Web文書分類　最終レポート

情報学群情報科学類

201811319　永崎遼太

1. アルゴリズムの概要

1.1 k-近傍法

k-近傍法では、入力文書と教師文書の距離を求め、入力文書の近傍に位置する教師のクラス情報に従って、入力文書のクラスを決定する。k-近傍法のkとは、近傍に位置する教師文書のうち、最も近いk個の文書のクラス情報を考慮することを意味している。具体的には、入力文書から最も近くに位置するk個の教師文書のクラス情報の多数決によって入力文書のクラスを決定する。

入力文書からの近傍を求める一般的な方法は、文書間の類似度を考え、類似度が大きいほど距離が近い、すなわち近傍であると考える。

類似度としてコサイン類似度を採用する。類似度を計算するにあたって、ベクトル空間モデルを考え、教師文書、入力文書をそれぞれ文書ベクトルで表現する。ベクトル空間モデルは、文書をベクトルとして表現するモデルである。この時、ある2つの文書ベクトル x_1, x_2 に対する類似度 $sim(x_1, x_2)$ を各ベクトルが張る角度の余弦とし、以下の式で求めるものがコサイン類似度である：

$$sim(x_1, x_2) = cos\theta = \frac{x_1 \cdot x_2}{||x_1|| \, ||x_2||}$$

右辺の分子はベクトル間のない積である。分母は各ベクトルの大きさの積である。あるn次元ベクトルの内積値は、

$$x_1 \cdot x_2 = \sum_{i=1}^{n} x_{1i} \cdot x_{2i}$$

で求めることができる。

k-近傍法の手続きは次のようになる。

0：kの値を決める

1: 教師文書を読み込む

2: i番目の入力文書に対して、

2-1: 各教師文書との間で類似度を求める（類似度計算）

2-2: 最も類似度の高いk個の教師文書を求める（近傍文書の獲得）

2-3: k個の教師文書が持つクラス情報を多数決し分類クラスを決定する（意思決定）

1.2 ナイーブベイズ

ナイーブベイズ法では、文書xが与えられた時のクラス c_i の生起確率を求め、その値が最大となるクラスを出力する。実際には以下求める。

$$\hat{c} = \underset{c_i}{argmax}\ P(c_i|x)$$

これを、次のようにして式変換を行う。

$$\hat{c} = \underset{c_i}{argmax}\ P(c_i|x)$$

$$= \underset{c_i}{argmax}\ \frac{P(x|c_i)P(c_i)}{P(x)} \qquad \text{（ベイズの定理）}$$

$$= \underset{c_i}{argmax}\ P(x|c_i)P(c_i) \qquad \text{（分母を無視）}$$

$$\approx \underset{c_i}{argmax}\ P(c_i)\prod_{w_j \in V} P(w_j|c_i)^{N(w_j,x)} \text{（ナイーブベイズ仮定、単語の位置情報を無視）}$$

$$= \underset{c_i}{argmax}\ \prod_{w_j \in V} P(w_j|c_i)^{N(w_j,x)} \qquad (P(c_i|x)\text{を無視})$$

ここで、ゼロ頻度問題への対処として、実際に観測された頻度の値を補正するスムージングを行い、$P(c_i|x)$ は次のようになる

$$P(c_i|x) = \frac{N_{c_i}(w_j) + \delta}{\sum_{w_j}\{N_{c_i}(w_j) + \delta\}}$$

また、プログラムの実装を考えると分類クラスの計算では桁落ちの危険があるため、最終的に対数変換を行なった次の式を用いて実装する。

$$\underset{c_i}{argmax}\ \prod_{w_j \in V} P(w_j|c_i)^{N(w_j,x)}$$

$$= \underset{c_i}{argmax}\ log\{\prod_{w_j \in V} P(w_j|c_i)^{N(w_j,x)}\}$$

$$= \underset{c_i}{argmax}\ \sum_{w_j \in V} logP(w_j|c_i)^{N(w_j,x)}$$

$$= \underset{c_i}{argmax}\ \sum_{w_j \in V} N(w_j,x)\ logP(w_j|c_i)$$

ナイーブベイズ法では、あらかじめ単語の生起確率を求めておく訓練ステップと、求めた生起確率を利用して、入力文書のクラスを決定する分類ステップに分けて考えることができる。

