

# データサイエンス

## 第9回

~クラス分類(3)・カーネル法~

情報理工学系研究科  
創造情報学専攻  
中山 英樹

# 非線形識別（回帰も同様）

- 大きく分けると二通りのアプローチ
  - 元の特徴空間で直接非線形の識別関数を学習する
    - 空間を局所分割する：k最近傍法、局所線形回帰、決定木
    - アンサンブル学習
    - ニューラルネットワーク
  - 非線形な特徴変換を行った後、線形識別関数を学習
    - カーネル法
    - Explicit Feature Maps

# アンサンブル学習

- 複数の(性能が低い)識別器を組み合わせてより強力な識別を行うアプローチの総称
- 「三人よれば文殊の知恵」

$$\text{混合モデル} \quad p(y | \mathbf{x}) = \sum_{k=1}^K \pi_k(\mathbf{x}) p_k(y | \mathbf{x})$$

$$\text{コミッティ (多数決)} \quad y_{COM}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K y_k(\mathbf{x})$$

- 分類に限らず、回帰等さまざまな問題へ適用可能な概念
  - c.f. コンピュータ将棋

# Bagging (**B**ootstrap **A**ggregating)

- ブートストラップ法により学習データのサブセットを繰り返し抽出し、それぞれ弱識別器を構築
- 決定木分類器(後述)とよく合わせて用いられる

1. For  $k = 1, \dots, K$

(a)  $n$ 個の学習データ  $\{\mathbf{x}_i, y_i\}_{i=1}^n$  から、重複を許してランダムに $n$ 個を抽出

(b) 抽出したサンプルを用い、弱識別器  $f_k$  を学習

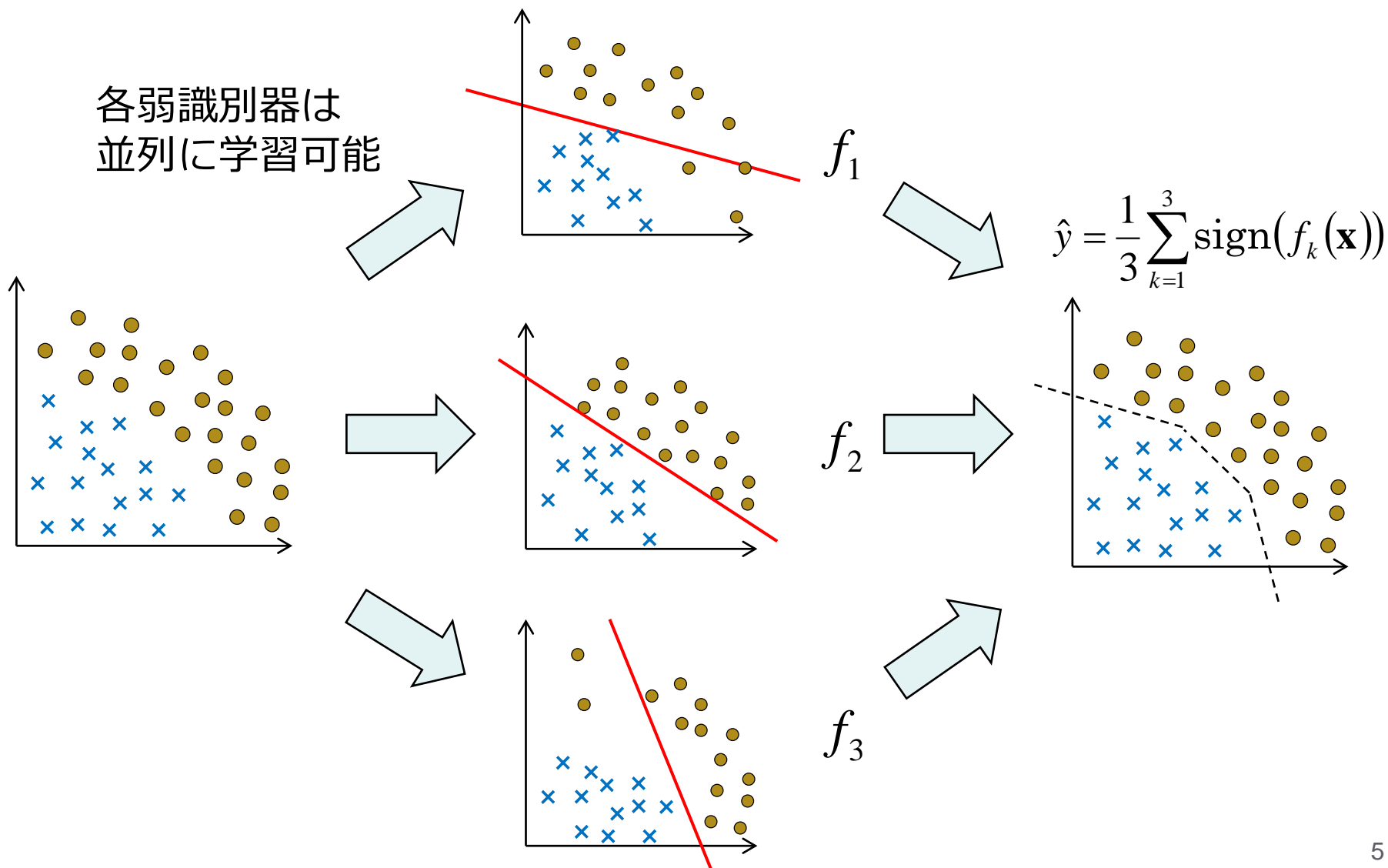
2. 全ての弱識別器  $\{f_k\}_{k=1}^K$  の平均を強学習器として出力

$$f(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K f_k(\mathbf{x})$$

- 識別関数の場合、要は多数決

# Bagging

各弱識別器は  
並列に学習可能

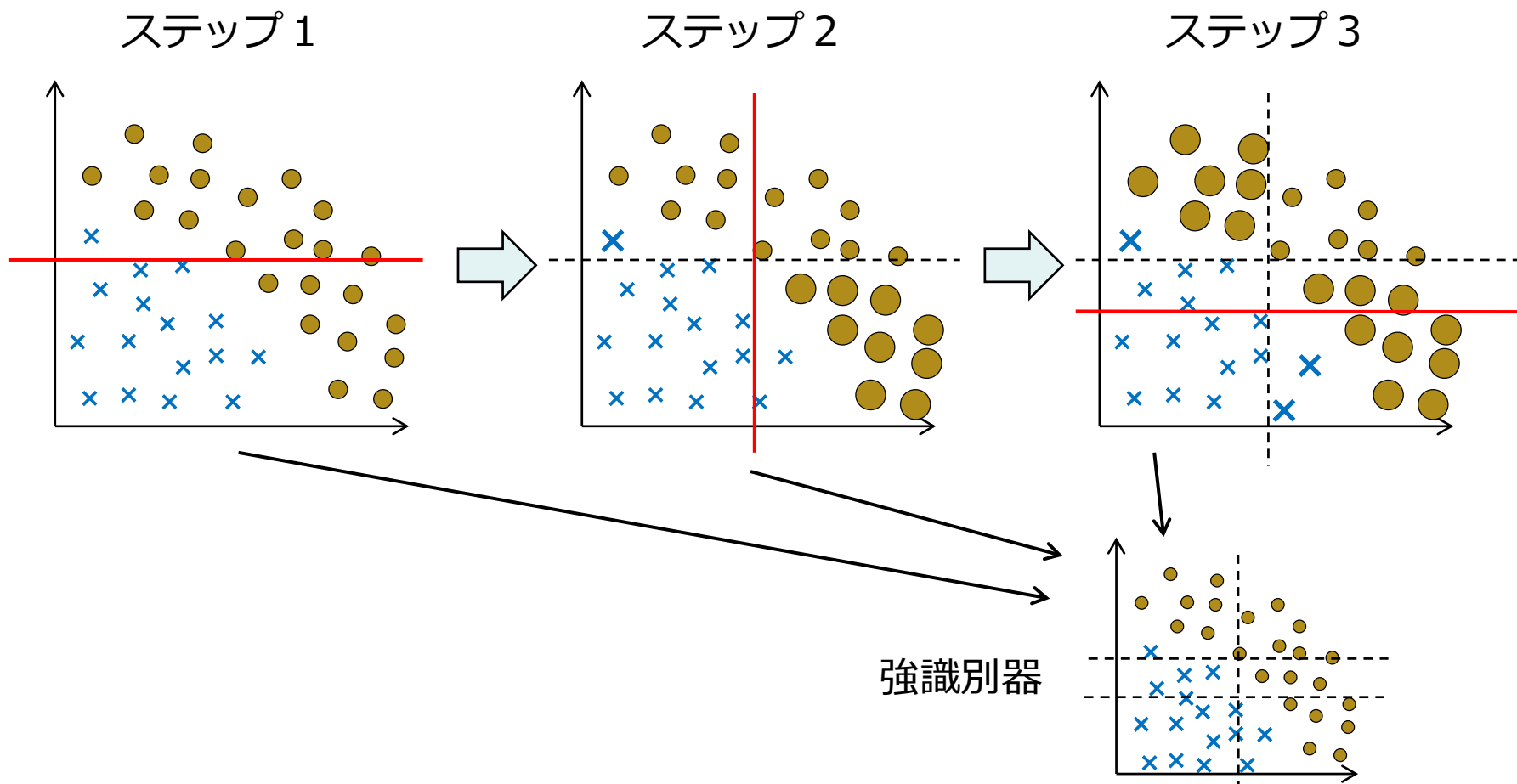


# Boosting

- 複数の弱識別器を順々に学習
  - それまでに学習した弱識別器がうまく分類できない、難しい学習データに強い重みをつけ次の弱識別器を学習
  - 最終的に、何らかの信頼度で重み付平均した結果を出力
- Baggingよりもよい識別性能を示すことが多い
- AdaBoost(Adaptive boosting)と呼ばれる手法が最も一般的
  - Y. Freund and R. E. Schapire. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

