Text Categorization

HIRAKATA

Univ. of Tokyo, CS

April 28, 2014

classification

NLP における重要な問題 classification あるいは categorization は任意の object を 2 つ以上の class あるいは category に割り当てること

上 2 つはもうやった。

Problem	Object	Categories
tagging	context of word	the word's tags
disambiguation	context of word	the word's sense
author	document	authors
language	document	languages
text categorization	document	topics

text classification

goal:

```
document -> topic (theme)
```

ニュースを興味在るものにフィルタするとかいう適用例がある

統計的分類の一般的な特徴づけ

訓練データ object があって各々の object は一つ以上のクラスがラベル付けされている

$$(\vec{x}, c)$$

where

- $\vec{x} \in \mathbb{R}^n$ is a vector of measurements,
- c is the class label.

linear classification

$$g(\vec{x}) = \vec{w} \cdot \vec{x} + w_0$$

これの正負によって二値に分類

keywords

- local optimum
  手法に依っては大域的最適解が保証されてたりされてなかったりする
- hill climbing
  線形分離に就いての山登り法として perceptrons がある

正解率

二値分類の場合自然に recall, precision が定義できる (略)

macro-averaging

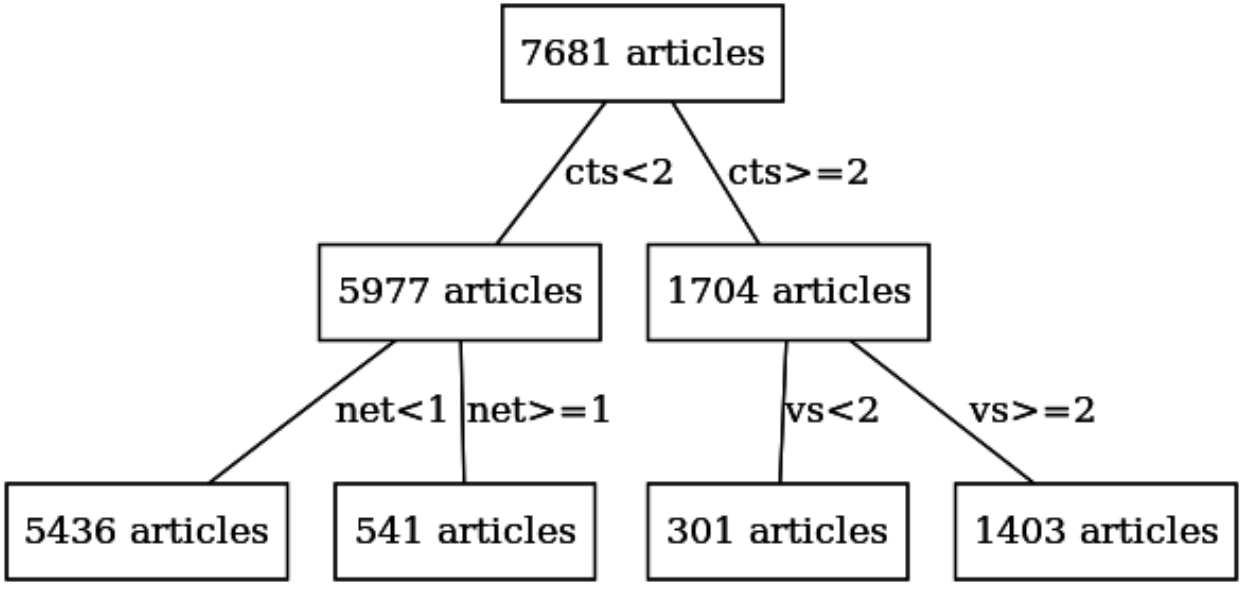
三つ以上のカテゴリ $(c_1, c_2, ..)$ に分類する場合全ての i に就いて c_i v.s. $\neg c_i$ という分割で recall, precision を計算して平均をとる

techniques

本章では次の 4 つの技巧を紹介する

- decision trees
- maximum entropy modeling
- perceptrons
- k nearest neighbor classification

決定木



1. start at the top node
2. test question, branching

K 個の term (word) を定めて document j を次の K 次元の整数ベクトルで表現する

$$x_j = (s_{ij}, \ldots, s_{Kj})$$

where

$$s_{ij} = \text{round}\left(10 \times \frac{1 + \log(tf_{ij})}{1 + \log(\ell_j)}\right)$$

- tf_{ij} とは document j における term i の出現頻度
- ℓ_j とは document j の長さ

例えば 89 words からなる docuemnt j に 6 回 term i が出現した場合そのスコアは $s_{ij} = 10 \times \frac{1+\log(6)}{1+\log(89)} \approx 5.09$

splitting

決定木を作るために次の 2 つの尺度が必要

- splitting criterion
- stopping criterion

stopping criterion は自明である (!) それ以上分割できなくなればつまりあるノードでそこにある要素が全て同じカテゴリーならばそこで止めればよい

information gain

splitting criterion としてここでは maximum information gain を用いる
属性 a について 値 y で分割するときその減少量は

$$G(a, y) = H(t) - H(t|a) = H(t) - (p_L H(t_L) + p_R H(t_R))$$

で表される

剪定

- 過学習を避けるために一旦木を作った後に剪定を行う
- よくある剪定の方法はノードごとにそのノードが寄与してる尺度を計算する (Quinlan 1993)

剪定

- 過学習を避けるために一旦木を作った後に剪定を行う
- よくある剪定の方法はノードごとにそのノードが寄与してる尺度を計算する (Quinlan 1993)
  1. 1つずつノードを消していって n 個の木を作る

