

Mining the Web : Chapter 5 (SUPERVISED LEARNING)

高村大也 (Hiroya Takamura)
takamura@pi.titech.ac.jp

平成 15 年 6 月 26 日

Internet の普及により、ディレクトリ型検索エンジンなどにおいて、サイトの内容を示すタグ付け技術 (topic tagging) が必要とされてきている。

News tracking も、文書の内容 (トピック) を見つける技術の必要性を表す良い例である (News tracking とは、同一の事件を扱ったニュースを同定するタスクである)。その他の例としては、E-mail 分類や、bookmark の分類などがある。

文書にトピックのラベルを付与するタスク (文書分類) は、supervised learning (教師有学習) の代表的な応用例であるといえる。文書分類は、著者推定などにも使われている。その場合は、各作品の writing style を学習することになる。また、別の応用としては、hyperlink の目的を分類するのにも使われる。たとえば、詳細説明、資料、批評など様々なページへのリンクが考えられる。

5.1 The Supervised Learning Scenario

図書館学者 (あるいは司書) は、(人手で) トピック・ラベルを正しく付与する訓練を受けている。コンピュータに同じことができるだろうか? このようなタスクのうち、特に、すでにクラスが与えられている事例を参考にして、対象となる事例にクラスを割り当てるようなタイプの学習を、classification、supervised learning という。これが本章で扱う内容である。

Supervised learning では、まず、learner (学習器、あるいは classifier (分類器)) が、訓練データを入力として受け取る。訓練データとは、クラスがすでにわかっている事例の集合である。この訓練データを用いて、分類器は訓練されるのである。訓練が終ると、クラスがわかっていないテスト (評価) 事例が与えられ、そのクラスを推定する。

ここでは文書分類を扱うが、事例空間の次元は非常に大きくなる。ベクター・スペース・モデルを採用した場合は、各単語 (の生起) が素性となり、単語の種類数が次元数になる。次元が高過ぎて、素性の同時確率を求めることはほとんど不可能である。よってなんらかの工夫が必要となる。

本章で扱う文書分類の手法は、後の章で出てくる hypertext classification などの基礎となるものである。

5.2 Overview of Classification Strategies

ここでは、文書分類のいくつかの手法に簡単に触れ、それぞれの長所と短所について述べる。

ベクター・スペース・モデルを用いた場合、nearest-neighbor(NN) 分類器が簡単に導入できる。NN 分類器とは、テスト事例に対し、それと最も類似している訓練事例を見つけ、その訓練事例の持つクラスをテスト事例に付与するアルゴリズムである。シンプルだが、classify に結構時間がかかる。Feature selection が有効に働く。

次にベイズ分類器について述べる。 $Pr(d|c)$ (d : 文書、 c : クラス) を使う。

また、文書が与えられたときのクラスの確率 $Pr(c|d)$ を、推定するタイプの方法もある。例えば、最大エントロピー分類器がある。

hypertext への応用の際は、異なるタイプの情報を一つにまとめる必要が出てくることもある（たとえば、テキスト部分に出現する単語、タイトルの単語、ヘッダーのテキスト、HTML タグの構造、リンク先の単語など）。そのような多種多様な素性に対して、規則推論（rule induction）などのアプローチがある。

文書分類においては、tokenization や feature extraction を注意深く行う必要がある。POS タグの情報などが精度を向上させると言われている。あるいは、単語の意味の曖昧性を除去することも効果があると言われている。また、表現の多様性を吸収することも大事である（mild steel = M.S. = M/S）。このように、素性の選び方は非常に重要で、それ自体が立派な研究対象と見なされる。

5.3 Evaluating Text Classifiers

分類システムを評価するための基準としては以下のように様々なものがある。

- 精度。
- スピード。訓練、分類の両方において。
- シンプルさ。Scalability（巨大データに対応できるか否か）。
- 可視性。結果の解釈が容易にできるか。人間の判断を分類器にフィードバックできるか。

ここでは精度に注目して話を進める。

5.3.1 Benchmarks

文書分類の実験のためのデータセットとして、public なものには以下のようなものがある。

- Reuters：英語の新聞記事。全部で約 10700 記事。135 カテゴリ。約一割の文書が複数のラベルを持つ。
- OHSUMED：医学関係の論文のアブストラクト。348566 記事。IR によく使われる。
- 20NG：ニュースグループへの投稿文章。約 18800 記事。20 カテゴリ。カテゴリは階層構造を持つ。
- WeKB：大学関連のウェブ・ページ。約 8300 記事。7 カテゴリ。
- Industry：産業関連のウェブ・ページ。約 10000 記事。105 カテゴリ（業種で分けられている）。カテゴリは浅い階層構造を持つ。

5.3.2 Measures of Accuracy

まず、文書分類においては、以下の二つ状況が想定できる：

- 各文書はただ 1 つクラスに関連付けられる（分類される）。
- 各文書はいくつかのクラスに関連付けられる（分類される）。

第一のケースでは、confusion matrix M が評価に使用される。confusion matrix では、要素 $M[i, j]$ は、本当の答えがクラス i で、分類器はクラス j を予測したような評価事例の数を表す。つまり、 $\sum_i M[i, i]$ は正解数であり、confusion matrix の対角成分が残りに比べて大きいとき、分類がうまくいったといえる。しかし、

Confusion matrix の例

		j		
		1	2	3
i	1	10	0	1
	2	0	8	1
	3	2	0	15

2 × 2 Confusion matrix の例

	class(predicted)	not-class(predicted)
class (true)	3	1
not-class (true)	2	12

confusion matrix が巨大な場合は、ひと目で良いかどうか判断できない（しかも、数値として現れないので比較しにくい）。そういう場合は、

$$\frac{\sum_i M[i, i]}{\sum_{i, j} M[i, j]}$$

が使用される。当然ながら、この値が大きい方が良い。

第二のケースでは、問題を二値分類問題に reduce して考える。つまり、各クラスに関して、そのクラスかそうでないかという問題を考えるのである（例えば、“sports” か、“not-sports” か）。複数のラベルを持っている事例は、そのラベルの中に“sports”が含まれていれば、正例となる。このような方法は、two-way ensemble または one-vs-rest 法と呼ばれる。

このように二値分類問題に reduce したとして、では、どのような評価基準を考えればよいか？ confusion matrix を作ると、この場合は、2 × 2 行列ができる (5.3.2)。さきほどのように、 $\sum_i M[i, i] / \sum_{i, j} M[i, j]$ を使おうと思えば使えるが、今回の場合これはあまり良い評価基準にならない。それは、一般に、“class” に対して “not-class” が非常に大きくなり、全て “not-class” と答えるようにならない分類器が大きな評価値を獲得してしまうからである。これを回避するために、まず、recall と precision という二つの評価値を導入する。2 × 2 confusion matrix を使って説明する¹。

recall とは、“class” と判定されるべき事例がどれだけもれなく “class” と判定されたかを表す：

$$recall = \frac{M[0, 0]}{M[0, 0] + M[0, 1]}$$

一方、precision とは、分類器によって “class” と判定された事例のうちどれだけが本当に “class” なのかを表す：

$$precision = \frac{M[0, 0]}{M[0, 0] + M[1, 0]}$$

recall と precision は、各クラスに関して定義されることに注意されたい。

さて、この recall と precision の両方が高い分類器があれば、もう言うことはない。しかし、一般にこの二つは trade-off の関係にある（recall を上げるためには、少しでも “class” っぽい事例は、“class” と判定するのがよいが、そうすると precision は大抵下がる）。よって、この二つをうまく融合した評価値が必要なのだが、その話に行く前に、「平均」について考える。

