THE UNIVERSITY OF TOKYO

数理情報工学特論第一
【機械学習とデータマイニング】
1章：概論（３）

かしま　ひさし
鹿島 久嗣
（数理６研）

kashima@mist.i.～

東京大学
THE UNIVERSITY OF TOKYO

DEPARTMENT OF MATHEMATICAL INFORMATICS

発展的な話題

少し発展的な話題を2つ：

- 多クラス分類
  - 2クラス分類の一般化
- カーネル法
  - 近年注目されている学習手法

発展的な話題１：
多クラスの分類（教師付き学習）

- これまでは2クラスの分類問題を考えていた
  - ロジスティック回帰の出力は $y \in \{+1, -1\}$ の2値
- Kクラスへの対応をするにはどうしたらよいだろうか
  - たとえば {A, B, C, D, E} の5クラス
  - 例： 文字認識、文書分類
- 主に2つのアプローチがとられる：
  - 2クラスの分類問題に帰着する
  - 多クラス版の分類モデルを直接考える

多クラスの分類に対する最もシンプルなアプローチは「2クラス分類への帰着」です

- すでに2クラスの分類問題については解法が分かっているから、2クラス分類に帰着できれば色々と都合がよいはず
- アプローチ1: 1対多(one-versus-rest)方式
  - あるクラスか、そうでないか、という分類問題をK個考える
  - K個のモデルができる
  - 予測時にはもっとも確率の高いクラスに予測する
- アプローチ2：1対1 (one-versus-one) 方式
  - 全ての2つのクラスの組について、2クラスの分類問題を考える（$K(K\text{-}1)$個の2値分類問題）
  - $K(K\text{-}1)$個のモデルができる
  - 予測時には、それぞれのクラスへの所属確率の和で多数決をとる

1対多方式と1対1方式の中間的な2クラス分類帰着方法として「誤り訂正出力符号方式」があります

- 1対多方式では表現力不足、1対1方式ではモデルが多く（クラス数の2乗）なりすぎる場合がある
- アプローチ1.5：誤り訂正出力符号 (error correcting output code; ECOC) 方式
  - クラスを適当に2グループに分けた、2クラス分類問題を複数作る
  - 予測時には、予測時には、それぞれのクラスへの所属確率の和で多数決をとる

クラス	2値分類問題					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1

「誤り訂正出力符号方式」のポイントは、うまくクラスが分離するようにグループ分けをするところです

- 分類がうまくいくためにはグループ分けの表の行（「符号」とよぶ）がうまく分離している（ハミング距離が離れている）とよい

クラス	2値分類問題					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1

クラス間のハミング距離

	A	B	C	D
A	0	4	4	3
B		0	4	3
C			0	3
D				0

- 「符号」をうまく設計するところがポイント

多クラス分類を直接行うモデル：多クラスロジスティック回帰

- ロジスティック回帰を多クラス版に拡張する
  - 2クラスのロジスティック回帰

$$P(y = +1|\mathbf{x}; \mathbf{w}) := \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

    - パラメータ$\mathbf{w}$は、$\mathbf{x}$ の各次元のクラス+1への寄与分を表す
  - 多クラスのロジスティック回帰

$$P(y = c|\mathbf{x}; \{\mathbf{w}_c\}_{c \in \mathcal{Y}}) := \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_c \exp(\mathbf{w}_c^\top \mathbf{x})}$$

実数値を正の値にマップ

足して1（確率）になるよう正規化

    - クラス集合 $\mathcal{Y}$ の各クラスcに対してパラメータベクトル $\mathbf{w}_c$ をもつ
    - パラメータ$\mathbf{w}_c$は、$\mathbf{x}$ の各次元のクラスc への寄与分を表す

発展的な話題 2: カーネル法

- ロジスティック回帰のカーネル法化
- 表現定理：カーネル法の正当化
- カーネル関数の例：グラフ・カーネル

カーネル法は、非線形モデルを線形モデルのように扱える近年話題の手法です

- ここ10年くらい機械学習の世界で研究が進んでいるモデル
- 理由は：
  - サポートベクトルマシン（SVM）というモデルが各地で大成功を収めた
  - データの見方を「特徴空間ビュー」から「類似度ビュー」に変換することで…
    - 高次元のデータに対しても適用できる（次元数→データ数）
    - 非線形なモデルの学習が行える
    - 木やグラフなどの非ベクトル的な対象を扱うことが出来る
- 多くのモデルが、カーネル法に変換することが出来る

