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（２）

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

機械学習問題の定式化

学習の定式化：
機械学習の問題をどのように数理的に定式化するか？

- 機械学習の問題を数理的に扱うために、まず、学習の対象と、学習するべきモデルを表現した
  - 対象：ものごとを実数値ベクトルで表現した
  - モデル：その上での確率モデルを考えた
- また、 教師付き／教師無しという学習の2つの目的を定義した
- つぎに、その目的を、最適化問題として定式化する
  - 最尤推定による最適化問題としての定式化

最尤推定：モデル推定問題のもっとも標準的な定式化です
データを最もよく再現するパラメータを求める最適化問題として定式化します

- 最尤推定の基本的な考え方：訓練データを最もよく再現するパラメータが良いパラメータとする
  - 訓練データを最もよく再現する ＝ 最も高い確率を与える
- 訓練データが互いに独立であるとすると、その同時確率は
  教師無し学習なら $\prod_{i=1}^{N} P(\mathbf{x}^{(i)})$ 、教師付き学習なら $\prod_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)})$
  で与えられる　「尤度」とよぶ
- これ（の対数）を最大にするパラメータを求める

  教師無し学習の場合

  $$L := \sum_{i=1}^{N} \log P(\mathbf{x}^{(i)})$$

  教師付き学習の場合

  $$L := \sum_{i=1}^{N} \log P(y^{(i)}|\mathbf{x}^{(i)})$$

  「対数尤度」とよぶ
- モデル推定の問題が、対数尤度を目的関数とした最適化問題として捉えられる

最尤推定：モデル推定問題のもっとも標準的な定式化です 教師つき/教師ナシそれぞれで実際例を見てみます

- 目的：訓練データから、モデルのパラメータを推定する
- 混合正規分布（教師無し学習）の場合
  - 訓練データ　$(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots, \mathbf{x}^{(N)})$
  - パラメータ

$$P(\mathbf{x}) := \sum_{k=1}^{K} \boxed{w^{(k)}} g^{(k)}(\mathbf{x})$$

$$g^{(k)}(\mathbf{x}) := \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boxed{\boldsymbol{\Sigma}^{(k)}}|^{1/2}} \exp(-(\mathbf{x} - \boxed{\boldsymbol{\mu}^{(k)}}) \boxed{\boldsymbol{\Sigma}^{(k)}}^{-1} (\mathbf{x} - \boxed{\boldsymbol{\mu}^{(k)}}))$$

- ロジスティック回帰（教師付き学習）の場合
  - 訓練データ

$$((\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}) \ldots, (\mathbf{x}^{(N)}, y^{(N)}))$$

  - パラメータ　$P(y = +1|\mathbf{x}) := \sigma(\boxed{\mathbf{w}^{\top}} \mathbf{x})$

最尤推定の例：多次元正規分布（教師なし学習）
最適なパラメータは、閉じた形で求まります

- 多次元正規分布の最尤推定（平均のみ。共分散行列は定数とする）

$$P(\mathbf{x};\boldsymbol{\mu}) := \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp(-(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

- 対数尤度は

$$\begin{aligned} L &:= \sum_{i=1}^{N}\log P(\mathbf{x}^{(i)};\boldsymbol{\mu}) \\ &= \sum_{i=1}^{N}(-(\mathbf{x}^{(i)}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(i)}-\boldsymbol{\mu}))+\text{const.} \end{aligned}$$

- **μ**で微分

$$\frac{\partial L}{\partial \boldsymbol{\mu}} = \sum_{i=1}^{N} 2\boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(i)}-\boldsymbol{\mu}) = 2\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\boldsymbol{\mu})$$

- := **0**とおいて解くと、

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N}\mathbf{x}^{(i)}}{N}$$

データの平均に
なった

ここまでのまとめ：機械学習の問題は最尤推定によって最適化問題として定式化されます

- 最尤推定：
  「訓練データを、最もよく再現するパラメータが良いパラメータとする」に基づいて学習を行う
- 対数尤度を目的関数として、パラメータについての最大化を行う
- 多次元正規分布の平均パラメータの最尤推定による推定値は、データの平均によってもとまる
  - ちなみに、共分散行列の推定値は、データの共分散行列によって求まる
- もっと複雑なモデル（混合正規分布、ロジスティック回帰）では、最尤推定はどのように行えばいいだろうか？

