分類問題

内山将夫@NICT
mutiyama@nict.go.jp

分類問題とは何か？

非常に一般的には，入力xに対して，出力yを与えること．自然言語処理においては，yは名義尺度であることが多い．

分類問題の例

- 2値分類問題
- 多値分類問題
- 複雑な出力に対する分類問題

(内山は，分類問題については，それほど良く知らないため，そのつもりで，この資料を読んで下さい．)

4つの尺度について簡単に

- 名義尺度 (黒か白かなど．異なるクラスは互いに異なるということが言える)
- 順序尺度 (1番，2番,... 順番があるので，大小比較ができる)
- 間隔尺度 (温度など．間隔が同じなので，足し算と引き算ができる)
- 比尺度 (長さなど．0点が定まるので，かけ算と割り算ができる)

テキストに関係する2値分類問題の例

- マーケティングにおいて，ある製品について，自由回答アンケートをとったとき，あるアンケートxが，その製品について
  - 不満を述べているか
  - 述べていないか

  を判定する．これを知ることは，ユーザーの不満を解消し，満足度を向上させるために，必要である．
- メールについて，
  - 迷惑メールか
  - そうでないか

  を判定する．これは，現代社会において，必須である．

多値分類の例

- ある記事があったとして，それが
  - 経済
  - 政治
  - スポーツ...

  のいずれのジャンルに属するかを判定する．これは，Web上から様々な記事を集め，それらから新聞を作りたいときなどに必要である．
- Webページを記述するには，様々な言語がある．それらの言語のうちで，当該のページは，
  - 日本語で記述されているか
  - 英語で記述されているか
  - 中国語で記述されているか...

  を判定する．これは，Webブラウザなどで，正しくページを表示するために必要である．

複雑な出力の分類

- 構文解析等では，入力した文の構文構造という，複雑な出力が必要になる

自然言語処理や生物情報学等では，複雑な出力の分類が必要になる．

2値分類問題の表現形式の例

- 入力ベクトル $\mathbf{x} = [x_1, x_2, ..., x_{n-1}]$
- 出力

$$y = \begin{cases} +1 & \text{望みのクラスのとき} \\ -1 & \text{それ以外のとき} \end{cases}$$

- 重みベクトル $\mathbf{w} = [w_1, w_2, ..., w_{n-1}]$
- 閾値 b

判定法

$\mathbf{w} \cdot \mathbf{x} = \Sigma_{i=1}^{n-1} w_i x_i \geq b$ のとき $+1$を出力し，$\mathbf{w} \cdot \mathbf{x} < b$のとき -1を出力する．

説明のときには，bを特別扱いすると面倒なので，$x_n = -1, w_n = b$とすることにより，$\mathbf{w} \cdot \mathbf{x} \geq 0$のとき，$+1$を出力し，$\mathbf{w} \cdot \mathbf{x} < 0$のとき -1を出力する．

つまり

- 入力ベクトル $\mathbf{x}$
- 出力値 y
- 重みベクトル $\mathbf{w}$

により(線形な問題については)2値分類を定式化できる．

迷惑メールの判定

分類対象のメールxについて，そのベクトル表現を$\mathbf{x} = [x_1, x_2, x_3]$とする．ここで，

$$x_1 = \begin{cases} 1 \text{ if 「お金」という言葉を含む} \\ 0 \text{ 上記以外} \end{cases}$$

$$x_2 = \begin{cases} 1 \text{ if 何らかの URL を含む} \\ 0 \text{ 上記以外} \end{cases}$$

$$x_3 = \begin{cases} 1 \text{ if ac.jp アドレスから発信されている} \\ 0 \text{ 上記以外} \end{cases}$$

のとき $\mathbf{w} = [1, 1, -2]$ とする．$\mathbf{w} \cdot \mathbf{x} \geq 0$のとき，迷惑メールと判定する．
さて，

- $\mathbf{x}_1 = [1, 0, 1]$, $\mathbf{x}_2 = [1, 1, 0]$

とする．つまり，$\mathbf{x}_1$は,「お金」という言葉を含み,「ac.jp」から発信されている．また，$\mathbf{x}_2$は,「お金」という言葉を含み，何らかのURLがある．
このとき $\mathbf{w} \cdot \mathbf{x}_1 = 1 - 2 = -1$, $\mathbf{w} \cdot \mathbf{x}_2 = 1 + 1 = 2$である．つまり，$\mathbf{x}_1$は迷惑メールでなく，$\mathbf{x}_2$は迷惑メールであると判定する．