¹本文では、クラスと文書の各ペアに関して一つの confusion matrix を導入しているが、わかりにくいと思われるので、ここではそのような matrix は導入しない

前述のように、recall と precision は各クラスに関して定義される。よって、全体の良さを測るためには、クラスの数だけ算出された recall と precision の平均をとる必要がある。単純には、全ての recall(precision) を足し合わせ、それをクラス数で割ればよいが、これは実は macroaverage と呼ばれる平均の出し方である：

$$\begin{aligned} \text{macroaveraged recall} &= \frac{\sum_c \text{recall}_c}{\text{number of classes}}. \\ \text{macroaveraged precision} &= \frac{\sum_c \text{precision}_c}{\text{number of classes}}. \end{aligned} \quad (5.1)$$

もう一つの平均は、microaverage と呼ばれ、次のように算出される：

$$\text{microaveraged precision} = \frac{\sum_c M_c[0, 0]}{\sum_c M_c[0, 0] + \sum_c M_c[1, 0]}.$$

つまり、macroaverage では、各クラスの recall(precision) が計算されてから、単純平均が計算されるのに対し、microaverage ではクラスを越えた全体の confusion matrix から直接 recall(precision) が計算される。よって、microaverage は、サイズの大きなクラスに影響を受けやすい。

さて、recall と precision を組み合わせた指標について説明する。ここでは、break-even point と F-measure について説明する。

まず、break-even point。多くの分類器では、パラメータをチューニングすることにより、recall と precision の trade-off の割合を変えることができる。例えば、「あるクラスらしさ」を表すスコアを算出するような分類器ならば、スコアに対する閾値を変えればよい (specifically, 閾値を上げれば普通 precision が高くなり、recall が下がる。vice-versa)。break-even point とは、recall と precision が等しくなったときの数値のことである。

つぎに F-measure (F_1 スコア)。これは、次のように定義される：

$$F_1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}.$$

これは、recall と precision の調和平均となっており、他の文献ではさらに一般的に定義されることもしばしばである。²

5.4 Nearest Neighbor Learners

似ている文書には同じクラスに属しているだろう、という直感を直接的に形にしたのが Nearest Neighbor (NN) 分類器である。その直感の定式化には、ベクター・スペース・モデルと、cosine 類似度を用いる。

以下、NN の説明をしていく。訓練時には、ほとんど何もしない (実際には、各文書に関し、出現した単語ヘインデックスを張り、またその文書のクラスを記憶しておく)。分類時には、評価事例 d_q に対し、(例えば cosine 類似度で) 最も近い訓練事例を見つける。その訓練事例のクラスラベルを評価事例に付与する。

しかしこのような単純な方法だと、noisy な事例に引きずられて、分類精度がよくなることが多い。よって、次のような拡張がなされる。訓練時は同じく何もしない。分類時に、評価事例 d_q に対し、最も近い方から k 個の訓練事例を見つける。その k 個のクラスラベルを見て、最も多いラベルを評価事例に付与する。このように拡張すると、たまたま例えば 1 個 noisy な訓練事例があっても、結果はそれに引きずられない。³

²より一般的な定義は、

$$F_\beta = \frac{1}{\beta \cdot 1/\text{precision} + (1 - \beta) \cdot 1/\text{recall}}$$

である。 F_1 と本文に書かれているが、 $F_{0.5}$ と書く方が普通な気がする。

³この k により、この手法はよく k -NN と呼ばれる。ヨーロッパの方で、memory-based learner と呼ばれている手法があるが、 k -NN と一緒。

NN はさらに拡張される。 k 個の訓練事例をとってきたあと、単純に最も多いラベルを選ぶのではなく、重み $s(d_q, d)$ を付けて足し合わせることもできる。また、各クラスに対し off-set b_c を使うこともできる (つまりクラスごとに異なる下駄を履かせる)。これらをまとめると、NN は評価事例に対し各クラス c のスコアを次式で与えることになる :

$$\text{score}(c, d_q) = b_c + \sum_{d \in k \text{ nearest examples}} s(d_q, d) \quad (5.2)$$

NN は、lazy learners の一種である (lazy learner とは、分類時に汎化を行うようなタイプの学習器である)。NN のパラメータは、cross-validation などによって求めるのが普通である。clustering を用いて k を求める方法もある。

5.4.1 Pros and Cons (長所と短所)

NN の長所

- シンプルでしかも昔からあるので、フリーのパッケージとかがたくさんある。
- 訓練時は indexing しかしないので、訓練事例の追加が簡単。
- パラメータを適切に tuning すれば、state-of-the-art な性能を発揮する。

NN の短所

- 評価事例に対し、その評価事例に含まれる単語を一つでも含む訓練事例は全て類似度計算し、ソーティングしなくてはならない。時間がかかる。
- 訓練事例をそのまま保存しているため、メモリもたくさん必要。

この短所を補うために、クラスタリングを用いた手法などが開発されている。つまり、類似した訓練文書をクラスタにまとめあげることにより、評価事例と、各訓練文書との類似度でなく、クラスタとの類似度を測ればよいことになる。

5.4.2 Is TFIDF Appropriate?

ここでは、TFIDF が本当に良い indexing といえるかどうかを考える。サイズが同じクラスが 2 つある場合を想定する。単語 t_1 は、どちらのクラスでも同じように出現し、全体では 10% の文書に出現する。単語 t_2 は、片方のクラスでしか出現しないが、全体では 20% の文書に出現する。つまり、 t_2 は、分類に大きく貢献する可能性を持っているにも関わらず、役に立ちそうにない t_1 より低い重みを付与されてしまう。⁴

5.5 Feature Selection

素性の重要性を推定する方法について考える。

訓練事例数が、素性数に比較して充分大きいとき、素性に関する分布などがある程度正確に推定できる。が、普通は事例数はそれほど大きくない。特に素性の組み合わせに対する同時分布を推定しようとするとき、その推定は破滅的である。

⁴IR の場合との差異に注意しなくてはならない。

同時分布をやめて、各素性の周辺分布を求めるならば、ずっとましである。が、それでもまだ大変である。ある単語がクラス 1 で出現しやすいのか、クラス 2 で出現しやすいのかを決めたいとき、もしその単語の全出現数自体が小さかったら、信頼できる判断を下せない。つまり、たまたま訓練データにおいて偏った出現の仕方をしたという可能性を捨て切れないのである。

よって、信頼できる判断を下せないそういった素性は扱わない方が良いかもしれない。このように、使える素性と使えない素性を分けることを feature selection という。単純には、全体の頻度が非常に小さい素性を削除するという方法が考えられる。また、a, an, the のように分類に貢献しないと思われる、いわゆる stop-words の削除も行われる。素性の削除はデータのコンパクト化という利点もある。しかし、低頻度語を削除するだけでなく、もう少し賢い方法を考えてみよう。

5.5.1 Greedy Inclusion Algorithm

feature selection の最も簡単なアルゴリズムの概要は：

1. 各素性について、クラス分類への貢献度の尺度を計算、
2. その尺度順に素性を並べ替える、
3. 上の方だけ使う。

大抵、尺度計算は、他の素性と独立して行われる。つまり、素性 A が使用されているとき素性 B はほとんど新しい情報を与えないかもしれないが、そういうことは考慮されないことが多い⁵。