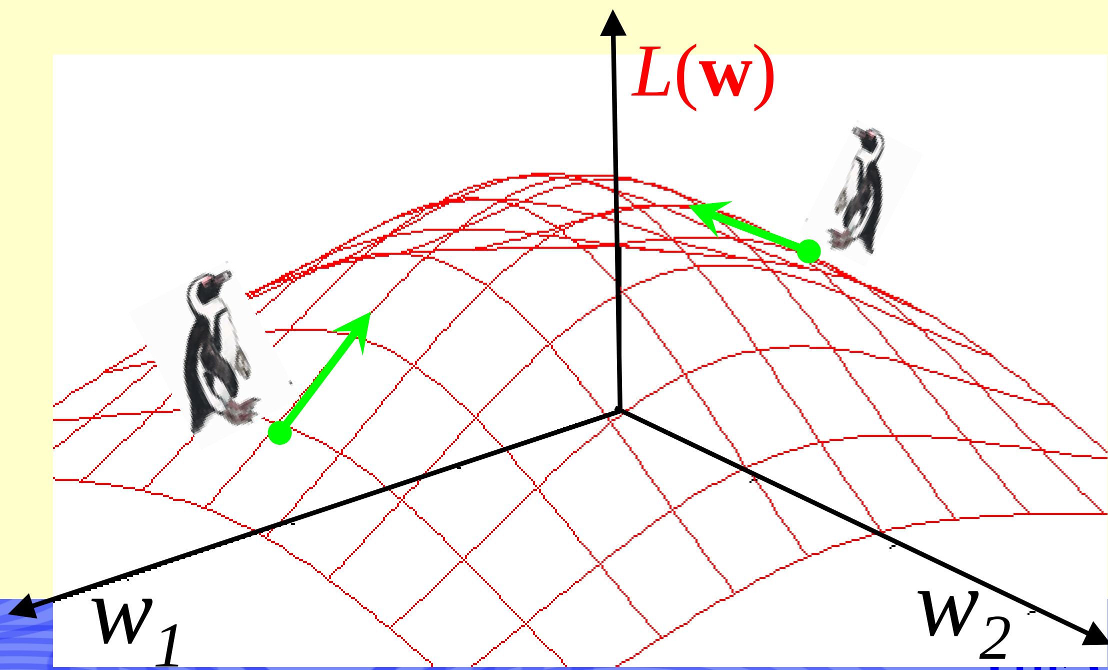
機械学習のアルゴリズム

学習のアルゴリズム：
対数尤度を最大化するパラメータを数値的に求めます

- 必ずしも正規分布のように閉じた形で解が求まるわけではない
- 最尤推定を数値的に行うためのアルゴリズム
  - 勾配法
  - EMアルゴリズム
- さらに、大規模なデータを用いた学習を、効率的に行うための方法
  - オンライン学習アルゴリズム

勾配法：もっとも基本的な最適化法
目的関数が最も急な方向にパラメータ更新を繰り返します

- 山（対数尤度*L*）の頂点を（なるべく速く）目指したい
- もっとも坂が急な方向に向かって（パラメータ上で）1m進む
  - 頂上付近だと、頂上を越えて向こう側にいってしまうことも
    - 実際には歩幅はだんだん小さくする



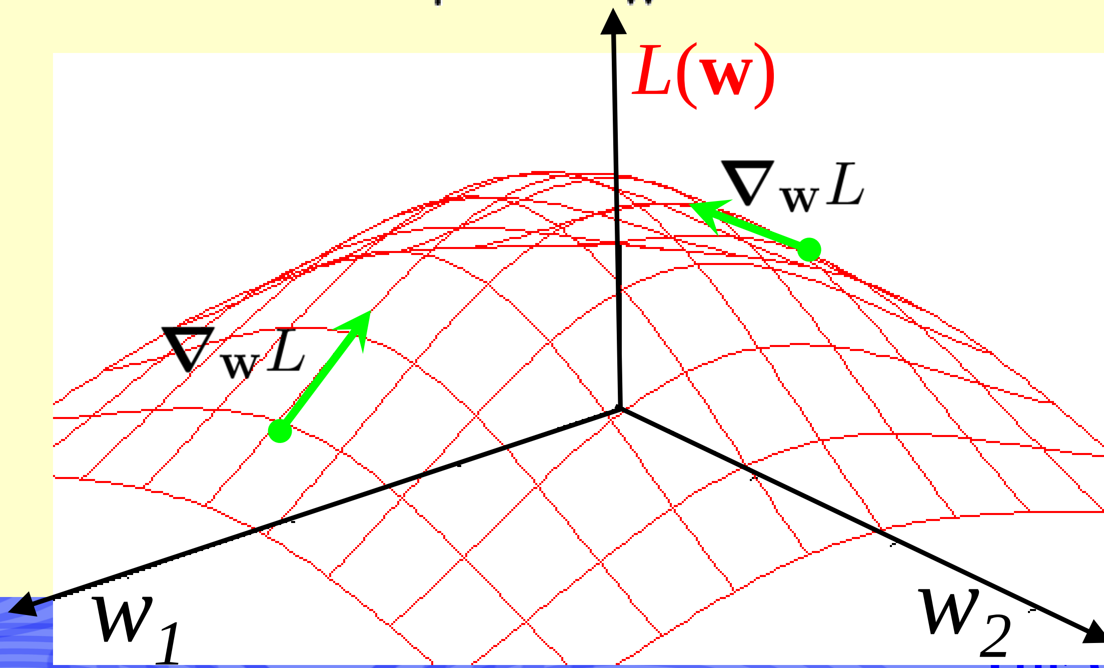
勾配法：もっとも基本的な最適化法
最急方向（勾配）は対数尤度の偏微分で求まります