訓練ステップ

教師文書を利用して、各クラス、各単語について $P(w_j|c_i)$ を求める

分類ステップ

入力文書 x に対して、分類クラスを決定する

1: 入力文書の各単語に対して $N(w_j, x)$ を求める

2: 入力文書の各単語に対して、訓練フェーズの結果を参照して、$P(w_j|c_i)$ を得る

3: $N(w_j, x)$ と $P(w_j|c_i)$ の情報から分類クラスを決定する

1.3 ロジスティック回帰

クラスc_iごとに関数 $g_i(x)$を用意し、

$$\hat{c} = \underset{c_i}{argmax}\ g_i(x)$$

のようにクラスを決定する方法を識別関数法という。ロジスティック回帰では、識別関数として以下のロジスティックシグモイド関数を用いる。

$$\sigma = \frac{1}{1 + exp(-a)}$$

ロジスティックシグモイド関数は条件付き確率 $P(c_+|x)$ を表している。

$$P(c_+|x) = \sigma(w \cdot x) = \frac{1}{1 + exp(-w \cdot x)} = g_+(x)\ ,\ P(c_-|x) = 1 - P(c_+|x)$$

ロジスティック回帰では教師データを利用して重みベクトルwを学習することが分類器の構築に相当する。クラスが2つの場合、クラスの確率はベルヌーイ分布に従う。

$$\prod_{t=1}^{|X|} P(c_+|x_t)^{y_{+,t}}(1 - P(c_+|x_t))^{1-y_{+,t}}$$

クラスがc_+であれば$y_{+,t} = 1$、そうでなければ$y_{+,t} = 0$である。

上記はデータの尤度と呼ばれる値であり、ロジスティック回帰では、負の対数尤度が教師データ全体の損失の総和となり、これは交差エントロピー誤差として知られている。重みwの学習を「教師データ全体における損失の総和が最小になるようなwを選ぶ」問題としてみなすと、これは評価関数最適化の枠組みで解ける。評価関数最適化では、評価関数に対して、確率的勾配降下法に基づく以下の式に基づいて繰り返し計算をし、最適なwを獲得する。

$$w' = w_i - \rho \frac{\partial L_t}{\partial w_i}$$

交差エントロピー誤差の偏微分は

$$\frac{\partial L_t}{\partial w} = \frac{\partial L_t}{\partial P(c_+ | x_t)} \frac{\partial P(c_+ | x_t)}{\partial w \cdot x_t} \frac{\partial w \cdot x_t}{\partial w}$$

$$= (P(c_+ | x_t) - y_{+,t}) x_t$$

となり、以下の更新式が得られる。

$$w' = w - \rho (P(c_+ | x_t) - y_{+,t}) x_t$$

$$= w - \rho (\sigma(w \cdot x_t) - y_{+,t}) x_t$$

さらに、過学習を起きにくくする正則化を考慮して、正則化項付きロジスティック回帰の更新式は次のようになる。

$$w' = (1 - \lambda\rho) w - \rho (\sigma(w \cdot x_t) - y_{+,t}) x_t$$

この式を、以下の手続きで利用する。

1: 重みベクトルwを適当に初期化する

2: 各教師データについて以下を行う(foreach $x_t \in X$)

　　正則化項付きロジスティック回帰の更新式を使って重みを更新

3: 重みに変化がなければ終了。そうでない場合は手続き2に戻る。

2. 必須課題・optional課題の結果、工夫点について

2.1 k-近傍法

　必須課題として、近傍数を5としてプログラムを実行した出力の結果が、out.vec=tfidf.k=5.txtと一致することが確認できた。

Optional課題として、k-近傍法の近傍数を変えて分類結果を評価する。性能評価は、eval.pdfに基づいて実行する。実際に、出力結果を計測プログラムを使って評価すると次のようになる（k=5の場合）。

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

> python3 eval.py -a ans.txt -b out.vec=tfidf.k=5.txt

0.83

cc:38 cm:12 mc:5 mm:45

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

これは、正解率が83%であることを表している。

kの値を1-100に変えて行なった実験結果の一部を以下の表で示す。

k-近傍法（tfidf, k=5）	83
k-近傍法（tfidf, k=7）	83
k-近傍法（tfidf, k=9)	81
k-近傍法（tfidf, k=13)	85
k-近傍法（tfidf, k=17)	87
k-近傍法（tfidf, k=25)	89
k-近傍法（tfidf, k=29)	90
k-近傍法（tfidf, k=33)	91
k-近傍法（tfidf, k=79)	91
k-近傍法（tfidf, k=101)	88