ロジスティック回帰のカーネル化

- ロジスティック回帰モデル

$$P(y = +1|\mathbf{x}) := \sigma(\mathbf{w}^\top \mathbf{x})$$

- 仮定：パラメータが、入力ベクトルの線形結合で表せるとする

$$\mathbf{w} := \sum_{i=1}^{N} \alpha^{(i)} \mathbf{x}^{(i)}$$

  - $\alpha := (\alpha^{(1)}, \alpha^{(2)}, ..., \alpha^{(N)})$ が新たなパラメータ

これを「カーネル化」という

- ロジスティック回帰モデルを書き直すと

$$P(y = +1|\mathbf{x}) := \sigma\left(\sum_{i=1}^{N} \alpha^{(i)} \langle \mathbf{x}^{(i)}, \mathbf{x} \rangle\right) := \sigma\left(\sum_{i=1}^{N} \alpha^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})\right)$$

  - $< \cdot , \cdot >$ は内積
  - $K(\cdot , \cdot) := < \cdot , \cdot >$をカーネル関数と呼ぶ (内積を置き換えただけ)

カーネル化によって何が起こったか？ 高速化と非線形化

- カーネルロジスティック回帰モデル

$$P(y=+1|\mathbf{x}) := \sigma\left(\sum_{i=1}^{N} \alpha^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})\right)$$

ただし、カーネル関数

$$K(\mathbf{x}^{(i)}, \mathbf{x}) := \langle \mathbf{x}^{(i)}, \mathbf{x} \rangle$$

内積

- カーネル化によって
  - モデルのパラメータがD個（次元数）からN 個（データ数）になった
  - データアクセスがカーネル関数（内積）を通じてのみ行われるようになった
- つまり
  - $N < D$のときに速い。 特に、カーネル関数の計算が $\mathbf{x}$ の次元よりも小さいオーダーであるときに速い
  - $\mathbf{x}$ が何だかよく分からない対象であっても、類似度らしきものがカーネル関数として与えられてさえいれば一応動く

表現定理：なぜパラメータを線形結合で表してよいのか？に答えます

- カーネル化は、パラメータが入力ベクトルの線形結合で表されるという仮定

$$\mathbf{w} := \sum_{i=1}^{N} \alpha^{(i)} \mathbf{x}^{(i)}$$

に基づくが、果たしてこれは正しいのか？

- 答え：L2正則化（リッジ正則化）ならば正しい
  - リプレゼンタ定理（表現定理）によって保証される
  - ちなみに、正則化の項は

$$\| \mathbf{w} \|_2^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^{(i)} \alpha^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

- L1正則化（ラッソ正則化）ではこれは保証されない
  - ひとつのアドホックな解決法としては $\| \boldsymbol{\alpha} \|_1$ を使う

表現定理の証明

$$\mathbf{w} := \sum_{i=1}^{N} \alpha^{(i)} \mathbf{x}^{(i)}$$

- 目的関数 $L := \sum \log P(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) - \lambda \parallel \mathbf{w} \parallel_2^2$ を最大化するパラメータを $\mathbf{w}^*$ と置く
- $\mathbf{w}^*$を線形結合で表現できる部分 $\mathbf{w}$ とそれ以外の部分（どの$\mathbf{x}^{(i)}$とも直交する）$\mathbf{w}'$ に分ける $\mathbf{w}^* = \mathbf{w} + \mathbf{w}'$

$$L := \sum_{i=1}^{N} \log P(y^{(i)}|\mathbf{w}^{*\top}\mathbf{x}^{(i)}) - \lambda \parallel \mathbf{w}^* \parallel_2^2$$

$$= \sum_{i=1}^{N} \log P(y^{(i)}|\mathbf{w}^{\top}\mathbf{x}^{(i)}) - \lambda \left(\parallel \mathbf{w} \parallel_2^2 + \parallel \mathbf{w}' \parallel_2^2 \right)$$

1）パラメータと入力ベクトルの積に依存することを明示

2）$\mathbf{w}'$は入力ベクトルとはすべて直交するので、$\mathbf{w}'$ に依存する部分が消える

3）これは対数尤度とは関係ないから、勝手に最小化（$= \mathbf{0}$）してよい。
よって、 $\mathbf{w}^* = \mathbf{w}$

カーネル関数：データ間の類似度を定義する関数
これがあればカーネル法は動く

- カーネル関数とは、2つの特徴ベクトルの内積

$$K(\mathbf{x}, \mathbf{x}') := \langle \mathbf{x}, \mathbf{x}' \rangle$$

  - 内積 = ある種の類似度 と考えることができる
- カーネル法は、内積（カーネル関数）として解釈できる類似度関数が与えられていれば動く
- 適切なカーネル関数さえ定義できれば、
  - （暗に）高次元の特徴ベクトル
  - 文字列、木、グラフなどの非ベクトル型データ