以下に尺度をいくつか紹介する。用いる尺度によって、素性のランキングは多少変わるが、オーバーラップも大きく、それほど精度に影響しない。

The χ^2 test

これは、クラスラベルと素性の間に相関があるか否かを調べる統計的な尺度である。簡単のため、クラス数が 2 の場合で説明する。さらにここでは、binary document model (see Section 4.4.1) を使用する。

t がある単語 (素性) を表すとし、クラスラベルは 0 か 1 をとるものとする。さらに、

$$\begin{aligned} k_{i,0} &= \text{クラス } i \text{ に属するもののうち、} t \text{ を含まない文書数} \\ k_{i,1} &= \text{クラス } i \text{ に属するもののうち、} t \text{ を含む文書数} \end{aligned}$$

とする。次のような 2×2 contingency (偶然性) matrix を考える。

		I_t	
		0	1
C	0	k_{00}	k_{01}
	1	k_{10}	k_{11}

ここで、 C はクラスを表す二値確率変数、 I_t は単語を含むか否かを表す二値確率変数。 k_{lm} は、 $C = l, I_t = m$ なる事象の生起回数。 $n = k_{00} + k_{01} + k_{10} + k_{11}$ とすると、周辺分布は以下のようになる：

$$\begin{aligned} Pr(C = 0) &= (k_{00} + k_{01})/n, \\ Pr(C = 1) &= (k_{10} + k_{11})/n, \\ Pr(I_t = 0) &= (k_{00} + k_{10})/n, \\ Pr(I_t = 1) &= (k_{01} + k_{11})/n. \end{aligned}$$

⁵逆に、素性 A と素性 B が同時に出現することに意味があるかもしれないが、これも考慮されないことが多い。

もし、 C と I_t が独立なら、 $Pr(C = 0, I_t = 0) = Pr(C = 0) \cdot Pr(I_t = 0)$ となる。観測値から推定すると、 $Pr(C = 0, I_t = 0)$ は、 k_{00}/n となる。 $C = 0, I_t = 0$ 以外の場合についても同じことがいえる。そこで、この独立性の指標とされる χ^2 統計量を使う。 χ^2 統計量は、観測値の理論値からのズレを表す：⁶

$$\chi^2 = \sum_{l,m} \frac{(k_{lm} - nPr(C=l)Pr(I_t=m))^2}{nPr(C=l)Pr(I_t=m)}.$$

ただし、理論値といっても、周辺分布を信じた上での理論値である。よって、上の式は、次のように書ける：

$$\chi^2 = \frac{n(k_{11}k_{00} - k_{10}k_{01})^2}{(k_{11} + k_{10})(k_{01} + k_{00})(k_{11} + k_{01})(k_{10} + k_{00})}. \quad (5.3)$$

χ^2 統計量が小さいほど独立性が高く、 χ^2 統計量が大きいほど、独立性が低い。

今我々は、クラスと素性間の独立性を調べている。独立でない素性はクラスに関するなんらかの情報を与えるので、 χ^2 統計量が大きいものが望ましい素性である。

Mutual Information

この指標は、Multinomial document model (see Section 4.4.1) に対して useful である (なぜか、は不明)。

X と Y を離散確率変数とする。すると Mutual Information (MI, 相互情報量) は、

$$MI(X, Y) = \sum_x \sum_y Pr(x, y) \log \frac{Pr(x, y)}{Pr(x)Pr(y)} \quad (5.4)$$

と定義される。

MI は二つの確率変数の依存度合を測る。もし二つが独立なら、 $\forall x, y \frac{Pr(x, y)}{Pr(x)Pr(y)}$ となり、MI は 0 になる。MI が大きくなると、二つの確率変数は依存していることになる。

MI は依存度合を測るといったが、一体具体的には何を測っているのか、もう少し詳しく見てみる。解釈の方法として二つここで紹介する。

一つは、 Y の値を知った時の X のエントロピーの減少である (Y と X を逆にしても同じ)。 X のエントロピーは $H(X) = -\sum_x Pr(x) \log Pr(x)$ と定義される。条件付エントロピーは、

$$\begin{aligned} H(X|Y) &= -\sum_{x,y} Pr(x, y) \log Pr(x|y) \\ &= -\sum_{x,y} Pr(x, y) \log \frac{Pr(x, y)}{Pr(y)} \end{aligned}$$

と定義される。⁷

エントロピーと MI の間には以下の関係がある：

$$\begin{aligned} MI(X, Y) &= \sum_{x,y} Pr(x, y) \log \frac{Pr(x, y)}{Pr(x)Pr(y)} \\ &= \sum_{x,y} Pr(x, y) \left(\log \frac{Pr(x, y)}{Pr(y)} - \log Pr(x) \right) \\ &= -H(X|Y) - \sum_{x,y} Pr(x, y) \log Pr(x) \\ &= -H(X|Y) - \sum_x Pr(x) \log Pr(x) \\ &= -H(X|Y) + H(X). \end{aligned} \quad (5.5)$$

⁶式の導出はここでは行わないが、多分、「多項分布を正規分布で近似」し、「正規分布の標本分散と分散の比が χ^2 分布に従うことを使えば導ける(と思う)。興味のある人は、統計や検定の本で調べて下さい。

⁷本文には、 $H(X|Y) = -\sum_{x,y} \log Pr(x, y)/Pr(y)$ と記されているが、間違いである。

つまり、 Y を教えることによって、曖昧性（エントロピー）がどれくらい減ったかを示している。

もう一つの解釈は、KL divergence である⁸。KL divergence とは、次のように定義され、二つの確率分布 Θ_1 と Θ_2 の距離（実際は擬距離）を与える：

$$KL(\Theta_1||\Theta_2) = \sum_z Pr_{\Theta_1}(z) \log \frac{Pr_{\Theta_1}(z)}{Pr_{\Theta_2}(z)}.$$

ここで $Z = (X, Y)$ とすると、MI は、実際の観測値から推定された分布 Θ_{true} と、 X と Y が独立だとしたときの分布 $\Theta_{independent}$ の KL divergence を表している：

$$\begin{aligned} KL(\Theta_{true}||\Theta_{independent}) &= \sum_{x,y} Pr(x,y) \log \frac{Pr(x,y)}{Pr(x)Pr(y)} \\ &= MI(X,Y) \end{aligned} \quad (5.6)$$

MI が大きいほど KL divergence の意味で Θ_{true} と $\Theta_{independent}$ は離れており、つまり、 X と Y は依存していることになる。

Feature Selection に適用するには、単に X を素性（単語）、 Y をクラスと見なせばよい。しかし、文書がどのようにモデル化されているかによって、具体的な計算式は当然異なるので注意しなくてはならない。 I_t を素性に関連付けられた確率変数とすると、binary model では、 $i_t \in \{0, 1\}$ だが、multinomial model では、 i_t は非負の整数全体をとる。

$$MI(I_t, C) = \sum_{l,m \in \{0,1\}} \frac{k_{l,m}}{n} \log \frac{k_{l,m}/n}{(k_{l,0} + k_{l,1})(k_{0,m} + k_{1,m})/n^2} \quad (5.7)$$