- もっとも急な方向 ＝ 勾配
- 勾配は、目的関数（対数尤度）$L(\mathbf{w})$ のパラメータ$\mathbf{w}$での偏微分

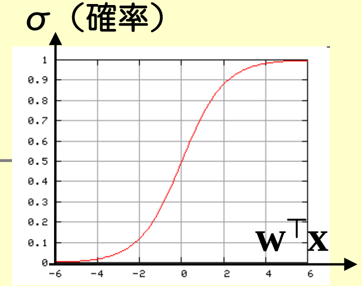
$$\nabla_{\mathbf{w}} L := \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \cdots, \frac{\partial L}{\partial w_D} \right)$$

- 勾配の方向に、少し（正の定数α）更新する

$$\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w}^{\mathrm{OLD}} + \alpha \nabla_{\mathbf{w}} L$$



ロジスティック回帰に対する勾配法：勾配を実際に計算してみます



- 条件付分布の対数尤度の勾配を求める

- カテゴリ+1 の訓練データのインデクス集合をPos,
カテゴリ-1 のインデクス集合をNegとすると、対数尤度は、

$$L(\mathbf{w}) := \sum_{i=1}^{N} \log P(y^{(i)}|\mathbf{x}^{(i)};\mathbf{w})$$

$$= \sum_{i\in\mathrm{Pos}} \log\sigma(\mathbf{w}^\top\mathbf{x}^{(i)}) + \sum_{i\in\mathrm{Neg}} \log(1-\sigma(\mathbf{w}^\top\mathbf{x}^{(i)}))$$

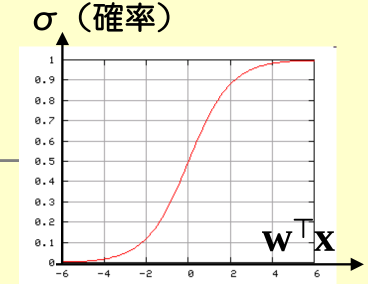
+1カテゴリのデータ

-1カテゴリのデータ

- 勾配は、

$$\frac{\partial L(\mathbf{w})}{\partial\mathbf{w}} = \sum_{i\in\mathrm{Pos}} \frac{\partial\log\sigma(\mathbf{w}^\top\mathbf{x}^{(i)})}{\partial\mathbf{w}} + \sum_{i\in\mathrm{Neg}} \frac{\partial\log(1-\sigma(\mathbf{w}^\top\mathbf{x}^{(i)}))}{\partial\mathbf{w}}$$

ロジスティック回帰に対する勾配法：比較的シンプルな形で求まります



- 勾配は、

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i \in \mathrm{Pos}} \frac{\partial \log \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})}{\partial \mathbf{w}} + \sum_{i \in \mathrm{Neg}} \frac{\partial \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}))}{\partial \mathbf{w}}$$

- ここで、$\sigma(a) := \frac{1}{1+e^{-a}}$ 、$\left(1 - \sigma(a) := \frac{e^{-a}}{1+e^{-a}}\right)$ より

$$\log \sigma(a) = -\log(1+e^{-a}) \Rightarrow \frac{\partial \log \sigma(a)}{\partial a} = \frac{e^{-a}}{1+e^{-a}} = 1 - \sigma(a)$$

$$\log(1-\sigma(a)) = -a - \log(1+e^{-a}) \Rightarrow \frac{\partial \log(1-\sigma(a))}{\partial a} = -\sigma(a)$$

- 結局、

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i \in \mathrm{Pos}} \left(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})\right) \mathbf{x}^{(i)} - \sum_{i \in \mathrm{Neg}} \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \mathbf{x}^{(i)}$$

EMアルゴリズム：「隠れ変数」が考えられるときの最適化法 混合分布を効率的に推定できます

- 混合正規分布の最尤推定も、勾配法で行ってよいが…
- EM (Expectation-Maximization) アルゴリズム
  - 本来なかった「隠れ変数」の存在が自然に導入できるようなモデルの最尤推定法
    - 混合正規分布は、正規分布をひとつ選んで、データを生成していると考えられる
    - 「どのデータがどの正規分布から発生したか」を「隠れ変数」として導入
  - 2つのステップの繰り返しアルゴリズム
    1. 隠れ変数を固定したときのパラメータの最尤推定
       - 単一の正規分布の最尤推定は、閉じた形で求まる
    2. パラメータを固定したときの隠れ変数の推定

混合正規分布のためのEMアルゴリズム：メンドウなので、K-meansアルゴリズムを紹介します

- 混合正規分布（$\mathbf{\Sigma}^{(k)}$は固定）

$$P(\mathbf{x}; \{w^{(k)}\}, \{\boldsymbol{\mu}^{(k)}\}) := \sum_{k=1}^{K} w^{(k)} g^{(k)}(\mathbf{x}; \boldsymbol{\mu}^{(k)})$$

$$g^{(k)}(\mathbf{x}; \{\boldsymbol{\mu}^{(k)}\}) := \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}^{(k)}|^{1/2}} \exp\left(-(\mathbf{x} - \boldsymbol{\mu}^{(k)}) {\mathbf{\Sigma}^{(k)}}^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(k)})\right)$$

