択(2)

- 次元の呪い(curse of dimensionality)
 - 超高次元になるとモデルが複雑に
 - 学習データが不足
 - 球面集中現象
 - 次元の増加に伴って各データ間の距離が互いに等しくなる
- まとめられるものはまとめる
 - ⇒ 特徴選択・次元削減
 - Ex. 単語の出現回数
 - 類語をまとめてカウント

素性の選択(3)

- SVMは素性を実数値として扱う
 - n種類の値を取る素性
⇒ n個のバイナリ素性に
 - Ex. {red, green, blue}
 - [0], [1], [2] とせずに、[1,0,0], [0,1,0], [0,0,1]
としたほうが結果が安定する事が多い。

素性の選択は勘と経験がモノを言う

**⇒事前にどのような素性を選べば良い
のかしっかり調査するの大事！**

スケーリングの話

- 値のとりうる範囲が大きい素性が支配的に
 - 正規化したほうが良い結果になる場合も
Ex. $0 \leq x \leq 1$ or $-1 \leq x \leq 1$
- 情報落ち誤差を防ぐためにも必要
 - 基本的なカーネル関数では素性ベクトルの内積計算を用いる
⇒ スケーリングを行なわないと
誤差が発生する恐れあり

モデル選択の話

1. カーネル関数の決定

2. パラメータの決定

(コストパラメータ & カーネルパラメータ)

カーネル関数の決定(1)

- 最初に試すのはRBFカーネル
 - 事前知識がない場合これが無難
 - 高次元の非線形空間
 - 線形カーネルはRBFカーネルの特殊系
 - シグモイドカーネルもRBFカーネルとほぼ同じように動作
 - γ パラメータ + C パラメータ のみの調整で良い
- じゃ他のカーネルを使う場合はあるの？

カーネル関数の決定(2)

- 線形カーネルの利点
 - 高速な LIBLINEAR が使用出来る
 - Cパラメータのみの調整で良い
- 事前に線形分離できると予想できる場合
 - ⇒ 線形カーネルを用いるほうがいいのかも

カーネル関数の決定(3)

1. 事例数 \ll 素性数 の場合

- 素性が高次元なので写像する必要がない
- 線形カーネルを使うべき

2. 事例数 \gg 素性数 の場合

- 非線形カーネルを利用して高次元に写像すべき

3. 事例数も素性数も大きい 場合

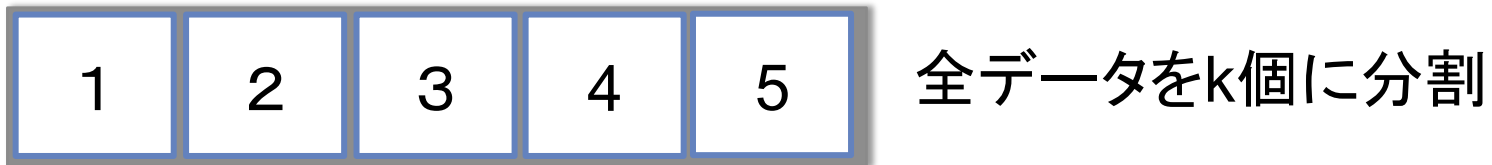
- 学習に時間がかかる。LIBSVMが苦手なケース
 - 線形カーネル & LIBLINEARの利用を検討

パラメータの決定

- RBFカーネルの場合
 - コストパラメータ: C
 - カーネルパラメータ: γ
- 線形カーネルの場合
 - コストパラメータ: C
- 最適なパラメータは？
 - ⇒ 交差検定・グリッドサーチ を利用して決定