MI の考えられる問題点は、文書の長さがモデルに入っていないことである。ある単語が、二つのクラスで同程度に出現する（例えば、10000 語に 5 回）とする。しかし、二つのクラスでは平均的な文書の長さが異なるとする。この場合、この単語は分類に貢献できると考えられる。しかし、MI ではこの単語を捉えることはできない。

Fisher's discrimination index

この指標は、文書の長さが正規化されているときに使われる。簡単のため、2 クラスの場合を考える。 X, Y を、それぞれのクラスに属する文書ベクトルの集合とする。ベクトルの要素は、正規化された頻度でも、(TFIDF などの) term-weighting が施されたものでよい。

それぞれのクラスの平均ベクトルを、 $\mu_X = (\sum_{x \in X} x)/|X|$ 、 $\mu_Y = (\sum_{y \in Y} y)/|Y|$ 、とおく。これらのベクトルは列（縦）ベクトルであるとしよう。それぞれの共分散行列は、 $S_X = (1/|X|) \sum_X (x - \mu_X)(x - \mu_X)^T$ 、 $S_Y = (1/|Y|) \sum_Y (y - \mu_Y)(y - \mu_Y)^T$ と表される。

Fisher の判別では、以下の二つの量の比が最大になるような列（縦）ベクトル $\alpha \in R^m$ を見つける：

- 両平均ベクトルを α へ射影したときの、両者の差の自乗： $(\alpha^T(\mu_X - \mu_Y))^2$,
- 平均射影分散： $(1/2)\alpha^T(S_X + S_Y)\alpha$.

つまり、

$$\begin{aligned} \alpha^* &= \operatorname{argmax}_{\alpha} J(\alpha) \\ &= \operatorname{argmax}_{\alpha} \frac{(\alpha^T(\mu_X - \mu_Y))^2}{\alpha^T(S_X + S_Y)\alpha} \end{aligned} \quad (5.8)$$

⁸KL divergence と KL distance は同じ。

なる方向ベクトル α を求めることになる。⁹

$S = (S_X + S_Y)/2$ とすると、式 (5.8) は、 $\alpha = S^{-1}(\mu_X - \mu_Y)$ のとき極値となる (もし S^{-1} が存在すればである)。¹⁰

この Fisher 判別は、扱う空間の次元が数百オーダーである、信号解析でよく用いられている。しかし、NLP のように、次元が数万オーダーになると計算量の点において現実的でない。さらに、このような線形変換はスパースなデータをそうでなくしてしまい、その後の処理が重くなる。しかし、ここでの目的は、いらぬ素性を省くことであるので、少し話が違ふ。

最適な方向 α を探す代わりに、各素性が候補となる方向 α_t を与えていると考えてみよう。当然この α_t は、対応する軸方向になる。つまり、 $\alpha_t = (0, \dots, 1, \dots, 0)^T$ 。Fisher インデックスを次のように定義する：

$$\begin{aligned} FI(t) &= J(\alpha_t) \\ &= \frac{(\alpha_t^T(\mu_X - \mu_Y))^2}{\alpha_t^T S \alpha_t}. \end{aligned} \quad (5.9)$$

$\mu_{X,t}$ を、 μ_X の第 t 成分とすると、 $\alpha_t^T \mu_X = \mu_{X,t}$ 。これを使うと、

$$FI(t) = \frac{(\mu_{X,t} - \mu_{Y,t})^2}{(1/|X|) \sum_X (x_t - \mu_{X,t})^2 + (1/|Y|) \sum_Y (y_t - \mu_{Y,t})^2} \quad (5.10)$$

と表され、これを素性 t の良さの指標として使用できる。今はクラス数 2 の場合について考えたが、クラス数が増えても以下のような形で簡単に表される：

$$FI(t) = \frac{\sum_{c_1, c_2} (\mu_{X,t} - \mu_{Y,t})^2}{\sum_c (1/|D_c|) \sum_{d \in D_c} (x_{d,t} - \mu_{c,t})^2}, \quad (5.11)$$

ここで D_c はクラス c の訓練事例集合。

Validation

ここまで、素性の「良さ」を測るいくつかの指標を見てきた。しかし、指標を与えるだけではダメで、それを使ってどのように feature selection をするかという問題がある。ここではその問題を解くことを考える。

指標によって、素性はランク付けされているので、そのランキングの上からどこまで使用するかという閾値が決まればよい。もっとも単純には、与えられた訓練データを二つに分け、片方を用いて素性をランキングする。いろいろな閾値を設けて、もう片方 (held-out データと呼ばれる) を分類し、精度がもっとも良い閾値を選ばばよい。この方法は validation と呼ばれるらしい¹¹。データを複数に分け、held-out データの役割を交替して順番に行わせるのを cross-validation という (たとえば、A,B,C の 3 つ分けたら、A+B を訓練に用い C

⁹ 一体これは何をしているのか。分子に関しては明らかで、

$$(\alpha^T(\mu_X - \mu_Y))^2 = (\alpha^T \mu_X - \alpha^T \mu_Y)^2.$$

つまり、二つの平均の射影先の差。最大化するということは平均の射影先が離れていた方がよいということ。分母に関しては、

$$\begin{aligned} \alpha^T S_X \alpha &= (1/|X|) \sum_X \alpha^T (x - \mu_X)(x - \mu_X)^T \alpha \\ &= (1/|X|) \sum_X (\alpha^T (x - \mu_X))^2 \\ &= (1/|X|) \sum_X (\alpha^T x - \alpha^T \mu_X)^2, \end{aligned}$$

であることを用いると、分母は α 方向への二つの分散を足していることになる。最大化なので、これが小さい方がよいということ。平均が離れているだけでなく、その平均に事例が集中しているとよりよいという直観と合致する。

¹⁰ 式 (5.8) は、数物理学ではよく知られた一般化レイリー商 (generalized Rayleigh quotient) と呼ばれる形になっていることからわかるらしい。が、そこまではフォローしてない。

¹¹ held-out 法という呼び名の方が知られていると思う

で評価、A+Cで訓練してBで評価、B+Cで訓練してCで評価と三回行う。ここでは三回行ったので、3-fold cross-validation という)¹²。leave-one-out 法は、cross-validation の極端な場合に相当し、訓練データ全体 D 中の各事例 d について、 $D - \{d\}$ で訓練し、 d で評価する。つまり、訓練事例数だけ実験を繰り返すことになる。SVM のような重い手法では大変。

精度がもっとも良い閾値を選ぶには、例えばランキング順に素性を足していけばよい。

例として、ベイズ分類器と Fisher 判別での feature selection を用いた文書分類の結果を示す (図 5.5)。元の素性数は、約 20000 であったが、そのうちたった 140 を用いた場合が最もよく、当然計算スピードも格段に上がっているはずである。

5.5.2 Truncation Algorithm

素性を加えていく前節の手法と異なり、ここで説明するアルゴリズムは、素性を削っていく。全素性 T から始まり、不要と思われる素性を落していくことにより、素性のある部分集合 $F \subseteq T$ を決定するのである。 F と T に間の望ましい性質とはどんなものか？

ほとんどの確率的分類器では、なんらかの形で、事例が与えられたときのクラスの確率分布 $Pr(C|T)$ を算出する¹³。素性を削ると、クラスの確率分布は、 $Pr(C|F)$ になる。よって、ここでは $Pr(C|F)$ がなるべく $Pr(C|T)$ に類似しているようにしながら、なおかつ F をなるべく小さくすることを考える。類似の尺度は KL divergence を使う。