# AdaBoost

- 弱識別器を順々に学習



# AdaBoostのアルゴリズム

$\{\mathbf{x}_i, y_i\}_{i=1}^n, y_i \in \{-1, +1\}$  : 学習データセット

$W_k(i)$  : k番目の弱識別器に与える各データの重み

- 重みの初期化  $W_1(i) = 1/n$
- For  $k = 1, \dots, K$ 
  - $W_k$ に基づいて弱識別器  $f_k$ を学習
  - 識別器の信頼度  $\alpha_k = \frac{1}{2} \log \left( \frac{1 - \varepsilon_k}{\varepsilon_k} \right)$  重み付経験誤差
  - サンプル重みの更新  $W_{k+1}(i) = \frac{1}{Z_k} W_k(i) \exp(-\alpha_k y_k f_k(\mathbf{x}_i))$   
正規化定数 ←
- Output:  $f(\mathbf{x}) = \text{sign} \left\{ \sum_{k=1}^K \alpha_k f_k(\mathbf{x}) \right\}$  ← 信頼度で重み付けた多数決



# BaggingとBoosting：まとめ

	Bagging	Boosting
弱識別器の学習	並列	逐次的
サンプルへの重みづけ	ブートストラップによるサブセット抽出	誤識別に基づき重みを更新
識別	単純な多数決	信頼度による重み付け多数決

# 議論

- どんな弱識別器を使うべき？
- そもそも何故集団学習はうまくいく？

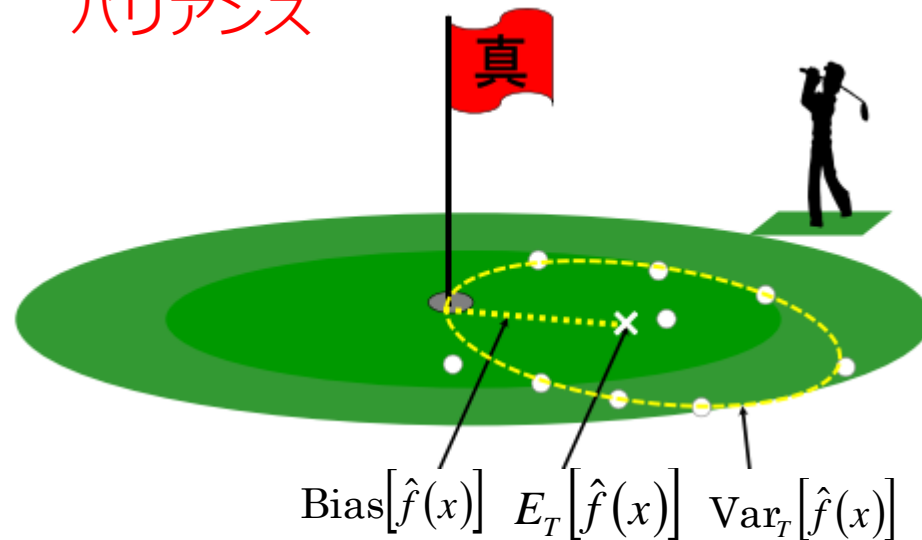
# バイアス-バリエンス分解

$f(x)$  : 真の分布       $\hat{f}(x)$  : あるモデルのもとで、有限の訓練データ  $T$  から推定した分布

$$\underline{E\left[\left(y - \hat{f}(x)\right)^2\right]} = \sigma^2 + \left(E_T\left[\hat{f}(x)\right] - f(x)\right)^2 + E_T\left[\left(\hat{f}(x) - E_T\left[\hat{f}(x)\right]\right)^2\right]$$

$$\text{二乗汎化誤差} = \sigma^2 + \underbrace{\text{Bias}^2\left[\hat{f}(x)\right]}_{\text{バイアス}} + \underbrace{\text{Var}_T\left[\hat{f}(x)\right]}_{\text{バリエンス}}$$

[http://www.nii.ac.jp/userdata/karuizawa/h23/111104\\_3rdlecueda.pdf](http://www.nii.ac.jp/userdata/karuizawa/h23/111104_3rdlecueda.pdf) より引用



# バイアス-バリエーション分解

- バイアス：仮定したモデルの、真の構造との本質的なずれ
- バリエーション：仮定したモデルの下での、訓練サンプルの違いに起因する推定結果のばらつき
- バイアス・バリエーションはトレードオフの関係
  - 線形など単純なモデルは、バイアス大、バリエーション小
  - 高次の複雑なモデルは、バイアス小、バリエーション大
- 適切な複雑さのモデルを選ぶ必要がある

# アンサンブル学習の効果

- 識別器のバリエーションを減らす
- Baggingの場合

$$E_T \left[ E_k \left\{ (y - f_k(x))^2 \right\} \right] = E_T \left[ (y - \bar{f}(x))^2 \right] + \underbrace{E_T \left[ E_k \left\{ (\bar{f}(x) - f_k(x))^2 \right\} \right]}_{\geq 0}$$

$$\therefore \underbrace{E_T \left[ (y - \bar{f}(x))^2 \right]}_{\text{平均識別器の誤差}} \leq \underbrace{E_T \left[ E_k \left\{ (y - f_k(x))^2 \right\} \right]}_{\text{もとの識別器の誤差}}$$

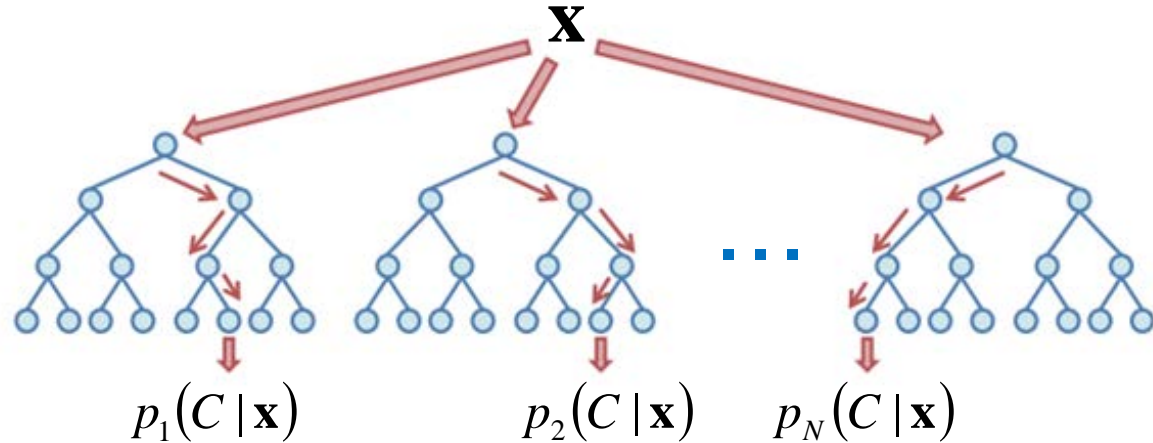
- Boostingの場合も同様
  - 誤差の大きい点での重みを増やし、より積極的にバリエーションを減らす

# 弱識別器のモデル

- 低バイアス・高バリエーションが理想
  - SVM等は低バリエーションのモデルなので、あまり集団学習の効果が得られない
  - 決定木（後述）などが代表的な弱識別器
- 直感的には
  - 一つ一つの識別器はデータに対して過学習しても、たくさん作って平均をとればいいところに落ち着く