剪定

- 過学習を避けるために一旦木を作った後に剪定を行う
- よくある剪定の方法はノードごとにそのノードが寄与してる尺度を計算する (Quinlan 1993)
  1. 1つずつノードを消していって n 個の木を作る
  2. n 個の木について validation を行って (精度を調べて) 最適なものを選択する

剪定による精度の絵

Figure 16.4 参照
剪定するノードの数を増やすにつれて

- training set に対する精度は下がる
- test set に対する精度は中程でピーク

cross-validation

- validation set を用いた手法は training set の大部分が無駄になる
- もっと良い手法として n-fold cross-validation を用いること
  - n 個に分割
  - $n-1$ 個を training set, 1 個を validation set として精度の平均をとる

決定木まとめ

- 決定木はナイーブベイズ分類器, 線形回帰, ロジスティク回帰より複雑
- データセットが小さい場合 上手く分離できず失敗する
- 問題が単純ならば より単純な手法を使うほうが好ましい

最大エントロピーモデルとは

- 異なるソースから成る情報を分類するためのフレームワーク

最大エントロピーモデルとは

- 異なるソースから成る情報を分類するためのフレームワーク
- 分類の問題をいくつかの (たいていは大量の)feature として記述する
- feature とはそのモデルにおける制約
- 制約を満たす中でエントロピーを最大化する
- feature の選択と学習は共に行われる (重み付のこと??)

最大エントロピーモデルを用いる目的はできるだけ不確かさを保つことにある

feature の定義

feature

feature f_i とは

$$f_i(\vec{x_j}, c) = \begin{cases} 1 & s_{ij} > 0 \wedge c = 1 \\ 0 & \text{otherwise} \end{cases}$$

- s_{ij} は前に書いたスコア
- c は $\vec{x_j}$ で記述される document のクラスだけど
- 注目してるあるクラスならば 1 さもなくば 0 とする

loglinear models

maximum entropy modeling の 1 つ

loglinear models

$$p(\vec{x}, c) = \frac{1}{Z} \prod_{i=1}^{K} \alpha_i^{f_i(\vec{x}, c)}$$

α_i が feature f_i に対する重みで
実際の分類には $p(x, 0)$ <> $p(x, 1)$ の比較を行えばよい

generalized iterative scaling

以下の制約を満たすような最大エントロピーを与える分布 p^* を探す

制約

訓練データにおける分布 $\tilde{p}$ での feature の期待値と一致すること

$$E_{p^*}[f_i] = E_{\tilde{p}}[f_i]$$

generalized iterative scaling とはこのような p^* を探すためのアルゴリズム

先の等式

$$E_{p^*}[f_i] = E_{\tilde{p}}[f_i]$$

を満たすために

$$f_{K+1} = C - \sum_{i=1}^{K} f_i$$

を追加する
ここで C は (定数ならなんでもよさそうだけど)

$$C := \max_{(x,c)} \sum_{i=1}^{K} f_i(x, c)$$

を用いる

等式の右辺は簡単に簡約できて

$$E_{\tilde{p}}[f_i] = \sum_{(x,c)} \tilde{p}(x,c) f_i(x,c)$$

$$= \frac{1}{N} \sum_{j=1}^{N} f_i(x_j, c)$$

左辺は $\forall \vec{x}$ として訓練データ中の $\vec{x_1}..\vec{x_N}$ に制限をする近似を用いて計算する (Lau 1004)

$$E_p[f_i] = \sum_{(x,c)} p(x,c) f_i(x,c)$$

$$= \sum_{j=1}^{N} \sum_{c} \frac{1}{N} p(c|x_j) f_i(x_j,c)$$

実際には次のようにして α_i を学習する

1. $p^1 = \{\alpha_i^1\}_{i=1..K+1}$; 何でもいいけど通常は全部 1 にする
2. let $n = 1$
3. $p^{n+1} \leftarrow$ update p^n
   - $\alpha_i^{n+1} = \alpha_i^n r_i$
   - where $r_i = \left(\frac{E_{\tilde{p}}[f_i]}{E_p^n[f_i]} \right)^{1/C}$
4. until converged { let $n = n + 1$; goto 2 }
5. let $p^* = p^n$

パーセプトロン

- gradient descent (山登り法) の簡単な例
- 線形分離する
- decide 'yes' for $\vec{x} \iff \vec{w} \cdot \vec{x} > \theta$

学習

$$\phi(w') = [w1..w_K; \theta] \cdot [x1..x_K; -1]^T$$

w' は左の行ベクトル
ϕ の勾配に沿って w を動かす

$$\nabla\phi(w') = [x; -1]$$

1. 訓練データ中の $(x_j, c); c \in \{-1, 1\}$ について
2. $$\begin{cases} c == (wx_j > \theta) & \Longrightarrow \ continue \\ otherwise & \Longrightarrow \ w' \leftarrow w' + c \times [x_j; -1] \end{cases}$$

k 近傍法; kNN

原理

新しいオブジェクトを分類するには訓練データから一番似てるものを探せばいい

- 肝は似てるとする尺度関数を持ってくること
- (鷲の絵の例え)
- もしそのような尺度を知らないのならば kNN は使えない

1NN algorithm

訓練データ X によって y を分類する

1. $sim_{max}(y) = \max_{x \in X} sim(x, y)$
2. let $A = \{x \in X | sim(x, y) = sim_{max}(y)\}$
3. decide $c = \arg\max_{c} size\{class(x) | x \in A\}$

$k > 1$ ならば similality が k 番目に高いものまでを A に含める

kNN まとめ

- 大事なのは、similarity metric
- k の値はそれに比べれば重要ではない
- 他の肝は計算量 訓練データとの類似度を毎回計算したら大変

以上の手法の比較

for Reuters category “earnings”

決定木	96.0 %
最大エントロピーモデル	93.91 %
パーセプトロン	83.3 %
1-NN	95.3 %