交差検定

- 訓練データとテストデータの分割方法



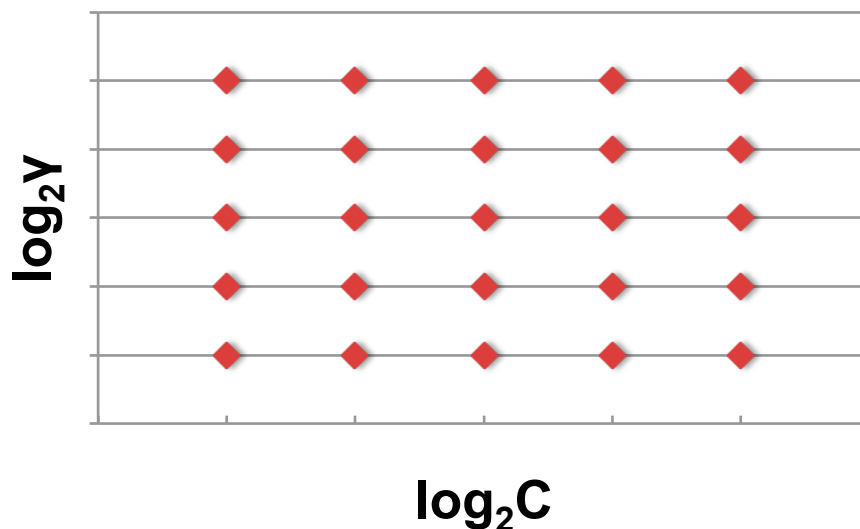
k回試行してその平均を利用

テストデータは常に未知のデータ

⇒ 過剰適応 (Over fitting) を防げる

グリッドサーチ

- 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. $C = 2^n$ ($n = -5 \sim 15$), $\gamma = 2^m$ ($m = -15 \sim 3$)



最適化のまとめ

- 素性の選択大事！
- 迷ったらRBFカーネル！
- パラメータ調整大事！

4. 金貨の真贋を 見分けよう

突然ですが問題！

週末に海賊船で催されたPRML読書会に参加したN君は、船内に山のように積まれた金銀財宝に目を奪われました。近くにあった宝箱の1つを開けてみると、きらきらと輝くコインが何枚も入っていました。

手に取ってみるとどれもずっしりと重みがあります。金貨に違いありません。好きなだけ持って帰ってよいと言われ、勉強会帰りに何枚か鞆に詰め込んで帰ることにしました。

家に帰ってから少し冷静になったN君は「気前よく配っていたけれど、この金貨は本物なのだろうか」と疑い始めました。

鞆には20枚の金貨が入っていましたが、友人のアルキメデスに計測してもらったところ、20枚とも体積も重さも異なりました。

ネットで検索してみたところ、金貨の体積と重さと真贋のデータを得られました。このデータを参考に、貰ってきた金貨の真贋を見分けてください。

http://next.rikunabi.com/tech/docs/ct_s03600.jsp?p=002315&tcs=kanren

コインのデータ

- 貰ってきた20枚のコインのデータ
 - 20行から成るテキストファイル
 - 1行にコイン1枚ずつ、そのコインの体積(cm^3)、重さ(g) が半角スペース区切りで記入されている。
- 金貨の体積と重さと真贋のデータ
 - 100行から成るテキストファイル
 - 1行にコイン1枚ずつ、そのコインの 体積(cm^3)、重さ(g)、真贋(本物なら '1', 偽物なら '0') が半角スペース区切りで記入されている。

コインのデータの詳細

- **CodeIQ_mycoins.txt**（貰ってきた20枚のコインのデータ）

0.988 17.734

0.769 6.842

0.491 4.334

0.937 16.785

0.844 13.435

...

- **CodeIQ_auth.txt**（金貨の体積と重さと真贋のデータ）

0.745 14.385 1

0.394 5.016 0

0.384 7.246 1

0.574 9.450 1

0.603 8.198 0

...