# ランダムフォレスト

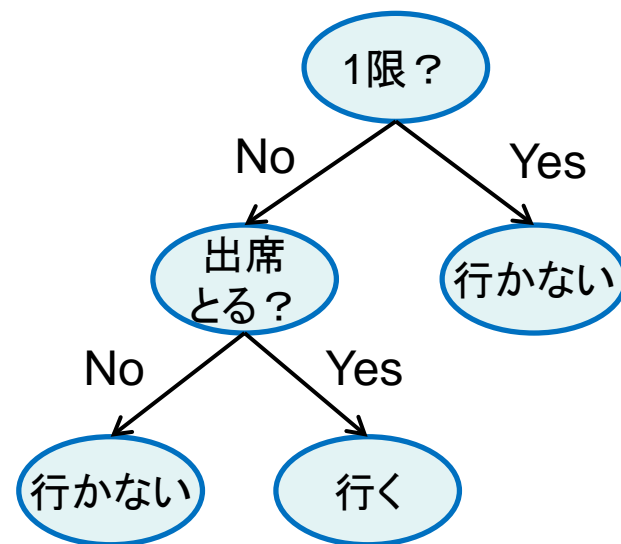
- 決定木を弱識別器としたバギング
  - ただし、特徴量・説明変数も決定木ごとにランダムにサブセットを選ぶ



- 識別以外にも回帰、クラスタリングなどに広く応用されている
- SVMと並んで(?)人気のある識別手法

# 決定木

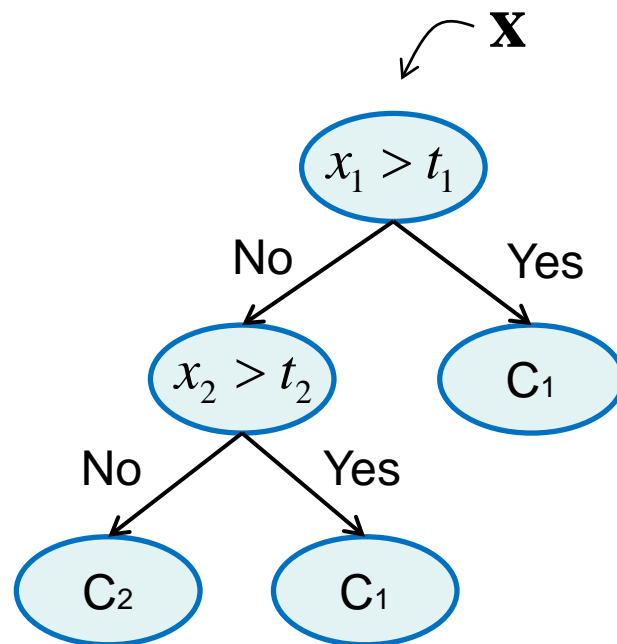
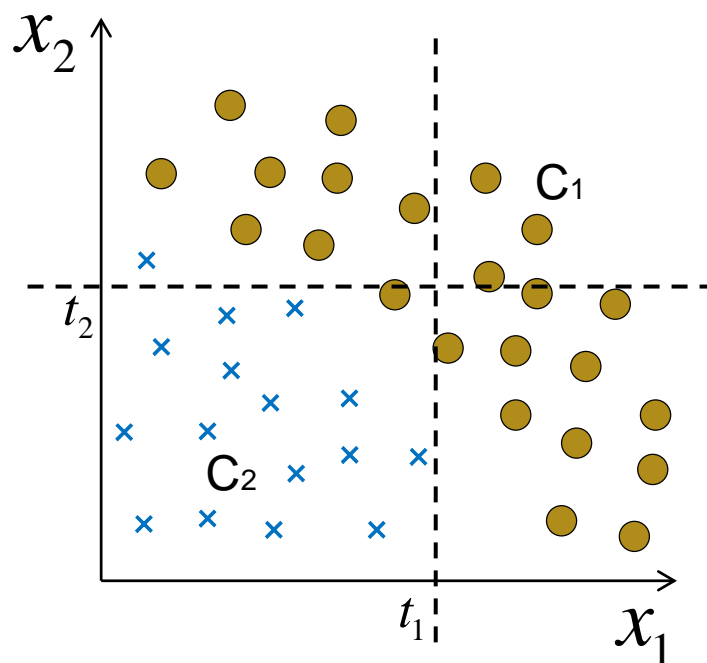
- 分類や意思決定の過程を樹形図で表現したもの
  - 葉ノード（末端）：目的変数の属性
  - 葉ノード以外：分類のルール
- 大きくわけて二種類
  - 分類木：葉ノードがカテゴリ
  - 回帰木：葉ノードが数値





# 決定木による分類

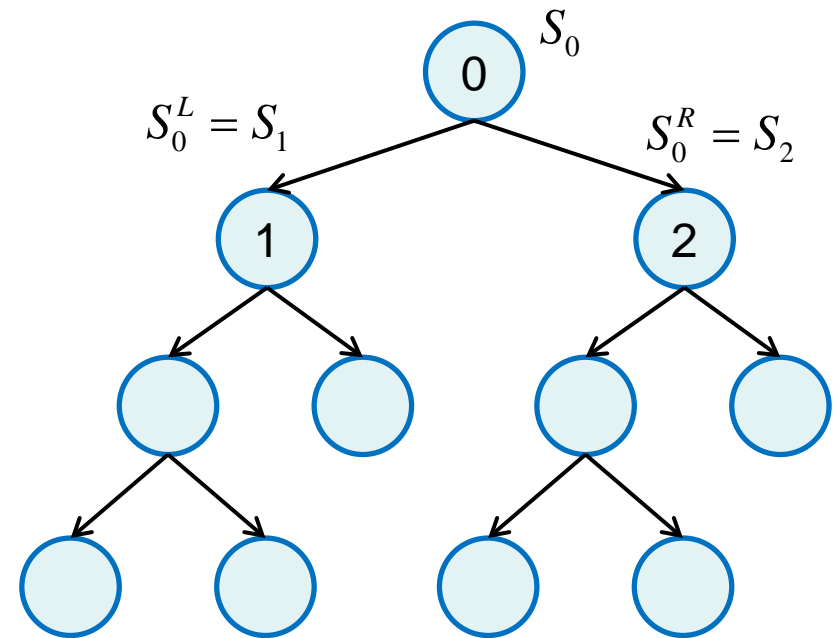
- 識別結果の解釈がしやすい
- 各特徴要素の重要度が評価できる



# 決定木学習の流れ

$S_i$  : あるノードに含まれる学習データ集合 (サンプル)  
(元の空間の部分領域に対応)

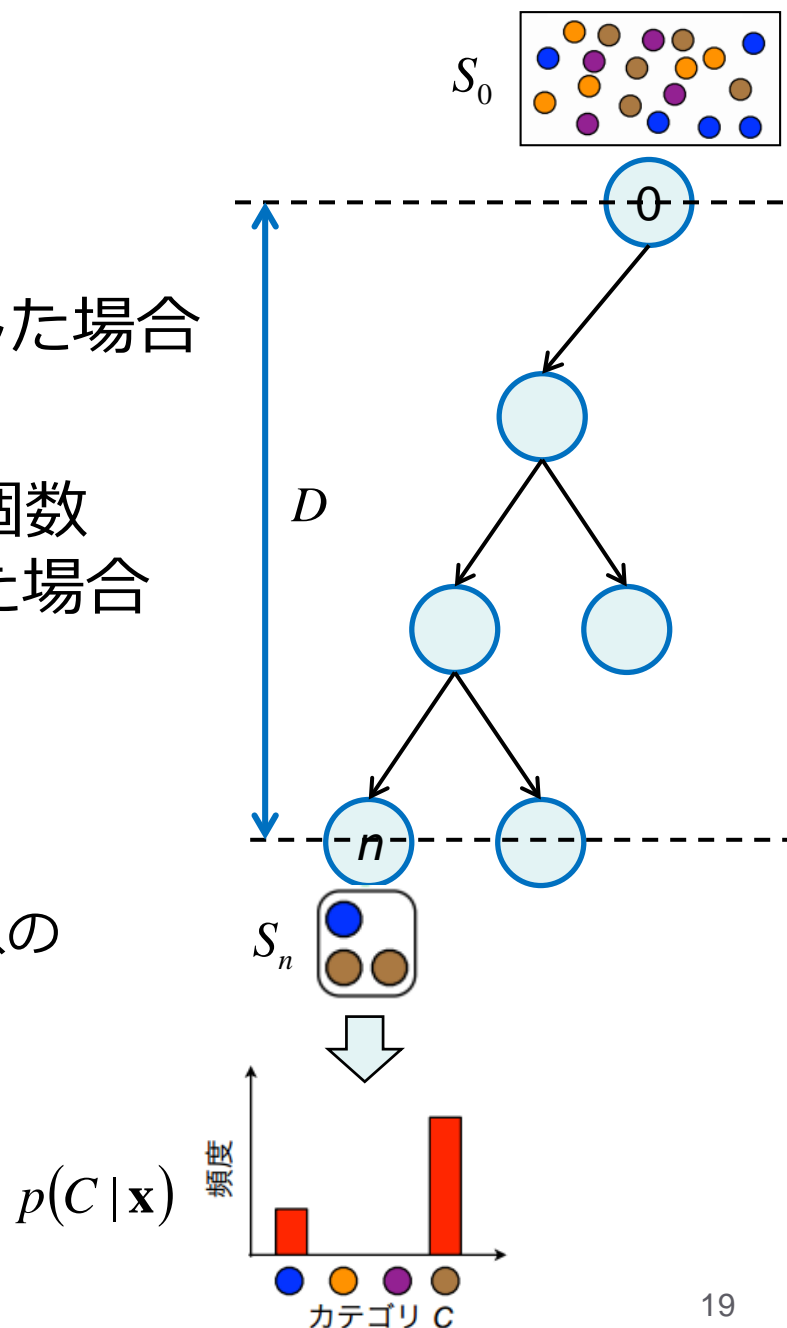
- Rootノードから再帰的に以下を実行
  - ノード  $i$  での分岐関数を求める
    - $S_i$  を最もよく分離するように
    - 分岐関数に従い、サンプルを二つに分割し、配下のノードを作成
- 全葉ノードで終了条件を満たせば終了