において、対数尤度↓を最大化するパラメータを求めたい

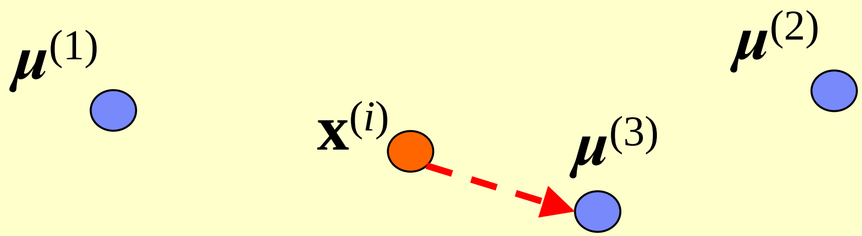
$$L := \sum_{i=1}^{N} \log P(\mathbf{x}^{(i)}; \{w^{(k)}\}, \{\boldsymbol{\mu}^{(k)}\})$$

- 煩雑になるので、単純化して、インチキEMアルゴリズム（*K*-meansアルゴリズム）を導くことにする

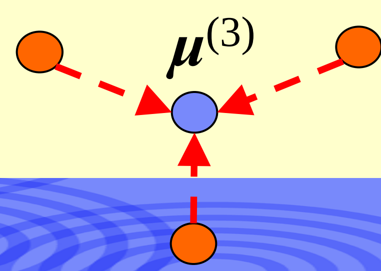
K-meansアルゴリズム：隠れ変数を考えることで、簡単な繰り返しアルゴリズムになります

- 混合正規分布は、正規分布を一つ選んで、それを使って $\mathbf{x}$ を生成していると解釈できる
- 各データ $\mathbf{x}^{(i)}$ が、K 個の正規分布のどれからでてきたのかはわからない。 もしわかっていれば、平均によって正規分布のパラメータが推定できた $\boldsymbol{\mu} = \sum_i \mathbf{x}^{(i)} / N$
- そこで、以下のステップを収束するまで繰り返す

1. 各データ$\mathbf{x}^{(i)}$を、最寄の平均をもつ正規分布に所属させる



2. 各正規分布に所属したデータから、それぞれの平均を新たに求める



学習アルゴリズムのオンライン化：大規模なデータを扱うときに有効です

- 勾配法、EM法、ともに各繰り返しは、(データ数 N)×(次元数D)に比例した時間がかかる
- しかし、
  - データ数が非常に大きいときには、結構時間がかかる
  - 実際には、本当にキッチリ最適化する必要も無い

    （モデルもホントだかどうだか…）
  - 時間とともにデータが到来するような場合もある
    - 時間とともに、正解のモデルも変化するかもしれない
- そこで、オンライン学習（逐次学習）アルゴリズム：訓練データをひとつずつ処理する
  - 人間の学習のイメージにちかい（「だんだん」「試行錯誤」）

ロジスティック回帰の勾配法のオンライン化：ひとつのデータに対しての対数尤度の勾配を用います

- データ $(\mathbf{x}^{(i)}, y^{(i)})$ について注目して最適化を行う
- 1つのデータに注目したときの対数尤度

$$
\begin{aligned}
L^{(i)}(\mathbf{w}) &:= \log P(y^{(i)}|\mathbf{x}^{(i)};\mathbf{w}) \\
&= \begin{cases} \log \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) & \text{if } y^{(i)} = +1 \\ \log(1-\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) & \text{if } y^{(i)} = -1 \end{cases}
\end{aligned}
$$

- 勾配の方向にパラメータを少し更新

$$
\begin{aligned}
\mathbf{w}^{\text{NEW}} &\leftarrow \mathbf{w}^{\text{OLD}} + \alpha \nabla_{\mathbf{w}} L^{(i)}(\mathbf{w}) \\
&= \mathbf{w}^{\text{OLD}} + \alpha \begin{cases} \left(1-\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})\right)\mathbf{x}^{(i)} & \text{if } y^{(i)} = +1 \\ -\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})\mathbf{x}^{(i)} & \text{if } y^{(i)} = -1 \end{cases}
\end{aligned}
$$

- 1ステップの計算量は$O(D)$

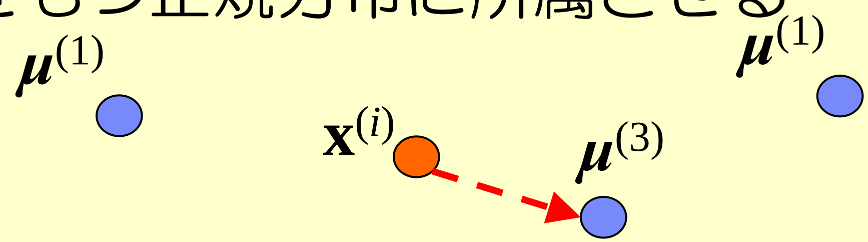
EMアルゴリズム（*K*-means）のオンライン化：オンライン異常検知を行うためには必須です