分類器を使うときに注意しないといけないこと

分類器の性能は100%ではないので，誤ったときにどうするかを考えておかないといけない．誤りには次の2通りがある

- 誤検出：普通のメールを迷惑メールとして判定すること
- 見逃し：迷惑メールを普通のメールと判定すること

迷惑メールの判定のときには，誤検出をなるべく少なくする必要がある．一方では，見逃しは多少は許容できる．
なぜなら，普通のメールが迷惑メールとして誤検出されるとすると，そのメールは，読まれない可能性が高いからである．もし，そのメールに重要なことが書いてあったとすると，それが読まれないことによる損失は，多大なものになる可能性がある．
一方，迷惑メールを普通のメールとして見逃して，ユーザーに配送したときには，ユーザーは，即座に，そのメールが迷惑メールであることを判定できる可能性が高い．そのため，迷惑メールがユーザーに配送されても，その量が多くなければ，それほどは困らない．
誤検出と見逃しのどちらを重視すべきかは，アプリケーションにより異なる．

分類問題において何が問題か

- どういう分類問題を解くべきか.
  手持ちの問題があるとして，それをどのようにして分類問題として定式化するか.
- 入力をどのようなベクトルとして表現するか
- 重みベクトルをどう設定するか

手持ちの問題をどう分類問題として定式化するか

- 実際の問題は，普通は，非常に複雑である
- 一方，2値分類問題は，入力を +1 か -1 の2値にわけるだけの単純な問題にしか適用できない
- そのため，2値分類問題を複雑な問題に適用するためには，元の問題の中で適切な抽象化をする必要がある.

例：アマゾンのレビューより

「ぐりとぐら」は子供の頃に叔母からプレゼントされた絵本の中の一冊。「しろくまちゃんのほっとけーき」と並んで料理する楽しさに目覚めるきっかけになりました。母に毎日「読んで～」とせがみ、2歳半からはお菓子作りの好きな父と一緒に台所で手伝うように…。懐かしくて大人になった今でももう一度読みたくなって買ってしまいました。リズムや美味しい匂いがしてきそうな絵が何度読んでも楽しいです。子供ができたらその子にもぜひ読んであげたい…私の大切な絵本です。(By 蜜花)

==========

のような書評について，これは，結局，

- 「ぐりとぐら」を勧めているのか，そうでないのか

を判定するのが，2値分類問題である．これだけでは，このレビューに含まれる情報の極くわずかしか伝えていないのは，間違いないと思われる．しかし，「勧めているのか」「いないのか」という重要情報が取り出されることも確かである．

一方，迷惑メールの判定のときには，対象が迷惑メールかどうかさえ判定できれば，十分である．迷惑メールの内容には興味はない．

分類問題を取り扱うときには，それにより，取り扱いたい問題が適切に取り扱えているかに注意する必要がある．

入力ベクトル$\mathbf{x}$をどうつくるか

ベクトルの要素としては，任意のものが考えられるが，テキストが入力のときには，単語iが入力テキストx中に存在するときに，$x_i = 1$とし，そうでないときに，$x_i = 0$としたり，あるいは，良く使われるものとして，

$$x_i = \mathrm{tf}_i \times \mathrm{idf}_i$$

がある．ただし，n_iを単語iが，入力テキスト中に出現する回数としたとき，

$$\mathrm{tf}_i = n_i$$

や

$$\mathrm{tf}_i = \log(n_i + 1)$$

としたり，Nを訓練データにおける文書の総数とし，d_iを単語iが出現する文書の数とするとき，

$$\mathrm{idf}_i = \log \frac{N}{d_i}$$

が利用されることが多い．

tf × idfについて

tf × idf には，さまざまな変種があるが，その源流は非常に古い．idf は，以下がオリジナルである．

- Karen Spärck Jones, A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28, 11-21, 1972.

また tf については，以下に記述があるらしい(上記論文からの孫引き)．

- Salton, G., 1968. Automatic Information Organization and Retrieval, McGraw-Hill

tf × idfは，テキスト中の単語の重要度を表すと考えられている．tf は，テキスト中に出現する回数が多い単語ほど大きくなる．idf は，コーパス全体で出現する回数が小さい単語ほど大きくなる．したがって，tf × idfは，コーパス中では余り出現しないが，そのテキストに限っては，良く出現するような単語で大きくなる．このような単語は，そのテキストのキーワードと言えるだろう．

$\mathbf{w}$をどう設定するか

問題が定義でき，入力がベクトルに変換できれば，あとは重み$\mathbf{w}$があれば，分類問題を解くことができる．この重み$\mathbf{w}$は，訓練データから決めることができる．
一般的に