  に対しても適用可能

高次元空間におけるカーネル関数の例：多項式カーネル

- カーネル関数を2つの特徴ベクトル $\mathbf{x} = (x_1, x_2)^\top$ と $\mathbf{x'} = (x_1', x_2')^\top$ の内積とする

$$K(\mathbf{x}, \mathbf{x}') := \langle \mathbf{x}, \mathbf{x}' \rangle = x_1 x_1' + x_2 x_2'$$

- このカーネル関数を2乗したカーネル関数を考えてみると

$$\begin{aligned} K^2(\mathbf{x}, \mathbf{x}') &= \langle \mathbf{x}, \mathbf{x}' \rangle^2 \\ &= x_1^2 {x_1'}^2 + x_2^2 {x_2'}^2 + 2 x_1 x_2 x_1' x_2' \\ &= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2)^\top, ({x_1'}^2, {x_2'}^2, \sqrt{2} x_1' x_2')^\top \rangle \end{aligned}$$

- 特徴の組み合わせの項 $\sqrt{2}\, x_1\, x_2$ が登場し、特徴ベクトルの次元が3になったにもかかわらず、計算量は元々の次元数(2)に依存
- 一般に、d 乗することでd 個の特徴の組み合わせを実現できる

非ベクトル型データに対するカーネル関数

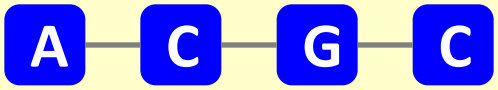
- 文字列、木、グラフなど、予め特徴ベクトルで表されていないようなデータを扱う方法は自明でない
  - いままでの議論は「特徴ベクトル」ありきであった
- どのようにしたらよいか？

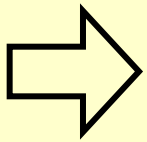
  → カーネル法なら、とりあえずカーネル関数（＝２つの非ベクトル型データの間の類似度）さえ定義できれば動くはず

1つの考えうるアプローチは「部分構造」を用いて特徴空間を構成することですが、計算量的な問題を抱えています

- 「部分構造」が構造の性質を担っていると考える
  - 配列データの性質は、部分配列が担う（＝マルコフモデル）
- 各部分構造の有無や出現回数を用いて、特徴ベクトル表現する
- しかし、すべての部分構造をテーブル要素の候補にするときりがない
  - グラフには、指数個の部分グラフが含まれるので、すべてを数え上げている場所と時間はない（これが本質的な問題）