# 葉ノードの終了条件

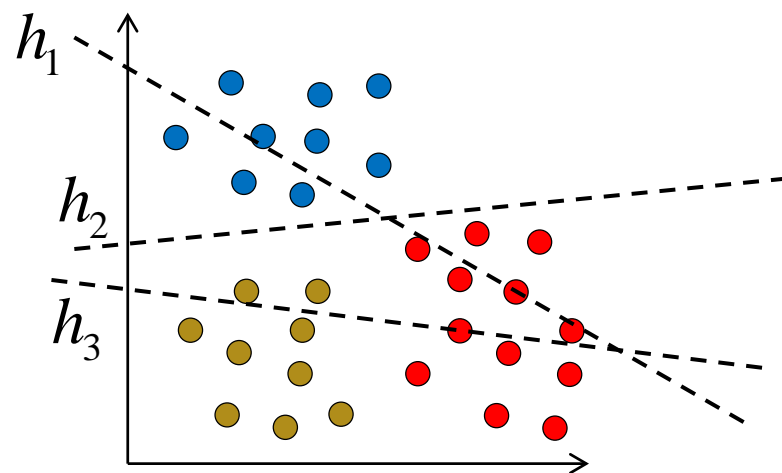
過学習を抑制

- 1. 予め定めた深さ $D$ に到達した場合
- 2. ノード内の学習データの個数 $|S_n|$ が一定値以下になった場合
- 3. 分割による情報利得が一定値以下になった場合
- 例えば、 $S_n$  がほとんど同じクラスのデータのみになった時



# 分岐関数の学習

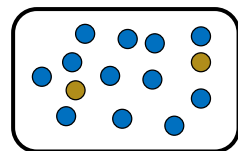
- 情報量を基準に複数の候補から選択
  - 情報利得、エントロピー (C4.5)
  - ジニ係数 (CART)
- あるいは単純にランダム分割
  - Random forest 前提
  - Extreme Randomized Trees [P. Geurts, 2006]



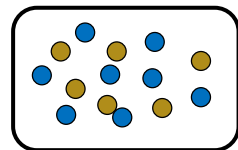
# 情報利得による分割

- エントロピー（平均情報量）
  - クラスの生成確率（つまりS中のデータ数の比）に偏りがあるほど小さい値となる
  - あるノード内サンプルの分割の良さに対応

$$H(S) = - \sum_{c \in C} p(c) \log(p(c))$$

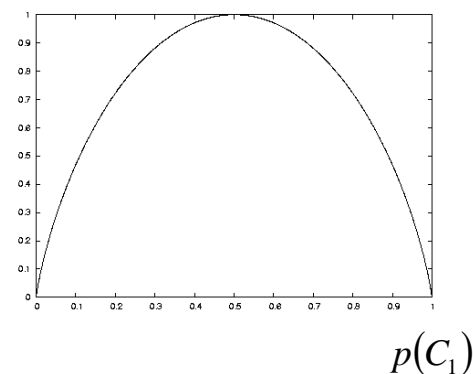


$H(S)$ : 小



$H(S)$ : 大

$H(S)$ （2クラスの例）



## ● 情報利得

- 分割前後のエントロピーの差

$$I = \underbrace{H(S)}_{\text{分割前}} - \sum_{i \in \{L, R\}} \underbrace{\frac{|S^i|}{|S|} H(S^i)}_{\text{分割後 (データ数で重み付け)}}$$

分割後 (データ数で重み付け)

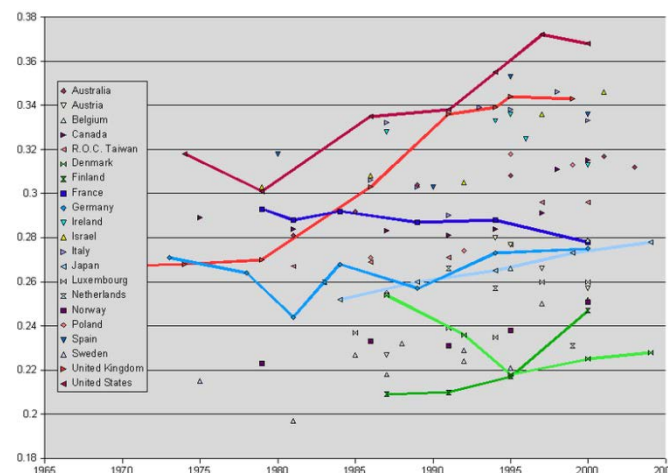
# ジニ係数による分割 (CART)

$$Gini\ Index(S) = 1 - \sum_{c \in C} p^2(c)$$

$p(c)$  は、ノード中でクラス  $c$  に属するデータの割合とする

- もともとは、所得分配の不平等さを示す経済学の指標

各国のジニ係数の推移



<http://ja.wikipedia.org/wiki/%E3%82%B8%E3%83%8B%E4%BF%82%E6%95%B0>

# 参考

- Top 10 algorithms in data mining [@IEEE ICDM 2006]
  - 1: C4.5
  - 2: K-Means
  - 3: SVM
  - 4: Apriori
  - 5: EM
  - 6: PageRank
  - 7: AdaBoost
  - 7: k nearest neighbor
  - 7: Naive Bayes□
  - 10: CART

# Rの例 (関数rpart)

- デフォルトではジニ係数による分割  
(エントロピーの場合はsplit="information"を指定)

```
> library("mvpart")
> set.seed(20)
> iris.rp<-rpart(Species~.,data=iris)
> print(iris.rp,digit=1)
n= 150
```

