@ishizakiiii 2018年06月14日に更新

機械学習ナイーブベイズ分類器のアルゴリズムを理解したのでメモ。そしてPythonで書いてみた。

Python アルゴリズム 機械学習 MachineLearning データ分析

概要

ナイーブベイズ分類器(ベイジアンフィルター)のアルゴリズムを具体的な数値を使って説明します。また、Pythonで実装してみました。自分の勉強メモのつもりで書いたのですが、他の方の役にも立てたら嬉しいです。

ナイーブベイズ分類器って？

あるデータ(文章)をどのカテゴリーに属するのかを判定させる、機械学習の教師あり学習の手法の一つです。
スパムメールフィルターやWEBニュース記事のカテゴライズによく使われています。

難易度

ベイズの定理を利用した単純な手法で、難易度は低です。
なるべく数式を使わないで説明してみました。

ナイーブベイズ分類器の計算

対象文章がどのカテゴリーに分類されるかを決めるための計算ロジックを、具体的な数値を使って説明します。

学習データが以下である場合、対象文章がどのカテゴリーに分類されるか計算します。

学習データ

- サッカー　[　ボール　|　スポーツ　|　ワールドカップ　|　ボール　]
- 野球　　[　ボール　|　スポーツ　|　グローブ　|　バット　]
- テニス　　[　ボール　|　ラケット　|　コート　]
- サッカー　[　ボール　|　スポーツ　]

※[　]内はつながっている1つの文書だと仮定。

対象文書
「ボールスポーツ」

1.　カテゴリー出現率の計算　P(C)

学習データの各カテゴリーの文書数を、総文書数で割った値を「**カテゴリー出現率 P(C)**」とします。

	サッカー	**野球**	**テニス**
カテゴリー出現確率	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

2.　カテゴリー内の文書出現率の計算　P(D|C)

学習データ内の各カテゴリーの単語数を数えます。
重複は排除せず、サッカーは合計とします。

	サッカー	野球	テニス
単語数	6	4	3

「ボール」と「スポーツ」の各カテゴリーごとの出現回数を数え、上記で数えた各カテゴリーの単語数で割ります。これを「**カテゴリー内の単語出現率 P(Wn|C)**」とします。

	サッカー	野球	テニス
ボール	$\frac{3}{6}$	$\frac{1}{4}$	$\frac{1}{3}$
	2	1	0

計算した「ボールの値」と「スポーツの値」を掛け合わせた値を「**カテゴリー内の文書出現率 P(D|C)**」とします。

	サッカー	野球	テニス
カテゴリー内の文書出現率	$\frac{6}{36}$	$\frac{1}{16}$	$\frac{0}{9}$

3. 文章内のカテゴリー出現率の計算 P(C|D)

上記の1. と2. で出した「**カテゴリー出現率 P(C)**」と「**カテゴリー内の文書出現率 P(D|C)**」を掛け合わせたものが文書に対する各カテゴリーへ属する確率（**文章内のカテゴリー出現率 P(C|D)**）です。

この値が最も高いものが対象文書のカテゴリーと分類します。

	サッカー	野球	テニス
文章内のカテゴリー出現率	$\frac{2}{4} \times \frac{6}{36}$	$\frac{1}{4} \times \frac{1}{16}$	$\frac{1}{4} \times \frac{0}{9}$
計算結果	$\frac{1}{12}$	$\frac{1}{64}$	$\frac{0}{36}$

結果、「ボールスポーツ」は「サッカー」へ分類されました!!

(あくまで例題なので、全部ボールスポーツであることはスルーで)

4. ゼロ頻度問題

この例では、テニスの確率が 0 になりました。「スポーツ」という単語が「テニス」の学習データになかったからです。

学習データには存在しない新ワードがあった場合、カテゴリーの確率が0になってしまいます。
これを「**ゼロ頻度問題**」といいます。

これを解決するために、加算スムージングという方法をとって、再計算します。

加算スムージングで再計算

2.の「カテゴリー内の単語出現率 P(Wn|C)」計算部分に、下記の太字部分の処理を加えるだけです。

「ボール」と「スポーツ」の各カテゴリーごとの出現回数を数えて**1加え**、上記で数えた各カテゴリーの単語数に**学習データ全単語数を加えた値**で割ります。

学習データ全単語数は重複排除するので、8個です。

「カテゴリー内の単語出現率 P(Wn|C)」を再度表にまとめました。

	サッカー	**野球**	**テニス**
ボール	$\frac{(3+1)}{(6+8)}$	$\frac{(1+1)}{(4+8)}$	$\frac{(1+1)}{(3+8)}$
スポーツ	$\frac{(2+1)}{(6+8)}$	$\frac{(1+1)}{(4+8)}$	$\frac{(0+1)}{(3+8)}$