ここで、条件付独立という言葉进行定義しておく。二つの確率変数 P, Q は、 $\forall p, q, r \ Pr(P = p|Q = q|R = r) = Pr(P = p|R = r)$ であるとき、 R が与えられたうえで条件付独立であるという¹⁴。条件付独立が意味するところは、 R がわかっているれば、新たに Q を教えたとしても情報として無意味であるということである。

さて、 X を T の中のある素性としよう。さらに、 M は、 $M \subseteq T - X$ であるとする。 M は、次の条件が成立するとき、 X に対する Markov blanket と呼ばれる：「 M が与えられたうえで、 X と $(T \cup C) - (M \cup \{X\})$ は条件付独立である」。¹⁵

Markov blanket を持つ素性を削除することにより、 $Pr(C|T)$ と $Pr(C|F)$ の KL divergence は減少 (もしくは変化なし) することが示せる。実際には、完全な Markov blanket は存在しないことが普通だが、それに近いものを探していけば良い。しかし、Markov blanket を (近似的にでも) 見つけることは非常に大きなコストがかかる。よって、コスト削減のために、Markov blanket の候補として k 個の変数から成る集合のみを考える、また、さらに X ともっとも相関があるもの限定する、などの工夫がなされる。pseudo-code は図 5.6。

¹²最終的な手法評価のときの cross-validation とやることは同じだが、その目的を混同しないよう注意する必要がある。

¹³前段落で、 T は全素性集合としたが、ここでは素性を全部利用したときのある事例のベクトル表現を表す。不正確かつ confusing だ。

¹⁴この式、気持ちは伝わるが、僕はこういう表現 (| を二つも使った式) にはお目にかかったことがない。条件付独立の式は (P, Q, R は省略して書く)

$$Pr(p, q|r) = Pr(p|r)Pr(q|r)$$

が普通だと思う。本文でいう $Pr(p|q|r)$ とは、 $Pr(p|q, r) (= \frac{Pr(p, q|r)}{Pr(q|r)})$ のことであり、そう解釈すれば、条件付独立のとき、

$$\begin{aligned} Pr(p|q|r) &= \frac{Pr(p, q|r)}{Pr(q|r)} \\ &= \frac{Pr(p|r)Pr(q|r)}{Pr(q|r)} \\ &= Pr(p|r) \end{aligned}$$

となって筋が通る。

¹⁵Markov blanket の概念は、定義を読んだだけでは非常にとらえにくいと思う。素性とクラスをノードとしたグラフを思い浮かべて欲しい。統計的に依存しているノードはエッジでつなぐことにする。 X の Markov blanket とは、そのグラフ上で、 X と他のノードを切断するようなノードの集合であり、 X に「毛布」のように覆い被さっている。

5.5.3 Comparison and Discussion

前節までで、素性の良さを表すいくつかの指標や、指標を用いて素性を選択する手法を説明してきた。どの指標や手法がよいのか、という疑問は、その分類タスクに依存する。

私の経験では、文書の長さの問題に対して正しく対処してありさえすれば、指標などの違いで結果が大きく変わることはない。

Markov blanket は、時間がかかるが、次の二点で本質的な良さが期待できる。

- X_1 と X_2 は、共にクラスと相関が高いが、同様の情報を与えるとする。Markov blanket では、片方だけが素性に組み込まれるのに対し、greedy inclusion では両方入ってしまう。
- X_1 と X_2 は、共にクラスと相関が低いが、二つ合わせると良い素性になるとする。greedy inclusion では、両方とも捨ててしまう。

が、どちらも気にするほどではない。一つ目はちょっとした効率の問題であり、二つ目はあまり起こらない現象である。

私の経験では、各素性を必要か不必要かのどちらかにわけるよりも、その程度に応じて素性空間を变形する方が良い。これは、4章で見たような次元圧縮手法により実現できる。

5.6 Bayesian Learners

この節では、Bayesian Learner について述べる。Bayesian Learner は、もっとも実用的な分類器の一つといてよい。

簡単のため、各文書はただ一つのクラスに属するとする。文書の生起は以下のようにモデル化される。

1. 各クラス c は、prior (事前分布) $Pr(c)$ を持つ ($\sum_c Pr(c) = 1$)。この prior に従って、一つクラスが決まる。
2. クラスが与えられたときの文書の生起確率 $Pr(d|c)$ に従い、文書が生成される。

文書 d が与えられたとして、 d がクラス c から生起した確率は、ベイズの公式を使って、

$$Pr(c|d) = \frac{Pr(c)Pr(d|c)}{\sum_{\gamma} Pr(\gamma)Pr(d|\gamma)} \quad (5.12)$$

と書ける。

さて、ここからがベイズ推定である¹⁶。 $Pr(d|c)$ が内部に持つパラメータの集合を Θ とする。訓練データを D とすると、

$$\begin{aligned} Pr(c|d) &= \sum_{\Theta} Pr(c|d, \Theta)Pr(\Theta|D) \\ &= \sum_{\Theta} \frac{Pr(c|\Theta)Pr(d|c, \Theta)}{\sum_{\gamma} Pr(\gamma, \Theta)Pr(d|\gamma, \Theta)}Pr(\Theta|D) \end{aligned} \quad (5.13)$$

となる。この式の和は、パラメータが連続的なときは積分になる。この和、あるいは積分を本当に計算する方法を、Bayes optimal estimation (最適ベイズ推定) という。が、これは普通、計算量的に無理。よって応用の際には、 Θ を一意に決定する¹⁷。たとえば、Maximum Likelihood Estimation (MLE, 最尤推定) を使うと、 $\operatorname{argmax}_{\Theta} Pr(\Theta|D)$ なる Θ を選ぶことになる (しかしこれはうまく働かない)。

¹⁶ベイズ推定では、パラメータは一意に決定されない。たくさんのパラメータの値が確率的に足し合わされる。

¹⁷だから結局、あまりベイズ推定っぽくない。

5.6.1 Naive Bayes Learners

シンプルで速い学習器に、Naive Bayes Learner がある。naive という形容語句は、素性どうしが独立であるという大胆な仮定による。

二つモデルがある (see Section 4.4.1)。まず、binary model で考えてみる。これは、各単語がその文書内で生じたか否かを見るモデルである (何回生じたかはここでは関係ない)：

$$Pr(d|c) = \prod_{t \in d} \phi_{c,t} \prod_{t \in W, t \notin d} (1 - \phi_{c,t}). \quad (5.14)$$

計算量の削減のために、実際は次の式で計算される：

$$Pr(d|c) = \prod_{t \in d} \frac{\phi_{c,t}}{(1 - \phi_{c,t})} \prod_{t \in W} (1 - \phi_{c,t}).$$