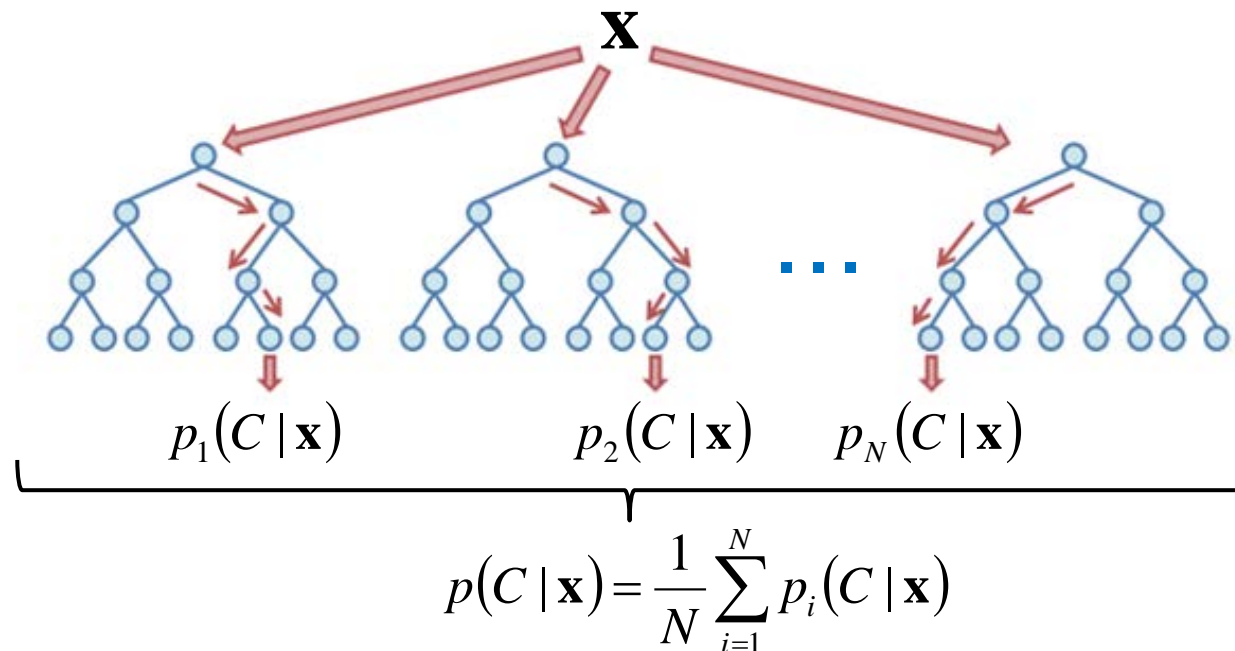
```
node), split, n, loss, yval, (yprob)
* denotes terminal node
```

```
1) root 150 100 setosa (0.33 0.33 0.33)
 2) Petal.Length< 2 50 0 setosa (1.00 0.00 0.00) *
 3) Petal.Length>=2 100 50 versicolor (0.00 0.50 0.50)
 6) Petal.Width< 2 54 5 versicolor (0.00 0.91 0.09)
 12) Petal.Length< 5 48 1 versicolor (0.00 0.98 0.02) *
 13) Petal.Length>=5 6 2 virginica (0.00 0.33 0.67) *
 7) Petal.Width>=2 46 1 virginica (0.00 0.02 0.98) *
```



# ランダムフォレスト

- 決定木を用いたバギング
- ブートストラップ法で学習サンプルのサブセットを生成し、それぞれ識別器を構築（特徴もランダムに選択）
- オーバーフィッティングしやすいので、データは大量に必要



# Rの例 (関数randomForest)

```
> library("randomForest")
> set.seed(20)
> iris.rf<-randomForest(Species~.,data=iris)
> print(iris.rf,digit=1)
```

Call:

```
randomForest(formula = Species ~ ., data = iris)
```

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 4.67%

Confusion matrix:

	setosa	versicolor	virginica	class.error
setosa	50	0	0	0.00
versicolor	0	47	3	0.06
virginica	0	4	46	0.08

# 特徴の重要度

- 当該特徴を除いた時のジニ係数の減少分で評価

```
> importance(iris.rf,scale=TRUE)
```

```
      MeanDecreaseGini
```

```
Sepal.Length      9.993809
```

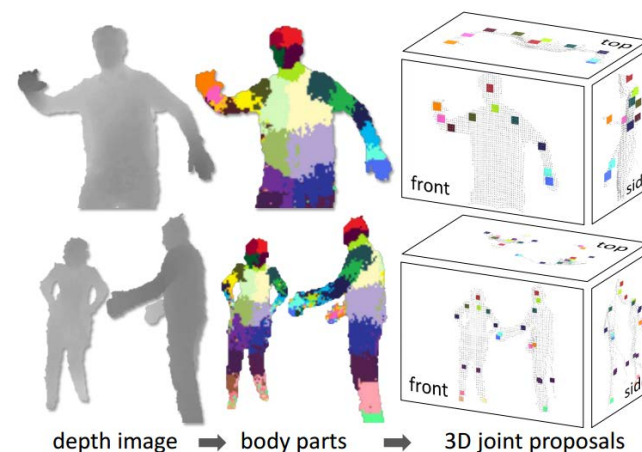
```
Sepal.Width       2.180028
```

```
Petal.Length     43.362837
```

```
Petal.Width      43.795833
```

# Kinect論文

- 人体のパーツ（手、肩、足、etc.）の識別にランダムフォレストを利用
- Xboxの人体姿勢推定アルゴリズム



- 学習 : 3 trees, 20 depths, 1 million samples  
→ “a day on 1000-core clusters”
- 認識 : 200fps on a consumer hardware

J. Shotton et al., “Real-Time Human Pose Recognition in Parts from Single Depth Images”, In Proc CVPR, 2011 **[best paper]**

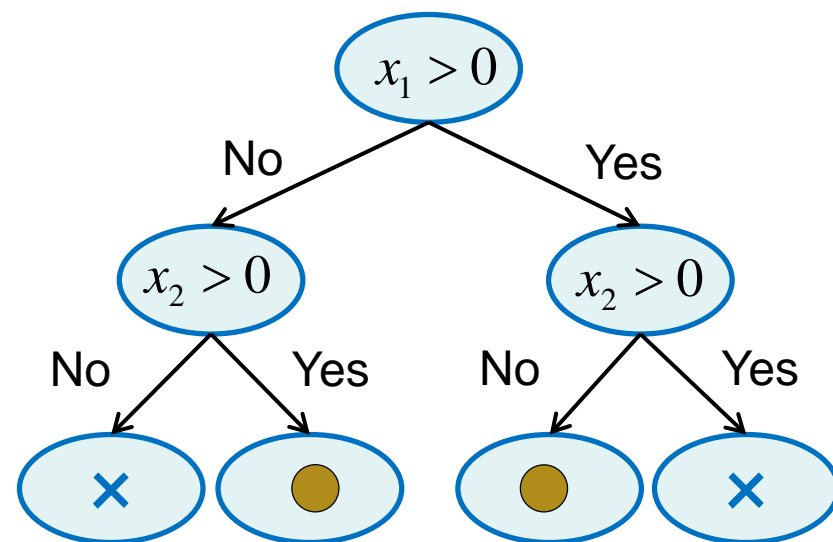
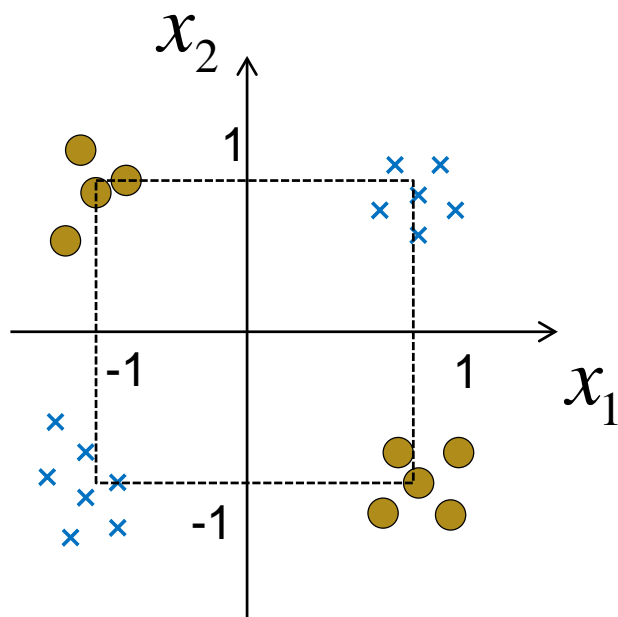
# 非線形識別

- 大きく分けると二通りのアプローチ
  - 元の特徴空間で直接非線形の識別関数を学習する
    - アンサンブル学習、ニューラルネットワーク、k最近傍法
  - 非線形な特徴変換を行った後、線形識別関数を学習
    - カーネル法
    - Explicit Feature Maps