- オンライン異常検知：データが時々刻々流れてくる中で
  - モデル推定（モデルを逐次的に更新）
  - 異常検知（おかしなデータを発見）

  を同時に行う

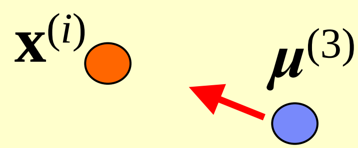
ここで、最寄の平均への距離が大きければ異常データと判断

- 以下のステップを繰り返す
  1. $\mathbf{x}^{(i)}$を、最寄の平均をもつ正規分布に所属させる (バッチ版と同じ)



  2. その最寄の平均を$\mathbf{x}^{(i)}$に（ちょっと）近づける

$$\boldsymbol{\mu}^{\mathrm{NEW}} \leftarrow (1-\epsilon)\boldsymbol{\mu}^{\mathrm{OLD}} + \epsilon \mathbf{x}^{(i)}$$



  - ε は正の小さい値

ここまでのまとめ：最尤推定のための数値計算アルゴリズム（勾配法とEMアルゴリズム）

- 最尤推定を数値的に行うためのアルゴリズム
  - 勾配法
  - EMアルゴリズム
- 大規模データを効率的に行うための方法としてオンライン学習アルゴリズム
  - ロジスティック回帰のオンライン化
  - K-meansアルゴリズムのオンライン化
    - オンライン異常検知
- 実際に学習してみたものの、その結果の良し悪しはどのように判断したらよいだろうか？

機械学習手法の評価

評価方法：何をもって学習の良し悪しを計るか？
まだ見ぬデータに対する性能が真の性能であると考えます

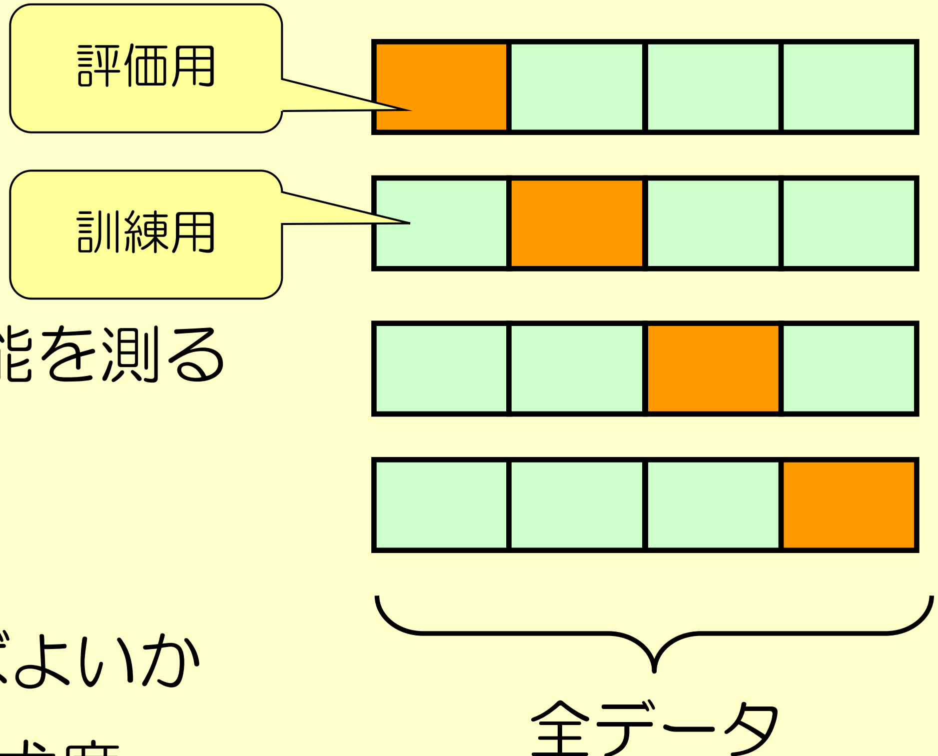
- 実際に学習してみたものの、その結果の良し悪しはどのように判断したらよいだろうか？
- モデルは、まだ見ぬデータに対してうまく働く必要がある
  - 教師無し学習：未知の入力 $\mathbf{x}$ に対して高い確率を割り当てる
  - 教師付き学習：カテゴリ y が未知の入力 $\mathbf{x}$ に対して正しいカテゴリを振る
- 訓練データとテストデータに分けて評価を行う
  - 訓練データ：モデルをつくるためのデータ
  - テストデータ：モデルの性能を評価するためのデータ（＝将来のデータとして、まだ見ていないことにする）