$\prod_{t \in W} (1 - \phi_{c,t})$ は d に依らないので、前もって計算しておける。スパース性より、 $t \in d$ なる t が実際は僅かであるので、速く計算できる。

multinomial model では、 $|W|$ 個の面を持つサイコロを、文章の長さ分だけ振ると考えれば良い。そのうち各単語が何回でるか、というわけである。クラス c での単語 t の生起確率を $\theta_{c,t}$ とする。単語 t の文書 d 内での生起回数 (観測値) を $n(d,t)$ で表すことにする。 d の長さを $l_d (= \sum_t n(d,t))$ とする。 $Pr(d|c)$ は、次のように、多項分布を用いて書ける：

$$\begin{aligned} Pr(d|c) &= Pr(l_d|c)Pr(d|l_d, c) \\ &= Pr(l_d|c) \frac{l_d!}{n(d,t_1)!n(d,t_2)! \cdots n(d,t_{|W|})!} \prod_{t \in d} \theta_{c,t}^{n(d,t)}. \end{aligned} \quad (5.15)$$

$\frac{l_d!}{n(d,t_1)!n(d,t_2)! \cdots n(d,t_{|W|})!}$ は、分類に寄与しないので、無視される。 $Pr(l_d|c)$ も無視される (一様分布と見なされる) 事が多いが、本当はこれはしっかりモデルに組み込まなくてははいけないだろう。

どちらのモデルでも、非常に小さい値の確率が出てくるので、計算機で問題なく扱えるよう、実際には対数にして扱うことが普通である。また、一つのクラスに関してだけ、確率が大きくなる事が多く、クラスの候補を複数出すような働きはうまくできない。

後者の問題点を克服するために、2クラス問題の場合は、

$$\text{logit}(d) = \frac{1}{1 + \exp(-LR(d))} \quad (5.16)$$

where

$$LR(d) = \frac{Pr(C = +1|d)}{Pr(C = -1|d)}$$

Parameter smoothing

さきほど少し触れたように、パラメータ推定に MLE を使うのは、あまりいい手でない。それは、訓練データにおいて各クラスに生起する単語の種類は限られており、MLE では生起しなかった単語はそのクラスでの生起確率が 0 になってしまう。評価事例がそのような単語を含んでいると、その事例がそのクラスで生起する確率が 0 になってしまう。たとえ、他の単語がそのクラスであることを示唆していても、である。よって、smoothing をすることにより、生起数 0 の単語にも確率を少しだけ割り振ってあげる。

ベイズ推定を用いる¹⁸。まず、ベイズの定理より、

$$\pi(\phi | \langle k, n \rangle) = \frac{\pi(\phi)Pr(\langle k, n \rangle | \phi)}{\int_0^1 \pi(p)Pr(\langle k, n \rangle | p)dp} \quad (5.17)$$

¹⁸式 (5.13) に戻っていると考えるのもいい、のかも知れない。が、やってみないとわからないので、ここでは単純に「パラメータ推定にベイズ推定を使った」と考えておく。

である。ここで、 $\langle k, n \rangle$ は、 n 文書中 k 文書が、 ϕ に対応する単語を含んでいたことを示す。つまり、観測データを示している。

ここでは、パラメータの事前分布を用いて、データが与えられたときのパラメータの期待値を算出することになる¹⁹。binary model の場合、

$$\begin{aligned}
\hat{\phi} &= \frac{\int_0^1 p\pi(\phi)Pr(\langle k, n \rangle | p)dp}{\int_0^1 \pi(p)Pr(\langle k, n \rangle | p)dp} \\
&= \frac{\int_0^1 p^{k+1}(1-p)^{n-k}dp}{\int_0^1 p^k(1-p)^{n-k}dp} \\
&= \frac{\mathcal{B}(k+2, n-k+1)}{\mathcal{B}(k+1, n-k+1)} \\
&= \frac{\Gamma(k+2)\Gamma(n+2)}{\Gamma(k+1)\Gamma(n+3)} \\
&= \frac{k+1}{n+2}
\end{aligned} \tag{5.18}$$

となる²⁰。注目すべきは、 k/n は MLE による推定値なので、その分母分子と、事前分布の分母分子をそれぞれ足し合わせた形になっていることである。観測回数が多ければ、事前分布の影響はほとんどなくなることがわかると思う。

Multinomial model の場合は、結果だけ示すと、

$$\hat{\theta}_{c,t} = \frac{1 + \sum_{d \in D_c} n(d,t)}{|W| + \sum_{d \in D_c, \tau \in d} n(d,\tau)} \tag{5.19}$$

となる。つまり、全ての単語に、生起頻度 1 回の下駄を履かせたことになる²¹。

Comments on accuracy and performance

Multinomial model は、一般に、binary model より分類性能が良い。図 5.5 は、その例である。k-NN は、k を適切に推定すれば、Naive Bayes より分類性能が一般に良い。ただし、k-NN は時間がかかる。

¹⁹しかし、ベイズ推定を理解するために、もう少し詳しく書いておく。今求めようとしているパラメータは単語 t に関する生起確率だとして。すると、 ϕ とは、 $Pr(T=t | \langle k, n \rangle)$ のことである。

MLE などでは、ある事象 ω の生起確率を求める場合、パラメータを決定して、そのパラメータを用いて確率を計算する。しかし、ベイズ推定では、パラメータを決定しないで、確率的に足し合わせる。つまり、

$$\hat{Pr}(\omega|D) = \int Pr(\omega|\theta, D)p(\theta|D)d\theta$$

を計算する。この場合、推定しようとしている確率自身が ϕ というパラメータなので、上式の $\hat{Pr}(\omega|D)$ は $\hat{\phi}$ となり、 $Pr(\omega|\theta, D)$ が ϕ 自身である。さらに θ と ϕ を入れ換えると、

$$\hat{\phi} = \int \phi p(\phi|D)d\phi$$

となる。

²⁰以下の定義式と関係式を使えばこの変形を追える。 \mathcal{B} はベータ関数。 Γ はガンマ関数。

$$\begin{aligned}
\mathcal{B}(\alpha, \beta) &= \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \\
&= \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, \\
\Gamma(\alpha) &= \int_0^\infty x^{\alpha-1}e^{-x}dx, \\
\Gamma(\alpha+1) &= \alpha\Gamma(\alpha).
\end{aligned}$$

²¹確率的言語モデルの世界では、よくラプラス推定と呼ばれる。

素性空間を見てみると、Naive Bayes は、クラスごとに素性空間を分割している。境界付近では分類間違いが多くなるだろう。一つのクラスの事例が集中している部分は、間違いは少なくなるだろう。

これをもう少し詳しく見てみよう。multinomial model では、

$$Pr(d|c, l_d) = \frac{l_d!}{n(d, t_1)!n(d, t_2)! \cdots n(d, t_{|W|})!} \prod_{t \in d} \theta_{c,t}^{n(d,t)} \quad (5.20)$$

となる。2 クラスの場合を考えてみる。対数をとって考えると、分類するということは、

$$\log Pr(c = 1) + \sum_{t \in d} n(d, t) \log \theta_{1,t} \quad (5.21)$$

and

$$\log Pr(c = -1) + \sum_{t \in d} n(d, t) \log \theta_{-1,t}$$

の大きさを比べていることになる。 $\theta_{c,t}$ に依存したベクトル α_{NB} を考えると、結局、 $\alpha_{NB} \cdot d + b$ というある平面で空間を切っていることになり、Naive Bayes は線形分類器であることがわかる。²²

ここで、Naive Bayes の一つの問題点を指摘しておこう。 α_{NB} の各成分は、その成分に対応する単語の統計量のみから計算される。つまり素性をグローバルに見ることができていない。この問題点を克服するような分類器が後で紹介される。