線型SVMなど、高バイアス・低バリエーションな手法のバイアスを減らす

# 簡単な例

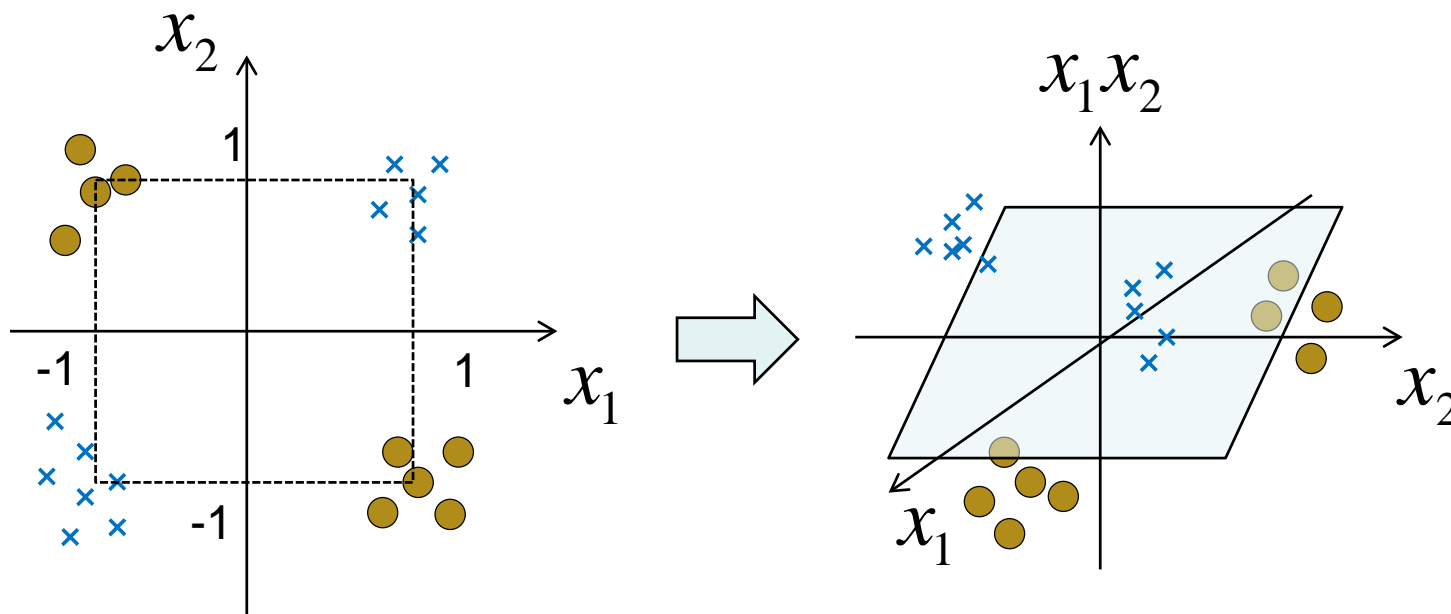
- XORの分離
  - 線型分離不可能



決定木なら…

# 特徴を高次元空間へ射影

- XORの例では、一つ特徴を増やすと分離できる



- 一般に、学習データ数の次元まで特徴次元を増やせば必ず線形分離できる
  - もちろんこれは汎化性を保証しない無意味な解（過学習）
  - 「意味のある」高次元空間への射影が重要

# カーネル法

- 線型モデル  $\hat{f}(\mathbf{x}) = \mathbf{a}^T \mathbf{x}$

- カーネルモデル  $\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i)$

$K(\mathbf{x}_i, \mathbf{x}_j)$  : カーネル関数。サンプル  $\mathbf{x}_i$  と  $\mathbf{x}_j$  の類似度を定義する

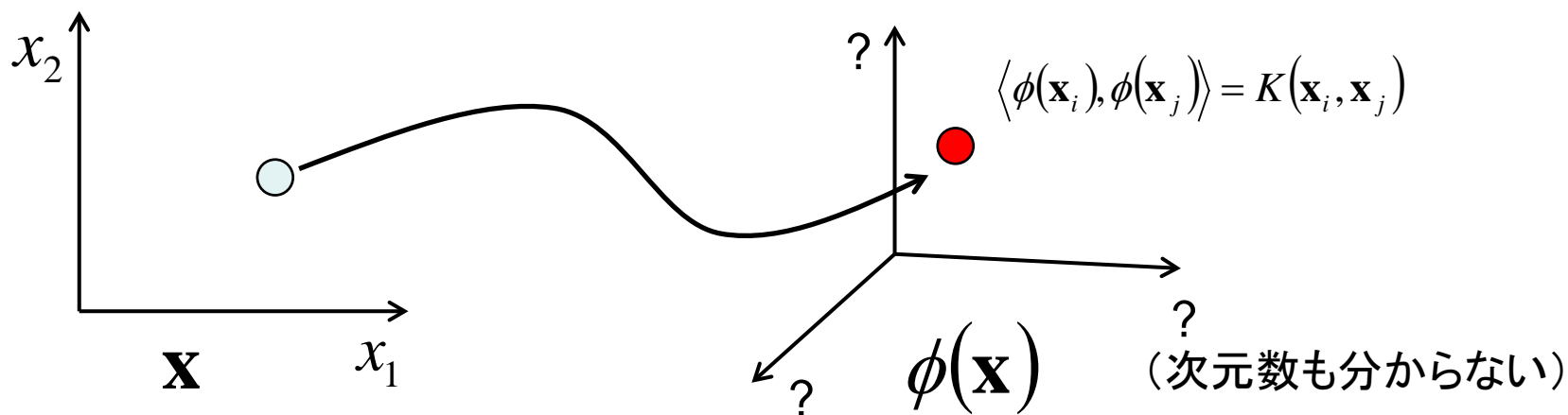
(例) ガウスカーネル 
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2c^2}\right)$$

- 直感的には、カーネル関数で定義される各学習サンプルとの類似度を新しい説明変数とした線形モデル



# カーネルトリック

- カーネルモデルは、特徴ベクトル  $\mathbf{x}$  を、内積がカーネル関数  $K$  で定義される高次元空間  $\phi(\mathbf{x})$  に陰に射影する



- 実際に  $\phi(\mathbf{x})$  がどのような形かは分からない
  - 多くの線形手法の学習・識別アルゴリズムはカーネルモデルを用いると内積計算だけで実現するように書き換えることができるので、分からなくてもよい =カーネルトリック

# そもそも論に戻る

- 空間って何だろう？
- 訓練データの特徴は、それ自体はあくまで離散・有限個の点の集合にすぎない
- 点の背後に、どのようなベクトル空間を仮定するか？

# 内積空間（計量ベクトル空間）

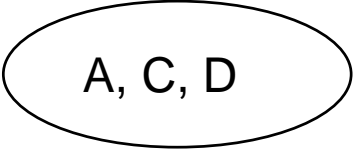
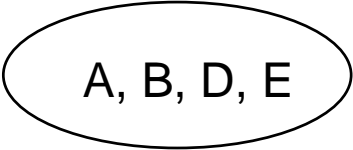
- 我々が（無意識に）考えているのは線形空間（ベクトル空間）
  - 和、スカラー倍が定義される（公理系に従う）
- さらに、**内積構造を持つ** → 内積空間
  - 演算 $(\mathbf{x}, \mathbf{y})$ が内積となる条件：
    - 1. 線形性
$$\begin{aligned}(\mathbf{x}, \mathbf{y} + \mathbf{z}) &= (\mathbf{x}, \mathbf{y}) + (\mathbf{x}, \mathbf{z}) \\ (\mathbf{x} + \mathbf{y}, \mathbf{z}) &= (\mathbf{x}, \mathbf{z}) + (\mathbf{y}, \mathbf{z}) \\ (c\mathbf{x}, \mathbf{y}) &= c(\mathbf{x}, \mathbf{y})\end{aligned}$$
    - 2. 共役線形性
$$(\mathbf{x}, c\mathbf{y}) = \bar{c}(\mathbf{x}, \mathbf{y})$$
    - 3. エルミート性
$$(\mathbf{x}, \mathbf{y}) = \overline{(\mathbf{y}, \mathbf{x})}$$
    - 4. 正定値性
$$\begin{aligned}(\mathbf{x}, \mathbf{x}) &\geq 0 \\ (\mathbf{x}, \mathbf{x}) &= 0 \quad \text{only if } \mathbf{x} = \mathbf{0}\end{aligned}$$
  - 例えばn次元ユークリッド空間では、 $(\mathbf{x}, \mathbf{y}) \equiv \sum_i^n x_i y_i$

# 内積空間（計量ベクトル空間）

- 内積は、二つのベクトルの類似度を示す
- 内積によってノルム・距離が定まる
  - ノルム：  $\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})}$
  - 距離：  $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{(\mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y})}$
- つまり、特徴点をユークリッド空間に配置する際は、  
内積がデータ間の類似度として妥当なものになっているかが重要

# 説明変数の設計、数量化

- 点をユークリッド空間に埋め込む作業
  - 内積が類似度として妥当になるように
- 例) ダミー変数

	A	B	C	D	E
 $\Rightarrow$	1	-1	1	1	-1
 $\Rightarrow$	1	1	-1	1	1

$\mathbf{x}_1 = (1, -1, 1, 1, -1)^T$

$\mathbf{x}_2 = (1, 1, -1, 1, 1)^T$

$\mathbf{x}_1^T \mathbf{x}_2 = 1 - 1 - 1 + 1 - 1 = -1$

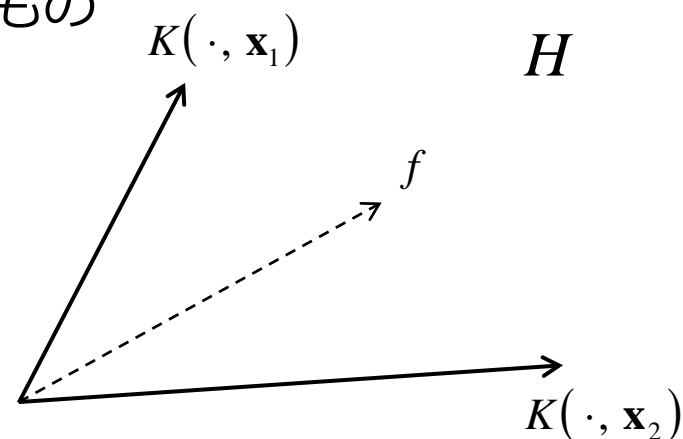
# 再生核ヒルベルト空間

- ヒルベルト空間

- ユークリッド空間を無限次元に拡張したもの

$$\phi: \mathbf{x}_i \rightarrow K(\cdot, \mathbf{x}_i) \text{ (無限次元の点)}$$

$$f = \sum_{i=1}^n \alpha_i K(\cdot, \mathbf{x}_i)$$



- **再生性** :  $f(\mathbf{x}) = \langle f, K(\cdot, \mathbf{x}) \rangle$

が成り立つ。したがって

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle K(\cdot, \mathbf{x}_j), K(\cdot, \mathbf{x}_i) \rangle = K(\mathbf{x}_i, \mathbf{x}_j)$$