x



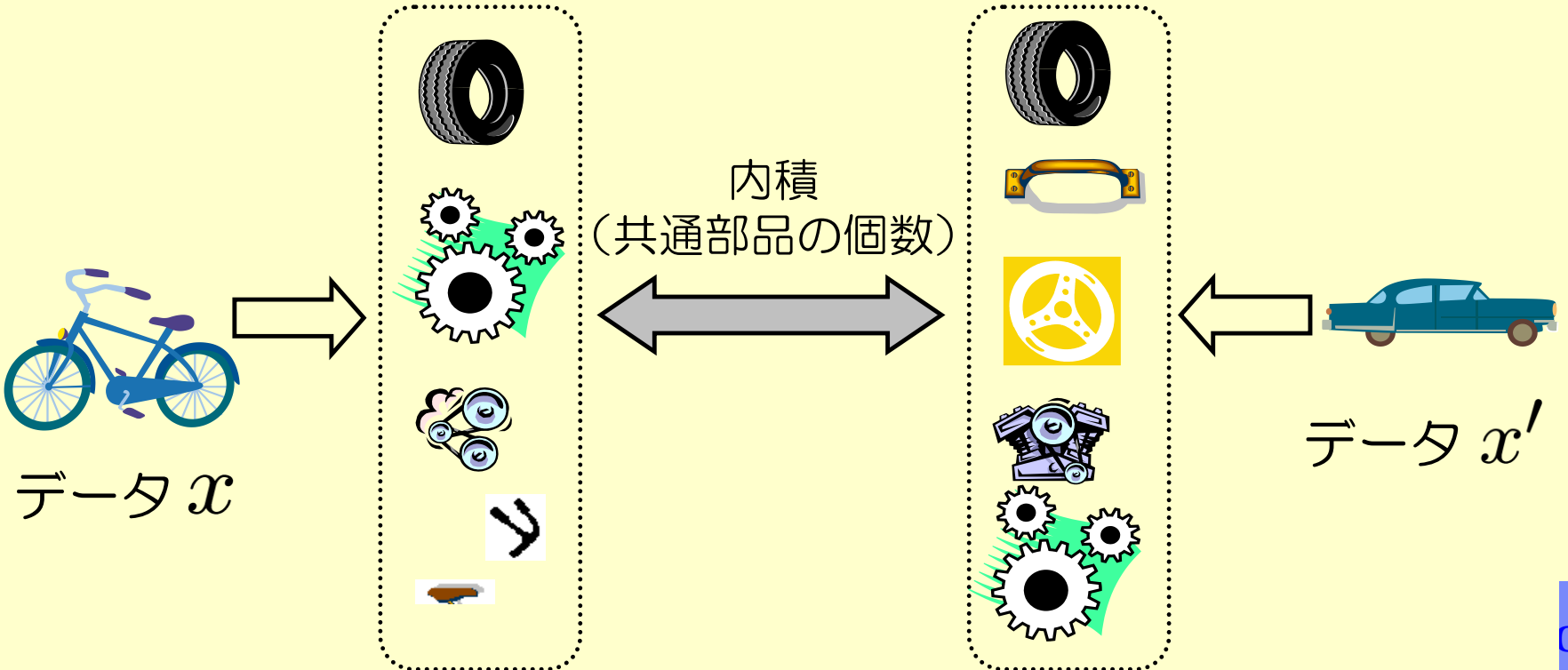


$\Phi(x)$

A C	T C	G C	A A	…
1	0	1	0	…

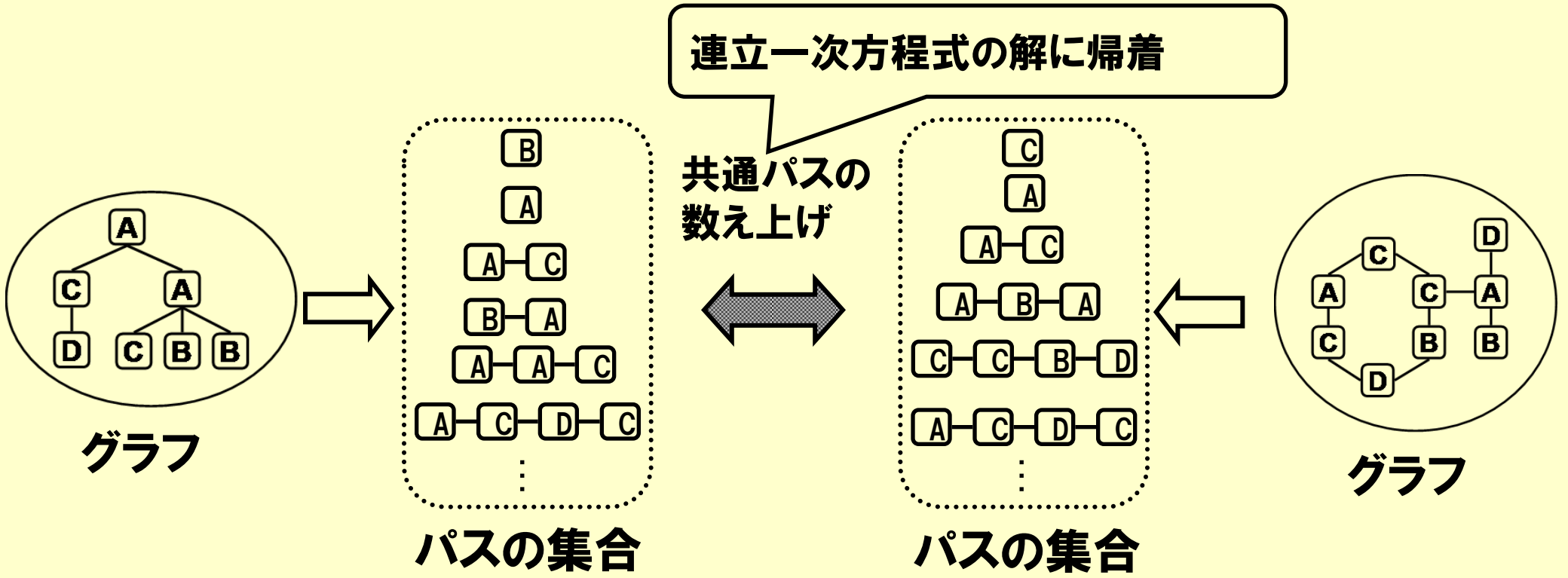
構造データのカーネル関数の設計指針：共通部分構造を数え上げる

- 2つの構造データのカーネル関数（＝部分構造によって定義された2つの特徴ベクトルの内積）を、特徴ベクトルを明示的に作らずに計算する
  - 対象とする構造データに対して、適切な部分構造を定義する
    - 木なら部分木、グラフならパス
  - カーネル関数の計算は、共通部分構造の個数を数える問題になる
    - 再帰構造を利用して、内積だけを効率的に計算