全データ	訓練用	評価用

交差確認法（クロスバリデーション）：まだ見ぬデータに対する性能を評価することができます

- 全体を K 等分し、
  - そのうち K-1 個を訓練用に
  - 1個を評価用に使う

を K 回繰り返し、その平均的な性能を測る



- では、性能を測る指標として、
  具体的にどのような指標を使えばよいか
  - 教師無し学習：（テスト）対数尤度
  - 教師付き学習：正解率、AUC

教師無し学習の場合、テストデータに対する対数尤度

- テストデータに対する対数尤度

$$L := \sum_{i \in \text{test set}} \log P(\mathbf{x}^{(i)})$$

  - テストデータと訓練データが同じ分布から出ているのであれば、訓練データに対して高い確率を与えるモデルは、テストデータに対しても高い確率を与えるはず

- もしくは、クラスタリングなどの場合に、正しいクラスラベルが分かっていたりすれば、それとの一致具合も使われる

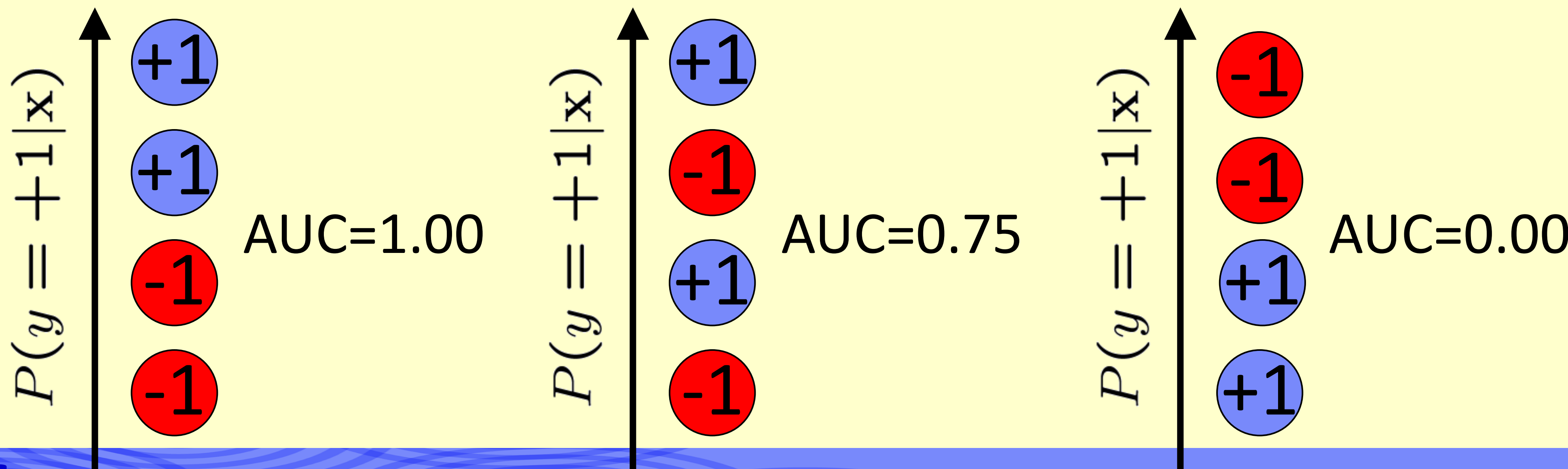
教師付き学習の場合、尤度より正解率のほうが評価値として好ましいが、評価値が閾値に依存するという問題があります

- 教師付き学習の場合にも、対数尤度を使ってよい
- が、本当はモデル $P(y=+1|\mathbf{x})$ の出力を用いて予測したカテゴリが正しいかどうかに興味がある
  - $P(y=+1|\mathbf{x}) \geq 0.5$ であれば、カテゴリ +1 と予測
  - $P(y=+1|\mathbf{x}) < 0.5$ であれば、カテゴリ -1 と予測
- 正解率 := (テストデータ中での正解数) / (テストデータの数)
- しかし、
  - 精度が閾値に依存してしまう
  - 特に、カテゴリのバランスが悪いときに
    - $P(y=+1|\mathbf{x})$ が0.5よりも全体的に高め／低めに出るので評価のベースラインがわからない

AUC: 閾値に依存しない教師付き学習の定番評価値

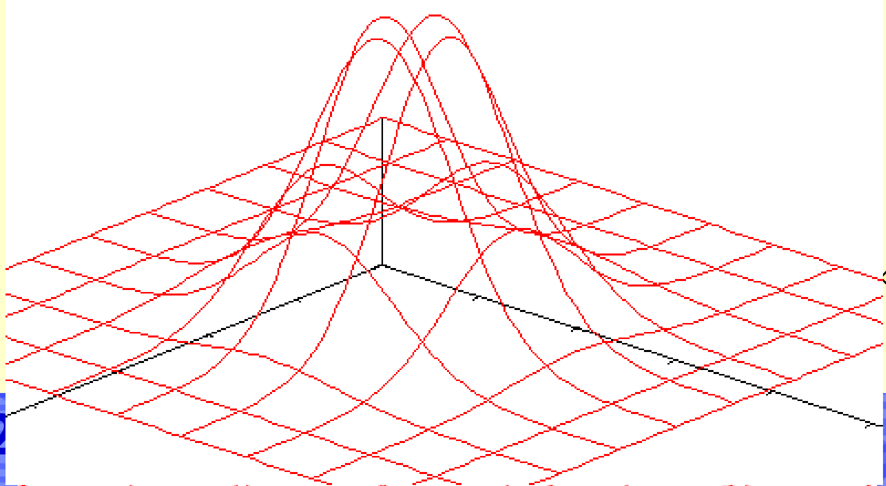
- ある閾値を決めたときの正解率ではなく、$P(y=+1|\mathbf{x})$ の相対的な順序に依存した指標
- AUC（Area Under the Curve）とは：
  - あるカテゴリ+1 の $\mathbf{x}^{(i)}$ をランダムに選び
  - あるカテゴリ-1 の $\mathbf{x}^{(j)}$ をランダムに選んだとき
  - $P(y^{(i)}=+1|\mathbf{x}^{(i)}) > P(y^{(j)}=+1|\mathbf{x}^{(j)})$ であるような確率