分数を計算します。

	サッカー	**野球**	**テニス**
ボール	$\frac{4}{14}$	$\frac{2}{12}$	$\frac{2}{11}$

	サッカー	野球	テニス
スポーツ	$\frac{3}{14}$	$\frac{2}{12}$	$\frac{1}{11}$

こうすると、「スポーツ」という単語が「テニス」の学習データになくても、確率が0にはならないことがわかります。

そして、続きをそのまま計算します。

1.で計算した「カテゴリー出現率 P(C)」はそのままです。
「カテゴリー内の文書出現率 P(D|C)」は上記で分数計算した「ボール」と「スポーツ」の値を掛け合わせたものです。

	サッカー	野球	テニス
カテゴリー出現確率	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
カテゴリー内の文書出現率	$\frac{12}{196}$	$\frac{4}{144}$	$\frac{2}{121}$
文章内のカテゴリー出現率	$\frac{2}{4} \times \frac{12}{196}$	$\frac{1}{4} \times \frac{4}{144}$	$\frac{1}{4} \times \frac{2}{121}$
計算結果	$\frac{3}{98}$	$\frac{1}{144}$	$\frac{1}{242}$

結果はテニスの確率が0にならずに、サッカーと分類されました!!

しかし、加算スムージングは「カテゴリー出現率 P(C)」の値の影響が大きくなり、精度が悪くなるように思います。ここをよく考慮して学習データをセットする必要がありますね。

5. アンダーフロー対策

例題はわかりやすくするために、単語数やデータセットを少なくしましたが、実際はもっと沢山の単語を取り扱います。そのため、計算結果の分母が非常に大きな数になり、アンダーフローを起こす可能性が高いです。

その回避策が、**対数で比較する**方法です。

0.001 は $\frac{1}{10} \times \frac{1}{10} \times \frac{1}{10}$ とも表すことができ、10の-3乗といい、**-3**が0.001の対数です。(底は10)

0.0001 は $\frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10}$ とも表すことができ、10の-4乗といい、**-4**が0.001の対数です。(底は10)

X, Y, Z の大小関係は、Xの対数, Yの対数, Zの対数 の大小関係と同じです。(底が同じで1より大きい場合)
対数は元の値より大きくなります。(値が1より小さくて、底が1より大きい場合)

また、値の掛け算の大小は対数の足し算の大小と同じになります。
２×２×２　：　対数３
２×２×２**×２**　：　対数３**＋１**

話は例題に戻り、
「カテゴリー内の文書出現率 P(D|C)」を「ボール」と「スポーツ」の「カテゴリー内の単語出現率 P(Wn|C)」の対数の足し算として算出し、「カテゴリー出現確率 P(C)」の対数と足し算してあげます。

	サッカー	**野球**	**テニス**
カテゴリー出現確率	$\log \frac{2}{4}$	$\log \frac{1}{4}$	$\log \frac{1}{4}$
カテゴリー内の単語出現率(ボール)	$\log \frac{4}{14}$	$\log \frac{2}{12}$	$\log \frac{2}{11}$

	サッカー	野球	テニス
カテゴリー内の単語出現率(スポーツ)	$\log \frac{3}{14}$	$\log \frac{2}{12}$	$\log \frac{1}{11}$
文章内のカテゴリー出現率	$\log \frac{2}{4} + \log \frac{4}{14} + \log \frac{3}{14}$	$\log \frac{1}{4} + \log \frac{2}{12} + \log \frac{2}{12}$	$\log \frac{1}{4} + \log \frac{2}{11} + \log \frac{1}{11}$

これでナイーブベイズ分類器の完成です!!