この結果より、近傍数を多くして調べてみると、k=33, 79の時に最大性能(91)が出た。従って、今回の課題データに最適な近傍数は33または79であることがわかった。

作成したプログラムで工夫した点は、オプション入力を受け付け、kを設定するようにした点である。

2.2 ナイーブベイズ

　必須課題として、ナイーブベイズ法に基づく文書分類プログラムを作成した。訓練ステップを実現するプログラムnb_train.py、分類ステップを実現するプログラムnb_classification.pyを実装した。

スムージングの値を1として実行して得られた出力を評価した結果は次のようになった。

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

python3 eval.py -a ans.txt -b output.txt

0.96

cc:47 cm:3 mc:1 mm:49

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

これより、正解率が96%であることがわかる。

　Optional課題として、スムージングを行わない場合の動作を同様に確認すると、正解率は95%となり、性能が低下したことがわかった。また、スムージングの値を50とした時に、正解率が97%となり、性能が向上したことが確かめられた。また、多変数ベルヌーイモデルを実装し、同様にして性能を評価した結果、正解率は96%であった。

　プログラムの工夫点として、スムージングの値を変数として定義して計算式を実装し、容易に値を変えて実験ができるようにした。従って、スムージングの値を0とすることで、スムージングを行わない場合の動作を確認することが容易にできた。

3.3 ロジスティック回帰

　正則化項付きロジスティック回帰に基づく文書分類プログラムを作成した。訓練ステップを実現するプログラムlr_train.py、分類ステップを実現するプログラムlr_classification.pyを実装した。rho = 0.001, lambda = 1, 重みベクトルは1で初期化した。また、最大の繰り返し回数を500として設定した。ロジスティック回帰による分類結果の性能は、交差検定法を用いて評価した。交差検定法の分割数は5とし、評価指標は正解率を用いた。実際に、交差検定法により評価した結果は次のようになった。

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

% python3 corpus_lr.py

Results

1 : 0.9055555555555556

2 : 0.9035714285714286

3 : 0.8916666666666666

4 : 0.9174603174603174

5 : 0.9138613861386138

Average

0.9064230708785164

ーーーーーーーーーーーーーーーーーーーーーーーーーーーー

このように、5回の分割学習法の結果をそれぞれ出力するようにし、最後に平均値を出力するように工夫した。

それぞれのステップで必要な機能を別のファイルにまとめて定義し、プログラムの冗長性を改善した。具体的には、

- tf法による特徴抽出結果を出力したファイルの内容を辞書データとして読み込む関数
- 辞書データ内の一部のデータを抽出する関数
- 辞書データをn分割する
- 辞書データのキーをnだけずらす
- 2つの辞書データを結合する

といった関数を実装した。また、交差検定法をfor文によって自動的に実行できるようにプログラムを作成した。

3. 文書分類を行なってみた全体的な感想

　かねてよりデータサイエンスに関する研究をしたい、将来的にはデータサイエンティストとしてキャリアを積んでいきたいと思っており、特にテータマイニング、テキストマイニングの分野には関心があったので、本授業の文書分類の実験を通し、自然言語処理のアルゴリズムや実装方法を学ぶことができ非常に有意義であった。kaggleといったプラットフォーム上でデータサイエンスに関しての学習をすることはできるが、それ以外の方法ではたいていirisデータやmninstデータぐらいでしかチュートリアルがないため、本授業の実験を通して、実際のレビュー文書などのテキストデータを扱うことができたのはとても貴重であった。pythonではさまざまな複雑なアルゴリズムを簡単に実行できるライブラリが用意されており、実務上はそれらを利用すれば問題はないと思うが、本実験のようにアルゴリズムを1から実装するというのは、アルゴリズムの深い理解と実装力の向上につながったため、とても良い経験となった。

4. 来年度受講生へのアドバイス

　基本的には全て自分で実験を進めていく必要があるため、まずは資料をしっかりと読んでアルゴリズムを理解してから実験を始めると良い。コーディングで詰まったらインターネット上で参考記事を探し、どうしてもわからないことがあればTAに聞くと良い。