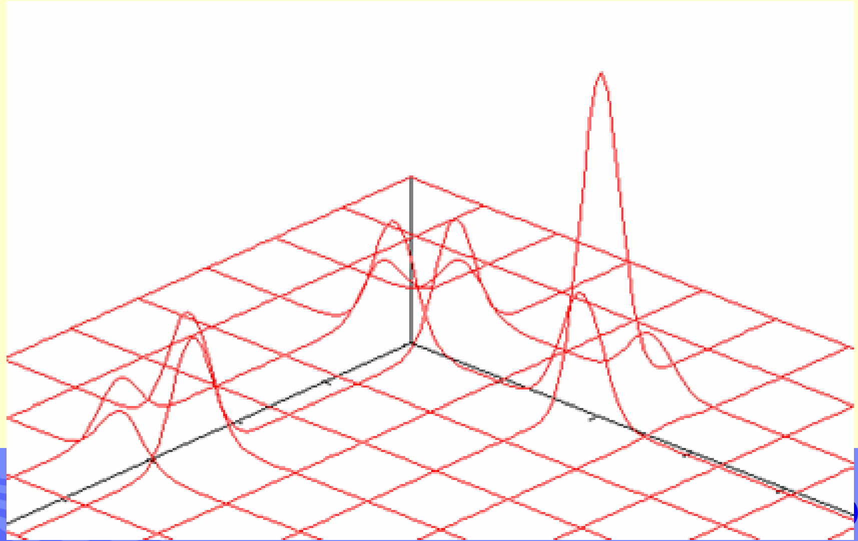
ちなみに、金融では、ほぼ同様の指標がAR値と呼ばれる



過学習：訓練データに適応しすぎると、性能が悪くなる現象

- （教師付き学習において）訓練データそのものを覚えてしまえば、訓練データに関しては100％正解できる
  - しかし、本当は、訓練データに含まれていないデータに対して正解したい
- 訓練データに拘りすぎると、性能が悪くなる現象「過学習」がおこる
  - とくに **x** が高次元のデータを扱うときに起こる
  - データの数に対して、モデルの自由度（パラメータの数）が大きすぎると、おこりやすい





正則化：訓練データへの過適合を防ぐ方法

- 尤度だけではなく「関数の滑らかさ」を表す項を目的関数に加える
- 具体的には、パラメータのノルム $\| \mathbf{w} \|$（ベクトルの大きさ）を使うことが多い
  - ノルムが大→極端なモデル
  - ノルムが小→滑らかなモデル

ロジスティック回帰（教師付き学習）の場合

$$L := \sum_{i=1}^{N} \log P(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w})$$

パラメータwに依存することを明示的にするためにこのように書く

$$L := \sum_{i=1}^{N} \log P(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) \quad - \lambda \| \mathbf{w} \|$$

ノルムがペナルティ項として加わる

λは適当な正の定数
（対数尤度とのバランスをとる）

L2-正則化（リッジ正則化）：もっとも一般的な正則化法

- ノルムとして2-ノルムを用いる

$$\| \mathbf{w} \| := \| \mathbf{w} \|_2^2 = w_1^2 + w_2^2 + \cdots + w_D^2$$

- これを対数尤度にペナルティ項として加えると

$$L := \sum_{i=1}^{N} \log P(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) - \lambda \| \mathbf{w} \|_2^2$$