※logの表記方法はこちらを見てください
http://kenyu.red/archives/3132.html

Pythonで書いてみた

以下のサイトを参考に実際にpythonで書いてみました。
http://yaju3d.hatenablog.jp/entry/2016/10/31/222307

上で説明した値がどこに該当するのか、出来る限りコメントで記載してみました。
形態素解析にはjanomeを使用してます。

naivebayes.py

```
class NaiveBayes:

    # コンストラクタ
    def __init__(self):

        # 学習データの全単語の集合(加算スムージング用)
        self.vocabularies = set()

        # 学習データのカテゴリー毎の単語セット用
        self.word_count = {}

        # 学習データのカテゴリー毎の文書数セット用
        self.category_count = {}

    # 学習
    def train(self, document, category):

        # 学習文書を形態素解析
        ma = MorphologicalAnalysis()
        word_list = ma.get_word_list(document)

        for word in word_list:

            # カテゴリー内の単語出現回数をUP
            self.__word_count_up(word, category)

        # カテゴリーの文書数をUP
        self.__category_count_up(category)

    # 学習データのカテゴリー内の単語出現回数をUP
    def __word_count_up(self, word, category):

        # 新カテゴリーなら追加
        self.word_count.setdefault(category, {})

        # カテゴリー内で新単語なら追加
        self.word_count[category].setdefault(word, 0)

        # カテゴリー内の単語出現回数をUP
        self.word_count[category][word] += 1

        # 学習データの全単語集合に加える(重複排除)
        self.vocabularies.add(word)

    # 学習データのカテゴリーの文書数をUP
```

```
    def __category_count_up(self, category):

        # 新カテゴリーなら追加
        self.category_count.setdefault(category, 0)

        # カテゴリーの文書数をUP
        self.category_count[category] += 1

    # 分類
    def classifier(self, document):

        # もっとも近いカテゴリ
        best_category = None

        # 最小整数値を設定
        max_prob = -sys.maxsize

        # 対象文書を形態素解析
        ma = MorphologicalAnalysis()
        word_list = ma.get_word_list(document)

        # カテゴリ毎に文書内のカテゴリー出現率P(C|D)を求める
        for category in self.category_count.keys():

            # 文書内のカテゴリー出現率P(C|D)を求める
            prob = self.__score(word_list, category)

            if prob > max_prob:
                max_prob = prob
                best_category = category

        return best_category

    # 文書内のカテゴリー出現率P(C|D)を計算
    def __score(self, word_list, category):

        # カテゴリー出現率P(C)を取得 (アンダーフロー対策で対数をとり、加算)
        score = math.log(self.__prior_prob(category))

        # カテゴリー内の単語出現率を文書内のすべての単語で求める
        for word in word_list:

            # カテゴリー内の単語出現率P(Wn|C)を計算 (アンダーフロー対策で対数をとり、加算)
            score += math.log(self.__word_prob(word, category))
```

```
        return score

    # カテゴリー出現率P(C)を計算
    def __prior_prob(self, category):

        # 学習データの対象カテゴリーの文書数　/ 学習データの文書数合計
        return float(self.category_count[category] / sum(self.category_count.value

    # カテゴリー内の単語出現率P(Wn|C)を計算
    def __word_prob(self, word, category):

        # 単語のカテゴリー内出現回数 + 1 / カテゴリー内単語数 + 学習データの全単語数 (加算スム
        prob = (self.__in_category(word, category) + 1.0) / (sum(self.word_count[c
                                                              + len(self.vocabulari
        return prob

    # 単語のカテゴリー内出現回数を返す
    def __in_category(self, word, category):

        if word in self.word_count[category]:
            # 単語のカテゴリー内出現回数
            return float(self.word_count[category][word])
        return 0.0
```

上記のクラスの学習と分類をViewから呼び出してあげました。

view.py

```
def matching(request):

    if request.method == 'POST':

        # 対象文書取得
        words = request.POST['words']

        nb = NaiveBayes()

        # 学習データセット
        nb.train('ボール スポーツ ワールドカップ ボール', 'サッカー')
        nb.train('ボール スポーツ グローブ バット', '野球')
        nb.train('ボール ラケット コート', 'テニス')
        nb.train('ボール スポーツ', 'サッカー')
```

```
        # 分類した結果を取得
        category = nb.classifier(words)

        dictionary = {'category': category}

        return render(request, 'matching.html', dictionary)

    elif request.method == 'GET':

        return render(request, 'matching.html')
```

おわり

機械学習は勉強を始めたばかりで、分からない事だらけです。
またPythonも始めて数日なので、書き方が変かもしれません。

上記の説明で誤りがありましたら、指摘いただけたら嬉しいです。

また、janomeだとカタカナ続きの単語を区切ってくれませんでした。
自分用メモのつもりでしたが、もしも誰かの役に立てたら嬉しいです😍

ishizakiiii
エンジニアです。機械学習を勉強して、アプリ作ってます。強化学習が好きです。