5.6.2 Small-Degree Bayesian Networks

Naive Bayes では、素性（単語）どうしの統計的依存性はないという仮定を置いていた。このモデルは、図 5.7(a) のような形になっている。素性どうしの依存性を考えるならば、図でいうと例えば、図 5.7(b) のようになる。これは、Bayesian network を形成することに相当する。

Bayesian network とは、確率変数の間の依存関係を表す directed acyclic graph のことである。各ノードは確率変数に相当する。ノード X の親ノードの集合を表す確率変数を $\text{Pa}(X)$ と書くことにする。また、 $\text{Pa}(X)$ のインスタンスを $\text{pa}(X)$ と書く。

Bayesian network では、

$$Pr(\mathbf{x}) = \prod_x Pr(x|\text{pa}(X)) \quad (5.22)$$

となる。

グラフ構造と訓練データが与えられれば、Bayesian network を構成することが可能である。難しいのは、グラフ構造を決定することである（特にノード数が大きいとき）。計算を簡単にする一つの方法は、各ノードが持つ親ノードの数を制限することである。k-dependence Bayesian network では、各単語に他の単語から入ってくるエッジ数は最大 k という制限を持っている。

²²式 (5.21) の差をとって考えればよい。

$$\begin{aligned} & \log Pr(c = 1) + \sum_{t \in d} n(d, t) \log \theta_{1,t} > \log Pr(c = -1) + \sum_{t \in d} n(d, t) \log \theta_{-1,t} \\ \Leftrightarrow & \log Pr(c = 1) - \log Pr(c = -1) + \sum_{t \in d} n(d, t) \log \theta_{1,t} - \sum_{t \in d} n(d, t) \log \theta_{-1,t} > 0 \\ \Leftrightarrow & \log \frac{Pr(c = 1)}{Pr(c = -1)} + \sum_{t \in d} n(d, t) \log \frac{\theta_{1,t}}{\theta_{-1,t}} > 0 \end{aligned}$$

となるので、 $\alpha_{NB} = (\log \frac{\theta_{1,1}}{\theta_{-1,1}}, \dots, \log \frac{\theta_{1,|W|}}{\theta_{-1,|W|}})$ 、 $b = \log \frac{Pr(c=1)}{Pr(c=-1)}$ とすればよい。

図 5.8 は k-dependence Bayesian network を構成するアルゴリズムの pseudocode である。

Bayesian network を multinomial model に適用するのは、求めるべき条件付確率が非常に多くなり、厳しい (see 図 5.7)。binary model ならまだなんとかなる。しかし、機械学習系のデータセットでは精度向上がはっきりと見られたのに対し、文書分類ではあまり向上しなかった。文書分類はタスクとしてシンプルすぎるのであろう。もっと難しい問題でこそ、Bayesian network の力が発揮されるのではないだろうか？

5.7 Exploiting Hierarchy among Topics

問題によってはクラス間に関係があることがある。例えば、癌の危険性が「高い」「普通」「低い」のように、順序関係を持つ場合がある。文書分類では、「スポーツ」の下に「サッカー」や「バドミントン」があるというように階層構造を持つ場合もある。

この節では特に、クラスの階層構造について考えていく。

5.7.1 Feature Selection

クラスが階層構造を持つ場合、各素性の良さを表す指標は、階層構造のどの部分で評価されるかに大きく影響を受ける。指標の評価は、階層構造の内部ノードで行わなくてはならない(?)。例えば、単語“can”は、Yahoo!のトップカテゴリでは noisy な単語になるが、/Science/Environment/Recycling の下では、分類に大きな役割を果たすであろう。

5.7.2 Enhanced Parameter Estimation

クラス階層がパラメータ推定に役に立つと考えられるのはなぜか？それは、階層の下の方のクラスにおいて訓練事例が少なく信頼できるパラメータ推定ができないとき、訓練事例が多い上部クラスのパラメータを用いて事前確率を推定することにより、訓練事例の少なさをカバーできるからである。

この直観は、shrinkage と呼ばれる方法で定式化されている。クラス階層の中のルートからのパス $c_1 (= root), c_2, \dots, c_n$ を考える。 c_1 の訓練事例は c_n よりずっと多いはずなので、関連するパラメータの値もより信頼できる。shrinkage では、上部クラスのデータを下部クラスのパラメータ推定に用いる。root c_1 の上に dummy クラス c_0 を付け加える事により、スムージングが綺麗な形で導入できる (c_0 は root でのパラメータのスムージングに使われる)。

まず、 $\theta_{c_i,t}$ の MLE を計算しておく ($\forall i$)。 $\theta_{c_0,t}$ は $1/|W|$ とする。この時点ではラプラス法などのスムージングを施す必要はない。“shrunk” 推定値 $\tilde{\theta}_{c_n,t}$ は、以下のような線形補間で求められる：²³

$$\tilde{\theta}_{c_n,t} = \lambda_n \theta_{c_n,t}^{MLE} + \dots + \lambda_1 \theta_{c_1,t}^{MLE} + \lambda_0 \theta_{c_0,t}^{MLE}, \quad (5.23)$$

ここで、 $\sum_i \lambda_i = 1$ である。重み λ_i の値を決めなくてはならないが、これは held-out data H_n の確率を最大化するように調整することによって決定する。図 5.9 は、shrinkage の pseudocode である。これは EM の特殊なケースになっている。²⁴

²³本文の式 (5.23) では $\tilde{\theta}_{c_n,t}$ を求めているが、間違ってる気がする。

²⁴これを導くには、式の意味をもう一度見直して、それを EM の式にあてはめる必要がある。そもそも $\tilde{\theta}_{c_n,t}$ とは何か？ 意味を考えると、 $Pr(t|C_0 \cup \dots \cup C_n)$ なのだが、そう考えると次のように式変形ができて、筋が通る：

$$\begin{aligned} Pr(t|C_0 \cup \dots \cup C_n) &= \frac{1}{Pr(C_0 \cup \dots \cup C_n)} Pr(t, C_0 \cup \dots \cup C_n) \\ &= \frac{1}{\sum_j Pr(C_j)} \sum_i Pr(t, C_i) \quad (\text{排反性より}) \end{aligned}$$

Shrinkage は、Industry と 20NG、さらに Yahoo!のある一部分に対して適用され、精度向上が確認されている。特にスパースなデータにはよく効き、多数の素性に対しても overfitting しにくいようである。

5.7.3 Training and Search Strategies

ここでは、クラスが階層構造を持つ場合に、どのようにして実際にクラスを決定していくのかを考える ($Pr(c|d)$ などの確率を与える方法は整っているとする)。

クラスの階層構造は、普通、“is-a”階層である。例えば、/Arts/Photography に属する文書は、/Arts にも属する。

評価事例 d に対し、目的は最も適当な葉ノードを決定することだとしよう。今、一つの文書は一つのパスだけに属し得ると考えると、 $\{c_1, c_2, \dots, c_m\}$ が c_0 の子ノードを表すとして、

$$\sum_i Pr(c_i|d) = Pr(c_0|d) \quad (5.24)$$

である。

$$\begin{aligned} &= \frac{1}{\sum_j Pr(C_j)} \sum_i P(C_i) Pr(t|C_i) \\ &= \sum_i \frac{P(C_i)}{\sum_j Pr(C_j)} Pr(t|C_i). \end{aligned}$$