- もっとも一般的に用いられる正則化法
- λ の決め方は後述

L1-正則化（ラッソ正則化）：スパース（疎）な解を得られる正則化法

- ノルムとして1-ノルムを用いる

$$\| \mathbf{w} \| := |\mathbf{w}|_1 := |w_1| + |w_2| + \cdots + |w_D|$$

- これを対数尤度にペナルティ項として加えると

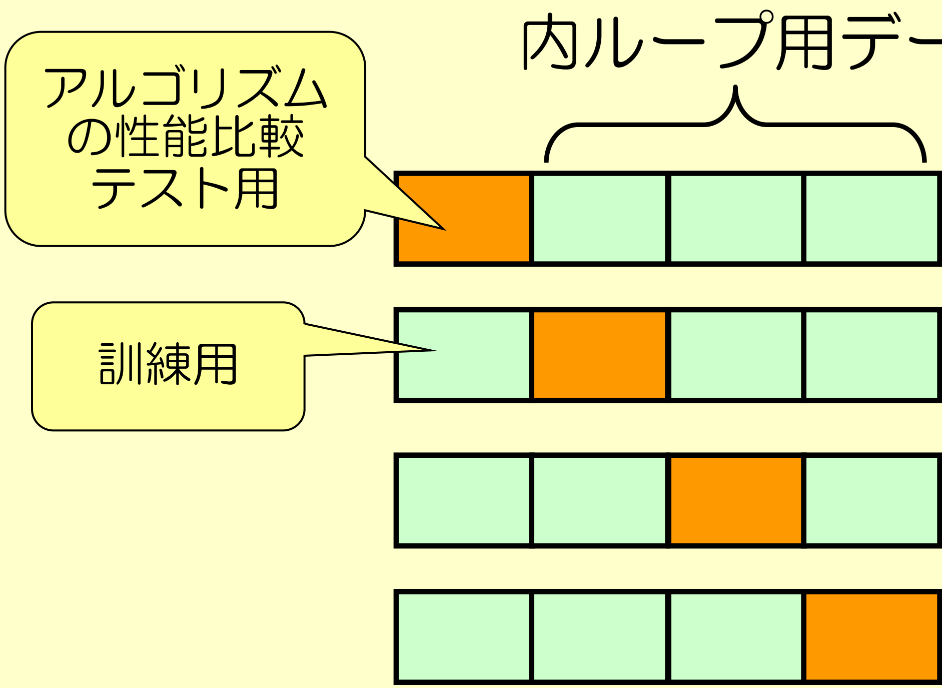
$$L := \sum_{i=1}^{N} \log P(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) - \lambda |\mathbf{w}|_1$$

- 得られる $\mathbf{w}$ が疎 になることが知られている
  - $\mathbf{w}$ の要素の多くが 0 になる
- $\mathbf{x}$ の次元が高いときに有効
  - テキスト分類など

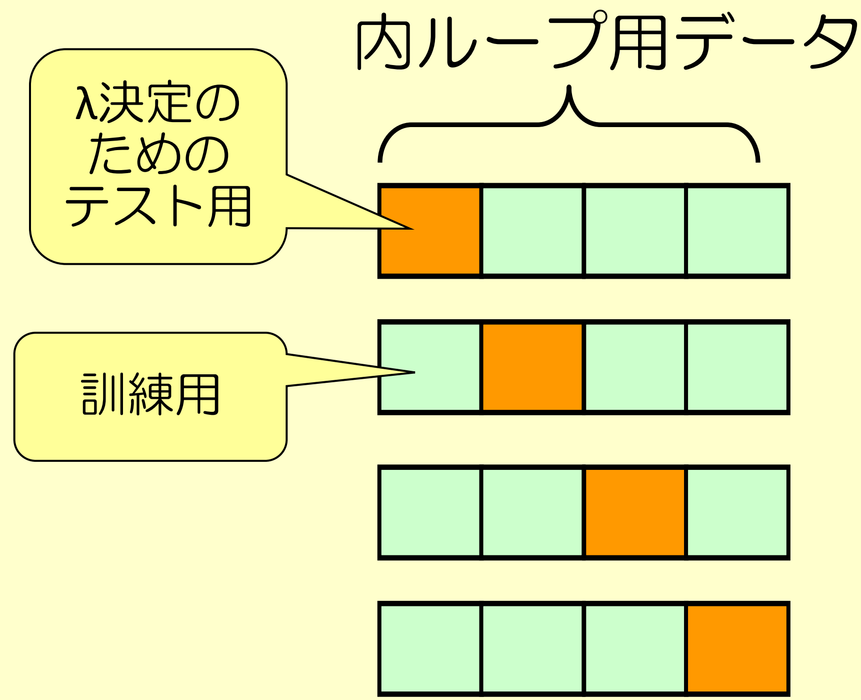
ハイパーパラメータλの決定：交差検定を利用することで決定できます

- 交差検定によって決定する
- 複数の学習アルゴリズムの比較には、交差検定の2重ループ

外ループでは、内ループで決定されたλを使って性能評価



内ループでは、さらに交差検定を行いλを決定



ここまでのまとめ：性能評価方法と過学習の問題、過学習を避けるための正則化法

- 性能評価の方法として、訓練データとテストデータに切り分けて複数回の評価を行う交差確認法（クロスバリデーション）
- 評価値としては、
  - 教師無し学習：テスト尤度
  - 教師付き学習：正解率、AUC
- 訓練データに適合しすぎて、性能が悪化する過学習の問題
- 過学習の解決法として、パラメータのノルムをペナルティ項として目的関数に加える正則化法
  - L2正則化（リッジ正則化）：世界標準
  - L1正則化（ラッソ正則化）：次元削減の効果あり