グラフに対するカーネル関数の設計例：パスを部品として用います

- グラフカーネルでは、部分構造を、グラフ上のランダムウォークによって取り出されるパスとして定義する
- 難しいところ： パスは無限個ありうる
  ← 解決法： 数え上げの計算を連立一次方程式に帰着することで計算可能になる

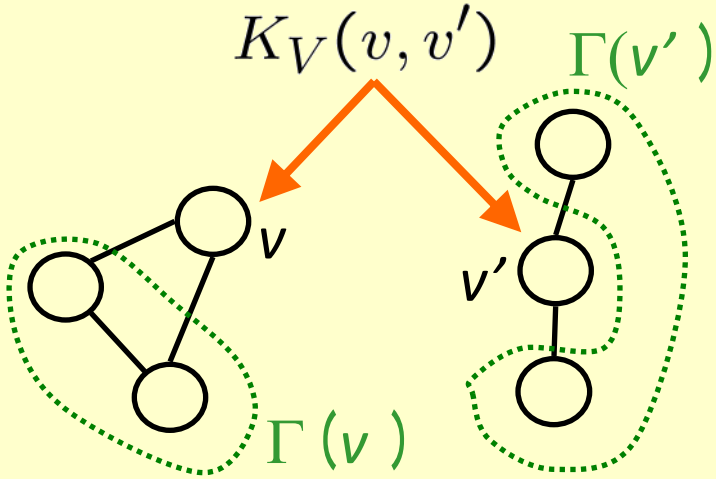


グラフ・カーネルの計算は、連立方程式に帰着され、効率的に行えます

- ランダム・ウォークによる無限個のパスによって特徴ベクトルを構成
  - 特徴ベクトル$\Phi(x)$ = (A–C の出現回数×λ^2, A–A–C の出現回数×λ^3, ...)
  - 発散しないように、パスの長さによって指数的に減衰される
- 再帰的表現に持ち込むことで効率的に計算可能になる
  - $S_v(x)$: ノード v で終わるパスの集合
  - ノード対 (v, v') で終わる同じラベル列をもつパスの個数 K_V の和に分解できる

$$K(x,x') = \sum_{v\in V}\sum_{v'\in V'} K_V(v,v')$$

$$K_V(v,v') = \sum_{s\in S_v(x)}\sum_{s'\in S_{v'}(x')} \lambda^{|s|}\lambda^{|s'|}\delta(s,s')$$



  - K_V(v, v')は近隣ノード同士の K_V の和によって再帰的に書けるので、連立方程式を解けばカーネル関数の値が計算できる

$$K_V(v,v') = \lambda^2\delta(\ell(v),\ell(v'))\Big(1+\sum_{\tilde{v}\in\Gamma(v)}\sum_{\tilde{v}'\in\Gamma(v')}\lambda^2 K_V(\tilde{v},\tilde{v}')\Big)$$

ここまでのまとめ：

- 多クラス分類：
  - 2クラスの分類を組み合わせれば、多クラスの分類問題を解ける
    - 1対多方式、1対1方式、誤り訂正出力符号方式
  - 多クラス用の分類モデルを直接設計することもできる
    - 多クラスロジスティック回帰
- カーネル法：
  - データの見方を「特徴空間ビュー」から「類似度ビュー」に変換することで、高次元のデータを扱えるようにする方法
    - 高次元のデータに対しても適用できる（次元数→データ数）
    - カーネルの定義によっては非線形なモデルの学習が行える
  - カーネル化を正当化するための表現定理
    - パラメータについて線形なモデルに、L2（リッジ）正則化を適用する場合、成立する

次回以降、各論に入っていきます

- 教師つき学習
  - 回帰
  - 分類
  - カーネル法
- 教師なし学習
  - 潜在変数モデル
  - グラフィカルモデル
  - 異常検知

この講義のWebページ

- 講義資料等は

  たいにー ゆーあーるえる どっとこむ ／ ２７ エイチ ダブリュ ２７ ティー

  http://tinyurl.com/27hw27t

  に置いていきます