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ...$$

が訓練データとして与えられたとき，そこから，$\mathbf{w}$を学習することが考えられる．
様々な学習法があるが，簡単なものとして，

- Naive Bayes モデル
- パーセプトロン

を説明する．

Naive Bayes モデル

Naive Bayes では，$\mathbf{w} = [1, 1, ..., 1]$ と固定されていて，$\mathbf{x}$ を工夫していると考えることができる．

Naive Bayes による 2 値分類器は以下のように導出可能である．

$P(+1|\mathbf{x})$ と $P(-1|\mathbf{x})$ を，入力 $\mathbf{x}$ を条件とする，クラス $+1$ と -1 の確率とする．このとき，

$$P(+1|\mathbf{x}) \geq P(-1|\mathbf{x})$$

すなわち，

$$\log \frac{P(+1|\mathbf{x})}{P(-1|\mathbf{x})} \geq 0$$

のときには，$+1$ と判定する．これは，次のように展開できる．

Naive Bayes モデル

$$\log \frac{P(+1|\mathbf{x})}{P(-1|\mathbf{x})} = \log \frac{P(\mathbf{x}|+1)P(+1)}{P(\mathbf{x}|-1)P(-1)}$$

ここで，$\mathbf{x}$中に単語iがn_i回出現するとすると，

$$P(\mathbf{x}|+1) = \prod_i P(w_i|+1)^{n_i}$$

ただし，

$$P(w_i|+1) = \frac{\text{クラスが}+1\text{の訓練テキスト中での}w_i\text{の総頻度}}{\text{クラスが}+1\text{の訓練テキストの総単語数}}$$

これらは，$P(\mathbf{x}|-1)$や$P(w_i|-1)$についても同様．よって，

$$\log \frac{P(+1|\mathbf{x})}{P(-1|\mathbf{x})} = \sum_i n_i \log \frac{P(w_i|+1)}{P(w_i|-1)} + \log \frac{P(+1)}{P(-1)}$$

$\log \frac{P(+1)}{P(-1)}$は訓練テキスト中での$+1$のテキストと-1のテキストの比から計算可能であるが，通常は，この項は，テストにおいては無視する．その理由は，テストにおける$+1$と-1の比は，訓練における比と違うことが多いからである．しかし，もし，両者の比が同じであれば，この項を残しておいた方が良い．以下では，この項は残しておかないものとする．

Naive Bayes モデル

以上より，

$$\mathbf{w} = [1, 1, ..., 1]$$

$$\mathbf{x} = [x_1, x_2, ..., x_n]$$

$$x_i = n_i \log \frac{P(w_i|+1)}{P(w_i|-1)}$$

とすれば，

$$\mathbf{w}\cdot\mathbf{x} = \sum_i n_i \log \frac{P(w_i|+1)}{P(w_i|-1)} = \log \frac{P(\mathbf{x}|+1)}{P(\mathbf{x}|-1)} \sim \log \frac{P(+1|\mathbf{x})}{P(-1|\mathbf{x})}$$

であり，$\mathbf{w}\cdot\mathbf{x} \geq 0$ のときに，クラス$+1$が事後確率が高くなるので，クラス$+1$を選択する．

Naive Bayes という名の由来は，

$$P(\mathbf{x}|+1) = \prod_i P(w_i|+1)^{n_i}$$

というように，単語間の条件付きの独立性を仮定しているところからのようである．

Naive Bayes モデルは，簡単で，かつ，性能も，他の複雑なモデルと比べて，大きく劣るというわけではない．また，訓練データが少ないときには，他のモデルよりも良い性能を示すこともある．そのため，Naive Bayes はベースラインとして使われることが多い．

スムージングについて

$$P(w_i|+1) = \frac{\text{クラスが}+1\text{の訓練テキスト中での}w_i\text{の総頻度}}{\text{クラスが}+1\text{の訓練テキストの総単語数}}$$

としたが，これでは，クラス$+1$にはでたが，-1にはでない単語などの扱いに苦労する． x_iの重みとしては，

$$x_i = n_i \log \frac{P(w_i|+1)}{P(w_i|-1)}$$

なので，スムージングなしだと，$n(w_i|+1)$を，クラスが$+1$の訓練テキスト中でのw_iの総頻度として

$$x_i = n_i \log \frac{n(w_i|+1)}{n(w_i|-1)}$$

となる．これにスムージングをするとすると，

$$x_i = n_i \log \frac{n(w_i|+1)+0.5}{n(w_i|-1)+0.5}$$

とするとかが考えられる．

パーセプトロン

$\mathbf{x}$を Naive Bayes により，

$$x_i = n_i \log \frac{P(w_i|+1)}{P(w_i|-1)}$$

としたとする．次に，$\mathbf{w}$を訓練することが考えられる．もし，Naive Bayes モデルにおける独立性の仮定が正しければ，$\mathbf{w}$の最適値は，$[1, 1, 1, .., 1]$である．しかし，実際には，独立性の仮定は，正しくないことが多いので，重みを適切に変更することにより，分類精度が向上する可能性が高い．
重みの推定方法には，