カーネル関数が定義する類似度を内積に持つ  
新しい線形空間を作る！

# 線形SVM（前回のスライドより）

- 最小マージンを最大化する識別超平面を求める

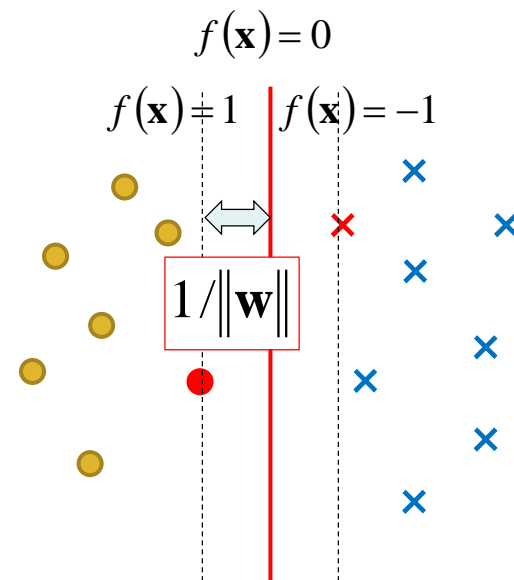
$$\max_{\mathbf{w}, b} \left( \min_i y_i f(\mathbf{x}_i) / \|\mathbf{w}\| \right) \quad f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- 線型分離可能（全学習データを正しく分離する平面が引ける）  
ことが前提

- 二次計画問題（等価な表現）

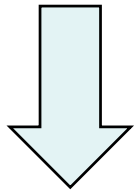
$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \longleftrightarrow \max 1 / \|\mathbf{w}\|$$

subject to  $y_i f(\mathbf{x}_i) \geq 1$  for  $\forall i$

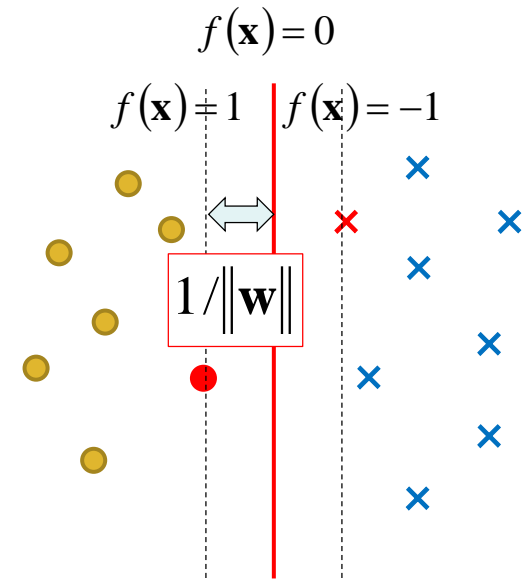


# 線形SVM：双対問題

$$\begin{aligned} \min & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{subject to} & y_i f(\mathbf{x}_i) \geq 1 \quad \text{for } \forall i \end{aligned}$$



双対形式に書き換えると…  
(導出は前回の資料参照)



$$\max \left[ \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \underline{\mathbf{x}_i^T \mathbf{x}_j} \right]$$

$$\text{subject to } \underline{\alpha_i \geq 0} \quad \text{for } \forall i$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

内積しか出てこない

$\alpha_i > 0$  に対応する  $\mathbf{x}_i$  が  
サポートベクター



# カーネルSVM

$$\max \left[ \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \underbrace{K(\mathbf{x}_i, \mathbf{x}_j)}_{\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)} \right]$$

subject to  $\alpha_i \geq 0$  for  $\forall i$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

高次元空間における内積を  
元の特徴空間で計算


# カーネル多変量解析

- カーネルPCA

高次元空間 $\phi(\mathbf{x})$ における分散最大化

$$\max_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N \left( \mathbf{w}^T \left\{ \phi(\mathbf{x}_i) - \frac{1}{N} \sum_{j=1}^N \phi(\mathbf{x}_j) \right\} \right)^2$$

$\text{Var}(\mathbf{w}^T \phi(\mathbf{x}))$




ここで、高次元空間における射影  $\mathbf{w}$  は

サンプル結合  $\mathbf{w} = \sum_{k=1}^N \alpha_k \left\{ \phi(\mathbf{x}_k) - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_l) \right\}$

を考えれば十分（ここがポイント！）

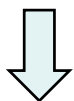
オフセット（サンプル平均）



# カーネルPCA (つづき)

代入して問題を書きかえると

$$\max_{\boldsymbol{\alpha}} \frac{1}{N} \sum_{i=1}^N \left( \sum_{k=1}^N \alpha_k \left\{ \phi(\mathbf{x}_k) - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_l) \right\} \cdot \left\{ \phi(\mathbf{x}_i) - \frac{1}{N} \sum_{j=1}^N \phi(\mathbf{x}_j) \right\} \right)^2$$

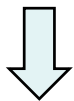


$$\max_{\boldsymbol{\alpha}} \frac{1}{N} \boldsymbol{\alpha}^T \tilde{K}^2 \boldsymbol{\alpha}$$

subject to

$$\boldsymbol{\alpha}^T \tilde{K} \boldsymbol{\alpha} = 1$$

(ノルム1の条件↑)



$$\tilde{K} \boldsymbol{\alpha} = \lambda \boldsymbol{\alpha} \quad (\boldsymbol{\alpha}^T \tilde{K} \boldsymbol{\alpha} = 1) \quad \text{サンプルサイズ次元の固有値問題}$$

$\tilde{K}$  : (中心化) **グラム行列**

$$\tilde{K}_{ij} = \underline{K(\mathbf{x}_i, \mathbf{x}_j)} - \frac{1}{N} \sum_{s=1}^N K(\mathbf{x}_i, \mathbf{x}_s) - \frac{1}{N} \sum_{t=1}^N K(\mathbf{x}_t, \mathbf{x}_j) + \frac{1}{N^2} \sum_{s,t=1}^N K(\mathbf{x}_s, \mathbf{x}_t)$$

要は、カーネルで定義されるデータ間類似度を要素に持つ行列

# カーネルPCA (つづき)

未知データ点  $\mathbf{x}^q$  のカーネル主成分空間への射影は

$$\begin{aligned} & \mathbf{w}^T \left\{ \phi(\mathbf{x}^q) - \frac{1}{N} \sum_{i=1}^N \phi(\mathbf{x}_i) \right\} \\ &= \left( \sum_{k=1}^N \alpha_k^{(p)} \left\{ \phi(\mathbf{x}_k) - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_l) \right\} \right) \left\{ \phi(\mathbf{x}^q) - \frac{1}{N} \sum_{i=1}^N \phi(\mathbf{x}_i) \right\} \\ &= \boldsymbol{\alpha}^{(p)T} \tilde{\mathbf{K}}^s \quad \swarrow \text{p番目の固有ベクトル} \\ & \quad \tilde{\mathbf{K}}_i^s = \underline{K(\mathbf{x}^s, \mathbf{x}_i)} - \frac{1}{N} \sum_{s=1}^N K(\mathbf{x}^q, \mathbf{x}_s) - \frac{1}{N} \sum_{t=1}^N K(\mathbf{x}_i, \mathbf{x}_t) + \frac{1}{N^2} \sum_{s,t=1}^N K(\mathbf{x}_s, \mathbf{x}_t) \end{aligned}$$

- 多変量解析手法は一般に同様の枠組でカーネル化できる
  - 回帰分析、判別分析、正準相関分析、etc.