R言語について

今回はR言語を使います

R言語のいいところ

- 統計学者が作った言語

R言語のわるいところ

- 統計学者が作った言語

準備

必要なパッケージのインストール

```
install.packages("e1071")  
install.packages("ggplot2")
```

データの読み込みと確認

```
coins.auth = read.table("./CodeIQ_auth.txt",header=F,sep="")  
names(coins.auth) <- c("volume","weight","truth")  
coins.auth$truth <- factor(coins.auth$truth)  
print(head(coins.auth))
```

```
coins.my = read.table("./CodeIQ_mycoins.txt",header=F,sep=" ")  
names(coins.my) <- c("volume","weight")  
print(head(coins.my))
```


データの概要のチェック

summary 関数の適用(真贋それぞれ別に)

```
print(summary(coins.auth[coins.auth$truth == 0,]))
```

```
print(summary(coins.auth[coins.auth$truth == 1,]))
```

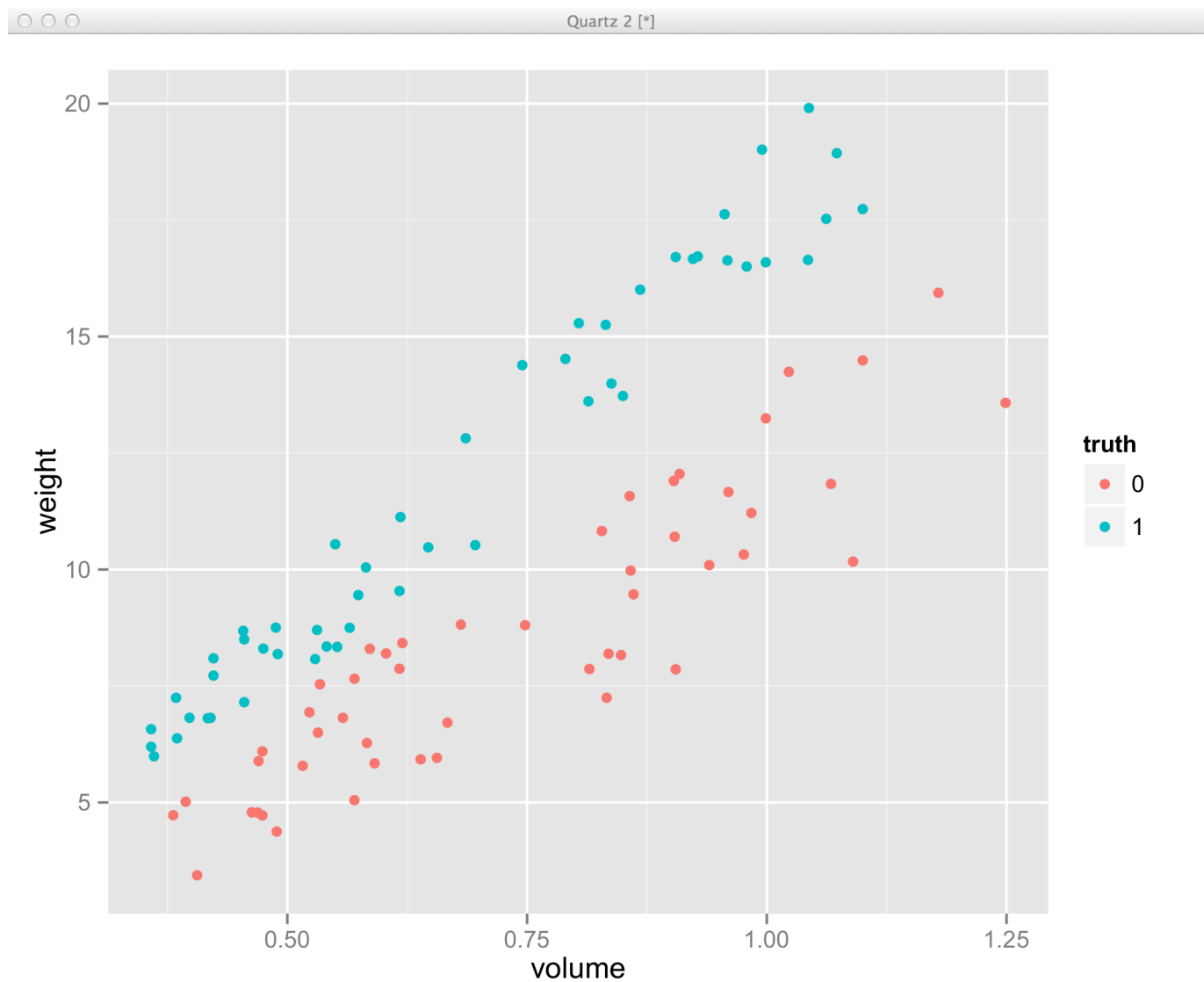
CodeIQ_auth.txt をグラフ化

```
graph = ggplot(coins.auth,aes(x=volume,y=weight))
```