- SVM (Support Vector Machine)
- Boosting
- Logistic regression
- Perceptron

などの様々な方法があるが，ここでは，簡単で，かつ，性能もほどほどに良く，複雑な問題にも応用が可能なパーセプトロンについて説明する．

パーセプトロンのアルゴリズム

訓練データ $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)$ に対して，T回の繰り返しにより，重み $\mathbf{w}$ を推定する場合

$\mathbf{w} = [0, 0, ...]$
for $t = 1..T$
　for $i = 1..n$
　　$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i)$
　　if $y_i \neq \hat{y}$ then
　　　$\mathbf{w} = \mathbf{w} + y_i \cdot \mathbf{x}_i$
　　end
　end
end

$y_i \neq \hat{y}$ のときにのみ，$\mathbf{w}$ を変更する．さて，$y_i = 1$ のときに，重みが変更されたとすると，$\mathbf{w}'$ を新しい重みとすると，$\mathbf{w}' \cdot \mathbf{x}_i = (\mathbf{w} + \mathbf{x}) \cdot \mathbf{x} = \mathbf{w} \cdot \mathbf{x} + \mathbf{x} \cdot \mathbf{x}$．つまり，$+1$ のクラスを誤って，-1と分類した場合には，$\mathbf{w}$ を $\mathbf{w}'$に更新することにより，$\mathbf{x} \cdot \mathbf{x} > 0$のだけの重みが加算されることになり，少しだけ，$+1$に有利な重みになる．
同様に，$y_i = -1$のときには，$\mathbf{w}' \cdot \mathbf{x}_i = (\mathbf{w} - \mathbf{x}) \cdot \mathbf{x} = \mathbf{w} \cdot \mathbf{x} - \mathbf{x} \cdot \mathbf{x}$なので，$\mathbf{x} \cdot \mathbf{x}$だけ少なくなり，クラス$-1$に分類されやすくなる．

平均化したパーセプトロン

$\mathbf{w} = [0, 0, ...]$
$\mathbf{v} = [0, 0, ...]$ (A)
for $t = 1..T$
  for $i = 1..n$
    $\hat{y} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}_i)$
    if $y_i \neq \hat{y}$ then
      $\mathbf{w} = \mathbf{w} + y_i \cdot \mathbf{x}_i$
    end
    $\mathbf{v} += \mathbf{w}$ (B)
  end
end
$\mathbf{w} = \frac{\mathbf{v}}{nT}$ (C)

(A),(B),(C) の部分を除くと，最初のアルゴリズムと変わらない．しかし，こうすることにより，性能が向上することが知られている．
これは，全ての訓練データにおける重みの平均を重みとしている．
平均化しないパーセプトロンの性能は，それほど高くないが，平均化することにより，性能が高くなることが知られている．
一番性能が高い類器には性能は及ばないが，それに近い性能がでる．また，非常に簡単なモデルなので，ハードウェア化もできるため，大量データを処理したい場合には，有効であると思われる．

多値分類

多値分類は，2値分類器を利用するとできる．

one-vs-rest 法

n個のクラスがあるときには，n個の2値分類問題を考える．$\mathbf{w}^{(i)}$が，クラスiとそれ以外のクラスとを分類する重みベクトルとすると，

$$\arg\max_i \mathbf{w}^{(i)} \cdot \mathbf{x}$$

なる．クラスiに分類する．このとき，$\mathbf{w}^{(i)} \cdot \mathbf{w}^{(i)} = 1$となるように，重みベクトルの長さを正規化しておく．

pairwise 法

n個のクラスがあるときに，$\frac{n(n-1)}{2}$個の2値分類問題を考える．$\mathbf{w}^{(i,j)}$が，クラスiとjを分類する重みベクトルとすると$\mathbf{w}^{(i,j)} \cdot \mathbf{x} > 0$のとき$i$に分類されるとして，$\mathbf{w}^{(i,j)} \cdot \mathbf{x} < 0$ は $\mathbf{w}^{(j,i)} = -\mathbf{w}^{(i,j)}$と考えて，$\mathbf{w}^{(j,i)} \cdot \mathbf{x} > 0$ とすると，

$$\arg\max_i \sum_j [\mathbf{w}^{(i,j)} \cdot \mathbf{x} > 0]$$

ただし，[事象]は事象が正のときに 1，そうでないときに 0.