# 代表的なカーネル

- 線形カーネル  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ 
  - 元の空間を使う場合（そのまま線形手法を適用することと等価）
  - 特徴次元数が大きく、データ数が少ない時有利
- ガウスカーネル  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$
- 多項式カーネル  $K(\mathbf{x}_i, \mathbf{x}_j) = (c + \mathbf{x}_i^T \mathbf{x}_j)^D$
- シグモイドカーネル  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\theta_1 \mathbf{x}_i^T \mathbf{x}_j + \theta_2)$

# カーネルの設計

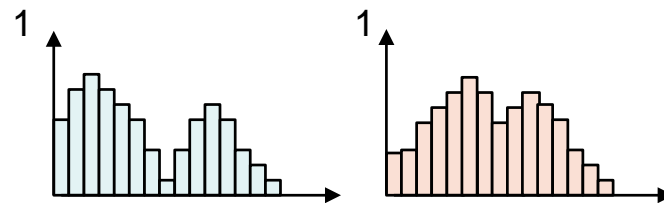
- データ間類似度だけ定義、あとはおまかせ
  - 数値的に特徴化できない（ベクトルでない）表現や構造でも、類似度さえ定義できればよい
  - 対象に関する事前知識は必要
- なんでもありなわけではない
  - Mercer's condition
  - 1. 対称性  $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$
  - 2. 正定値性  
任意の実数  $\alpha_i, \alpha_j$  について  $\sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

# 例えば

- カイ二乗カーネル

- コンピュータビジョン分野でよく使われる(ヒストグラム特徴)
- 統計のカイ二乗検定を参考に作られた
- 離散分布間の類似度を測る (ある程度) 妥当性のあるカーネル

$$K_{\chi^2}(\mathbf{x}_i, \mathbf{x}_j) = 1 - \sum_{m=1}^d \frac{2(x_i^m - x_j^m)^2}{x_i^m + x_j^m}$$

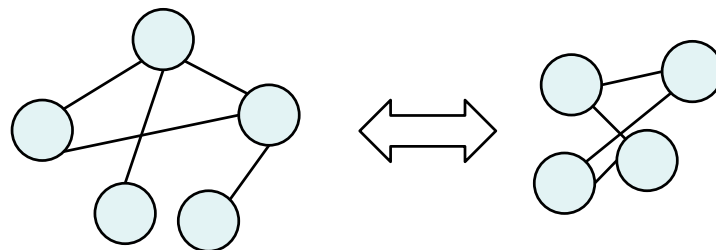


カイ二乗検定  $\chi^2(P, Q) = \sum_{m=1}^d \frac{(P_m - Q_m)^2}{Q_m}$

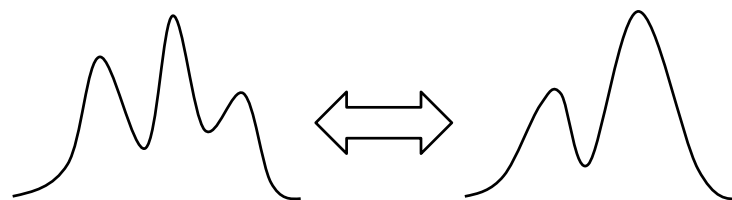
# 構造を扱うカーネル

- String kernel
  - 二つの文字列の一致度
  - 方法はいろいろ (DPマッチングなど)

- Graph kernel
  - 二つのネットワークの類似度
  - e.g. 分子の構造



- Probabilistic kernel
  - 二つの確率分布の類似度
  - e.g. 音声認識、画像認識





# カーネル法の注意点(1)

- カーネルで仮定するデータ間類似度が適切でないとあまり効果は得られない
  - 魔法の道具ではない！
  - 対象に関する知識がないと適切なカーネルの選定は難しい
- よくあるパターン
  - 「とりあえず、教科書通りガウスカーネル使ってみたけど線形カーネルとほとんど変わらない…」

## カーネル法の注意点(2)

- サンプルサイズに対するスケーラビリティに乏しい
  - グラム行列を保持しておく必要がある
  - 大規模な問題ではそのまま使えない

	カーネルモデル	線形モデル
学習	$O(n^2) \sim O(n^3)$	$O(n)$
テスト	$O(n)$	$O(1)$

- 対処法
  - 1. グラム行列の低ランク近似
    - 不完全Cholesky分解、Nyström近似（ランダムサンプリング）
  - 2. Explicit feature map

# Explicit feature maps

- 高次元空間  $\phi(\mathbf{x})$  を陽に導出するアプローチ
  - 多くの場合は近似的に
- もちろんいつでも都合よく求まるわけではない
  - 実用上よく使われるものを中心に研究されている
  - ヒストグラム向けのカーネルなど
- データの大規模化に伴い、計算コストのボトルネックが逆転
  - 特徴次元数を大幅に増やしてでも、データ数に対して線形の計算コストに収めたい


# Additive kernel

- 特徴要素ごとの1次元のカーネルの総和で表されるカーネル

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^d k(x_i^m, x_j^m)$$

- ヒストグラム特徴など（ビンごとのカーネル）
- 1次元のカーネルについてfeature mapができれば、あとは並べるだけ
- 例) Bhattacharyya kernel
  - もとの特徴を先にsqrtしておけばOK  
(これも立派なfeature map)

$$k(x_i, x_j) = \sqrt{x_i x_j}$$


$$BC(p, q) = \int \sqrt{p(\mathbf{x})q(\mathbf{x})} d\mathbf{x}$$

Bhattacharyya距離

# Additive kernel: Histogram intersection

- ビンごとに小さい方の値をとるカーネル

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^d k(x_i^m, x_j^m) \quad k(x_i, x_j) = \min(x_i, x_j)$$

- $x$  を量子化し, 1で埋める

- 例)  $0 \leq x \leq 1, x_1 = 0.45, x_2 = 0.8$

$$\phi(x_1) = \frac{1}{\sqrt{5}}(1, 1, 0, 0, 0)^T, \phi(x_2) = \frac{1}{\sqrt{5}}(1, 1, 1, 1, 0)^T \implies \phi(x_1)^T \phi(x_2) = 0.4$$

- 近似の精度は量子化数に依存

- 非常に高次元になるが、それでも元のカーネル法よりずっと高速

S. Maji and A. C. Berg, "Max-Margin Additive Classifiers for Detection",  
In Proc. ICCV, 2009.

# Polynomial embedding

- 昔の課題で出たアイデア

$$\mathbf{x} = (x_1, x_2)^T \quad \Rightarrow \quad \phi(\mathbf{x}) = (x_1^2, x_1x_2, x_2^2)^T$$

$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = x_{i1}^2 x_{j1}^2 + x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2 + \dots$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = (c + \mathbf{x}_i^T \mathbf{x}_j)^D = (c + x_{i1}x_{j1} + x_{i2}x_{j2})^D$$

$$c = 0, D = 2 \quad \rightarrow \quad x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2$$

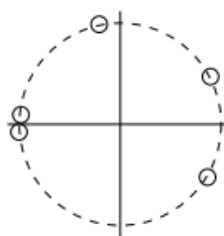
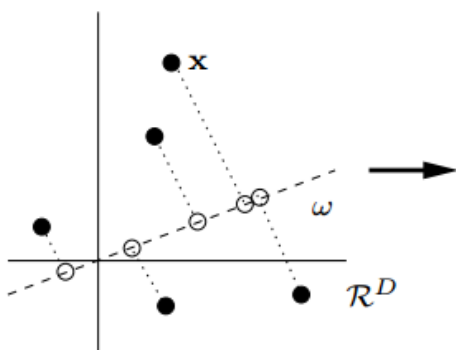
(参考)

N. Pham and R. Pagh, “Fast and Scalable Polynomial Kernels via Explicit Feature Maps”, In Proc. KDD, 2013.

# より一般的な場合

- RBF kernel (ガウスカーネル)

- A. Rahimi and B. Recht, "Random features for large-scale kernel machines", In Proc. NIPS 2007.
- Random Fourier features による近似



Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\ \Delta\ _2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\ \omega\ _2^2}{2}}$
Laplacian	$e^{-\ \Delta\ _1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\ \Delta\ _1}$

- Generalized RBF kernel

- S. Vempati, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Generalized RBF feature maps for Efficient Detection", In Proc. BMVC 2010.

RBF kernel + additive kernel (例えばこんなの)

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2} \chi^2(\mathbf{x}, \mathbf{y})\right), \quad \chi^2(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^d \frac{2x_m y_m}{x_m + y_m}$$

# まとめ

- ランダムフォレスト
- 非線形識別
  - 非線形識別関数
  - 非線形特徴抽出 + 線形識別関数
- カーネル法
  - データ間類似度の定義が重要
  - 明示的に高次元空間を（可能なら）書き下すのも一つの手
- 次回
  - ニューラルネットワーク、deep learning