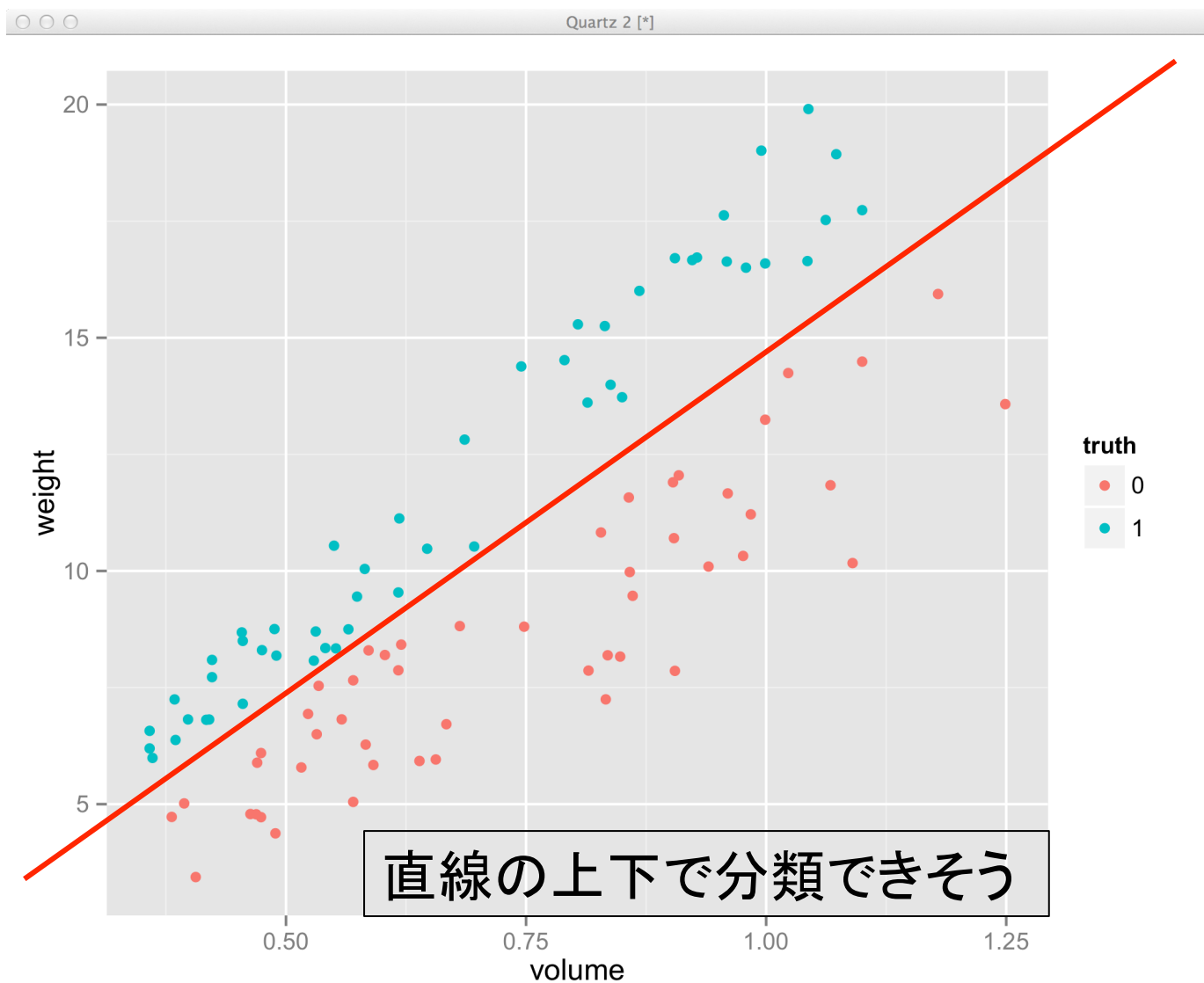
```
+ geom_point(aes(color=truth))
```

```
print(graph)
```