これは，クラスiのスコアとして，クラスiとその他のクラスとを全部比較して，クラスiとして判定された個数を利用する．

複雑な出力の分類

入力が x のとき，その出力の候補を $\mathrm{Gen}(x)$ とする．$y \in \mathrm{Gen}(x)$ について，ベクトル $\Phi(x, y)$ を割当てられるとする．このとき，

$$\hat{y} = \arg \max_{y \in \mathrm{Gen}(x)} \mathbf{w} \cdot \Phi(x, y)$$

なる $\hat{y}$ を出力とする．
つまり，複雑な出力については，

- まず，解候補集合 $\mathrm{Gen}(x)$ を得る．
- 次に，スコアが最大の解を得る．

という2段階のステップを踏む．その理由は，出力が複雑になると，解候補の数が非常に多くなり，絞り込みをしてからでないと，上手くいかないからである．

複雑な出力と，そのベクトル化の例

たとえば，入力xが

the man saw the dog

のとき，出力候補として

- y_1 = the/D man/N saw/V the/D dog/N
- y_2 = the/D man/N saw/N the/D dog/N

の2つがあるとする．このとき，

- 第1要素を D の数
- 第2要素を N の数
- 第3要素を V の数

とすると，

- $\Phi(x, y_1) = [2, 2, 1]$
- $\Phi(x, y_1) = [2, 3, 0]$

である．このとき，重みとして，文にはVが必ず必要なので，少し，それの重みを重くしたベクトル$\mathbf{w} = [1, 1, 2]$を利用するとすると，$\mathbf{w} \cdot \Phi(x, y_1) = 6$と$\mathbf{w} \cdot \Phi(x, y_2) = 5$なので，$y_1$が優先される．

平均化したパーセプトロン

$\mathbf{w} = [0, 0, ...]$
$\mathbf{v} = [0, 0, ...]$
for $t = 1..T$
　for $i = 1..n$
　　$\hat{y} = \arg\max_{y \in \mathrm{Gen}(x)} \mathbf{w} \cdot \Phi(x, y)$
　　if $y_i \neq \hat{y}$ then
　　　$\mathbf{w} = \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, \hat{y})$
　　end
　　$\mathbf{v} += \mathbf{w}$
　end
end
$\mathbf{w} = \frac{\mathbf{v}}{nT}$

2 値分類問題と，ほとんど同じ方法で学習できることがわかる．$\Phi(x_i, y_i) - \Phi(x_i, \hat{y})$ に注目すると，あるベクトル要素について，$\Phi(x_i, y_i)$ における値の方が，予測値 $\Phi(x_i, \hat{y})$ よりも大きければ，その要素については，差分だけを w に加える．これにより，予測値よりも少し重みを加算する．

構造が複雑な問題の例

- 形態素解析
- 構文解析
- 機械翻訳

これらについても，入力 x と出力 y の組をベクトル $\Phi(x, y)$で表現することができれば，機械学習により，重み $\mathbf{w}$ を適切に調整することができる．
しかし，この方法により調整した重み $\mathbf{w}$ を利用したとしても，元々の解候補 $\mathrm{Gen}(x)$ にある解を劇的に向上することはできないようで，少し向上するくらいのようだ．その理由は，出力が複雑になると，そもそも，それをベクトル表現したときに，その表現が適切かどうかという問題がある．さらに，出力が複雑になると解空間が非常に大きくなるので，$\mathrm{Gen}(x)$ によりカバーできる解候補が解空間に比べて，相対的に少なくもなる．

まとめ

訓練データ $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$ から重みベクトル $\mathbf{w}$ を推定する方法として，パーセプトロンを紹介した．

難しい問題

- 手持ちの問題を，意味のある分類問題として定式化すること
- 訓練データを収集すること

課題

- テキスト処理に関係する面白い分類問題を探してみて下さい．
- 分類問題のためのソフトウェアなどを利用して，その問題を解いて下さい．

分類問題のためのソフトウェア

- Weka (http://www.cs.waikato.ac.nz/ml/weka/)
- Mallet (http://mallet.cs.umass.edu/index.php/Main_Page)
- SVM$^{\text{light}}$ (http://svmlight.joachims.org/)
- libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
- BBR (http://www.stat.rutgers.edu/~madigan/BBR/)
- BACT (http://chasen.org/~taku/software/bact/)