真贋の境界は...



こんな感じ...？



今回の場合

- 学習用データ
 - 素性(識別に使う情報: Feature)
 - Volume
 - Weight
 - 正解ラベル
 - 真贋値 (0 or 1)
- 学習させた SVM に Volume 、Weight からなる2次元の素性ベクトルを渡せば、真贋(ラベル)を判別してくれる。
- 線形分離可能っぽいので、**線形カーネル**を使用してみる。

2次元の素性ベクトル

というわけで**SVM**の準備

SVM のモデルを作成してみる

```
model <- svm(  
  truth ~ ., data = coins.auth, # coins.auth の truth を他の変数から予測  
  kernel = "linear", # 線形カーネル  
  cross = 5 # 交差検定の分割数  
)  
  
print(summary(model)) # モデルの確認
```

結果

Call:

```
svm(formula = truth ~ ., data = coins.auth, kernel = "linear", cross = 5)
```

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1

gamma: 0.5

Number of Support Vectors: 32

(16 16)

Number of Classes: 2

Levels:

0 1

5-fold cross-validation on training data:

Total Accuracy: 100

Single Accuracies:

100 100 100 100 100

やたー

精度: Accuracy 100 %

達成できたよー (棒)

用意した問題が簡単すぎましたね...

最適化するなら(一応掲載)

最適化

```
tune = tune.svm(  
  truth ~ ., data = coins.auth,  
  kernel = "radial",          # RBF カーネルを指定  
  gamma=2^(seq(-15, 3)),  
  cost=2^(seq(-5, 15)),  
  tunecontrol=tune.control(  
    sampling="cross",  
    cross=10)  
)  
  
cat("- Best parameters:\n")  
cat("Gamma =", tune$best.parameters$gamma,  
  "; Cost =", tune$best.parameters$cost, ";\n")  
cat("Accuracy:", 100 - tune$best.performance * 100, "%\n\n")
```


最適化の結果

- Best parameters:

Gamma = 4 ; Cost = 0.5 ;

Accuracy: 100 %

予測させてみる

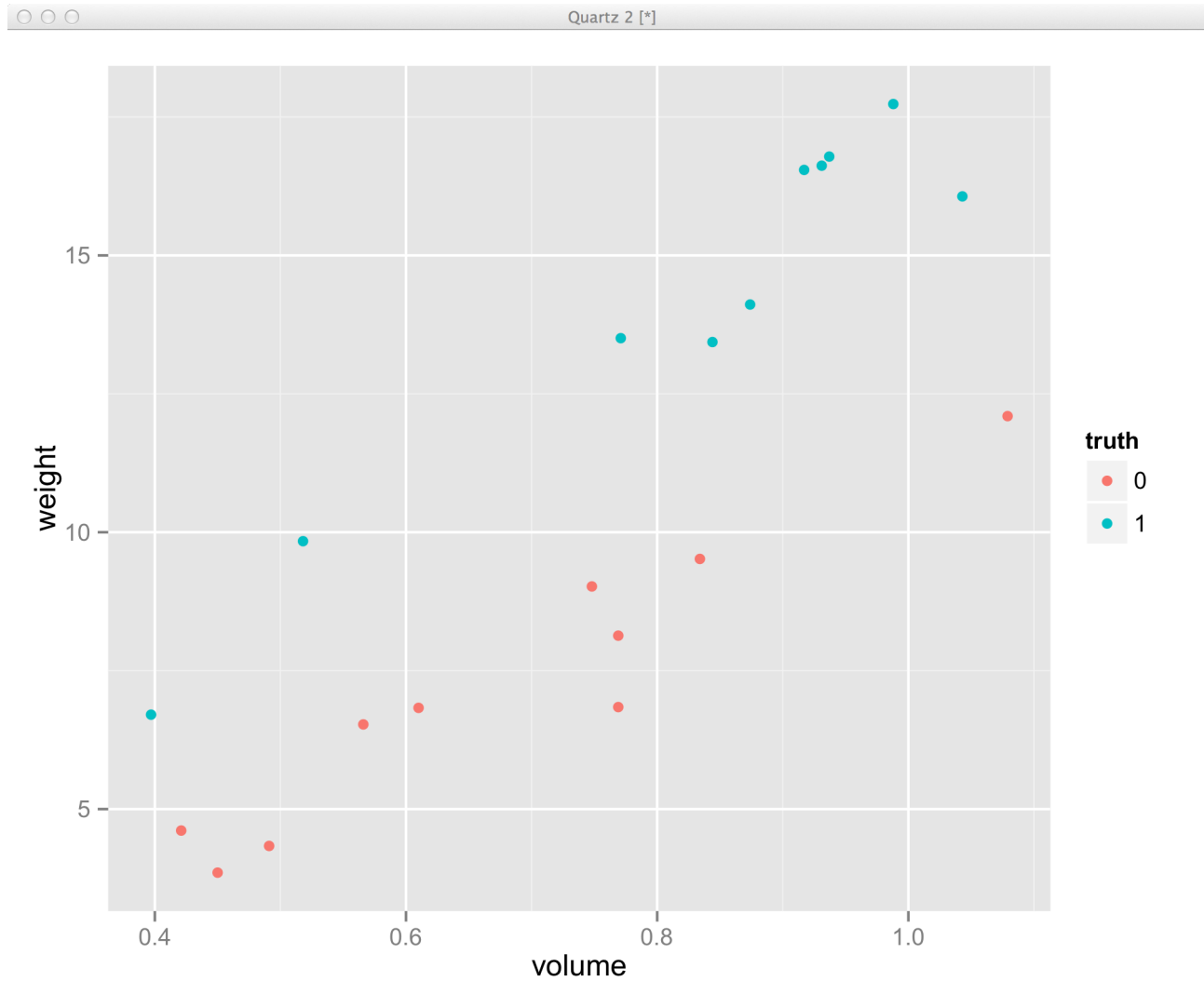
作成したモデルで分類してみる

```
model <- svm(  
  truth ~ ., data = coins.auth, # coins.auth の truth を他の変数から予測  
  kernel = "linear" # 線形カーネル  
)  
coins.my$truth <- predict(model, coins.my)
```

グラフ化

```
graph = ggplot(coins.my, aes(x=volume, y=weight))  
  + geom_point(aes(color=truth))  
print(graph)
```

識別できた！



まとめ

**機械学習を使えば、
いろいろな物を
自動で識別できるよ！**

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

- ・カーネル法

<http://www.eb.waseda.ac.jp/murata/research/kernel>

- ・TAKASHI ISHIDA HomePage SVM

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

- ・「機械学習基礎」簡単な問題を解いて理解しよう！ 前篇

http://next.rikunabi.com/tech/docs/ct_s03600.jsp?p=002315&tcs=kanren

- ・金貨の真贋を見分けよう(サンプルコード)

https://github.com/Salinger/svm_gold