つまり、 $\lambda_i = \frac{P(C_i)}{\sum_j Pr(C_j)}$ であることがわかる。 t が C_i から生じた確率を求めると、

$$\begin{aligned} \frac{P(t, C_i, C_0 \cup \dots \cup C_n)}{P(t, C_0 \cup \dots \cup C_n)} &= \frac{Pr(t, C_i)}{\sum_j Pr(t, C_j)} \\ &= \frac{Pr(t|C_i) Pr(C_i)}{\sum_j Pr(t|C_j) Pr(C_j)} \\ &= \frac{\lambda_i Pr(t|C_i)}{\sum_j \lambda_j Pr(t|C_j)} \quad (\text{分母分子を } \sum_l Pr(C_l) \text{ で割った}) \\ &\equiv \gamma_{i,t} \end{aligned}$$

となって、 $\beta_i = \sum_t \gamma_{i,t}$ であることに注意。EM に従って対数尤度の期待値をとると、

$$\begin{aligned} Exp[L] &= \sum_t \sum_i \gamma_{i,t} \log Pr(t, C_i) \\ &= \sum_t \sum_i \gamma_{i,t} \log(\lambda_i Pr(C_i \cup \dots \cup C_n) Pr(t, C_i)). \end{aligned}$$

これを、 $\sum_i \lambda_i = 1$ なる条件下で最大にする λ 's を求めたい。Lagrange 関数を作ると、 $Lagr = Exp[L] + \alpha(\sum_i \lambda_i - 1)$ 。 λ_k で偏微分すると、

$$\frac{\partial Lagr}{\partial \lambda_k} = \sum_t \gamma_{t,k} \frac{1}{\lambda_i} + \alpha$$

なので、これを 0 とおくと、

$$\begin{aligned} \sum_t \gamma_{t,k} \frac{1}{\lambda_i} + \alpha &= 0 \\ \frac{1}{\lambda_i} \beta_i &= -\alpha \\ \lambda_i &= -\frac{1}{\alpha} \beta_i \\ &= \frac{\beta_i}{\sum \beta_i}. \quad (\text{足して 1 の制約を使った}) \end{aligned}$$

このようにして、shrinkage のアルゴリズムが得られる。

Greedy search

この方法では、ルートノードから始めて、greedily にクラスを決定する。各ノードにおいて最も良い子ノードを選ぶという作業を繰り返していく、という単純なアルゴリズムである。明らかにわかる欠点としては、階層の上の方で間違った判断を下してしまうと、もう救いようがないということである。

Best-first search

一般に、

$$Pr(c_i|d) = Pr(c_0|d)Pr(c_i|c_0, d) \quad (5.25)$$

が成り立つ。変形すると、

$$-\log Pr(c_i|d) = (-\log Pr(c_0|d)) + (-\log Pr(c_i|c_0, d)) \quad (5.26)$$

となる。ここで、エッジ (c_0, c_i) に重み $-\log Pr(c_i|c_0, d)$ が付与されていると考えてみる。そうすると、最も適当な葉ノードを見つけるということは、ルートから最短であるような葉を見つけることと等価であることがわかる。pseudocode は図 5.10 である。

各ノードにおいて、 $Pr(c_i|c_0, d)$ の値が、一位のものだけ非常に大きいと、greedy search と best-first search は変わらなくなってしまう。よって、5.6.1 節の最後あたりで紹介したような、確率の rescaling が必要である（たとえば、logit 関数とか使うやつ）。

三層の階層構造を持つ、U.S. Patent taxonomy をデータとして実験を行った。実験結果は図 5.11 に示す。best-first search の方が、スピード、精度、ともに優れていた。

The semantics of hierarchical classification

これまで、文書を葉ノードに対応付けることを目標にしてきたが、内部ノードに対応付けたい場合もあるだろう。たとえば、どの子ノードも適切なノードと言えなかったとき、多くの子ノードが適切だと判断されるとき、子ノードに対応付けることによる間違いを避けたいとき、などがある。そのような場合は分類方法を修正しなくてはならないだろう。また、階層構造が“is-a”階層でない場合もあるだろう。そのような場合は新しい研究対象となるだろう。

5.8 Maximum Entropy Learners

Bayesian アプローチでは、 $Pr(d|c)$ をモデル化し、それを用いて $Pr(c|d)$ が計算される。しかし、二つの問題点を指摘しよう。一つは、 d は非常に高次元の空間に存在するので、小さな訓練データでは $Pr(d|c)$ を正確に推定できないことである。もう一つは、synthesize された素性（フレーズ、POS タグ、意味タグなど）を素性として使用するのが非常に危険であるということである。そういった素性は他の素性と高い相関を持つことが多く、naive bayes assumption が崩れるばかりか、他の有効な素性の効果を薄めてしまう可能性がある。この節では、こういった問題点を持たない方法を考える。もう少し言うと、 $Pr(c|d)$ を直接モデル化する方法である。

訓練データ $\{(d_i, c_i), i = 1, \dots, n\}$ が与えられたとしよう。 d_i はベクトル。 c_i はクラスラベルである。ここで、 d と c に関連した条件の真偽を表す関数、indicator functions $f_j(d, c)$ を定義する²⁵。ある indicator f_j の期待値は、

$$E(f_j) = \sum_{d,c} Pr(d, c) f_j(d, c) \quad (5.27)$$

²⁵一般の文献では、単に「素性」と呼ばれるが、ここでは前出の素性と明確に区別するために indicator function と呼ぶことにする。

$$\begin{aligned}
&= \sum_{d,c} Pr(d)Pr(c|d)f_j(d,c) \\
&= \sum_d Pr(d) \sum_c Pr(c|d)f_j(d,c)
\end{aligned} \tag{5.28}$$

と表される。さて、ここで、推定したいの $Pr(c|d)$ を除いた、 $Pr(d,c)$ と $Pr(d)$ をデータから算出される値 $\hat{Pr}(d,c)$ と $\hat{Pr}(d)$ と置き換えても、式 (5.27) = 式 (5.28) が成立するという制約を置く。つまり、

$$\begin{aligned}
\sum_i \hat{Pr}(d_i, c_i) f_j(d_i, c_i) &= \sum_i \hat{Pr}(d_i) \sum_c Pr(c|d_i) f_j(d_i, c) \\
&or \\
\sum_i \frac{1}{n} f_j(d_i, c_i) &= \sum_i \frac{1}{n} \sum_c Pr(c|d_i) f_j(d_i, c)
\end{aligned} \tag{5.29}$$

が成立するという制約を置いたのである。文書 d_i は確率 $1/n$ で生じたと考えている。

この式で、未知なのは $Pr(c|d)$ のみであり、それらについて解けば $Pr(c|d)$ を直接モデル化できたことになる。しかし、そうはいかない。一般に上方程式の数は $Pr(c|d)$ の数より少なく、解に不定性が残るからである。多くの可能解のうち、どの解を選べばよいのだろうか？ ここで、「最大エントロピー原理」に従う。上制約を満たすもののうち、もっともエントロピーが高い、つまり、各 $Pr(c|d)$ に最も平等に確率を割り振っているものを採用する。これは、Occam's razor の考え方と同じであるといえる。

さて、ではこの最大化問題を解いてみよう。Lagrangian は次のように書ける：

$$\begin{aligned}
G(Pr(c|d), \Lambda) &= - \sum_{d,c} Pr(d)Pr(c|d) \log Pr(c|d) \\
&+ \sum_j \lambda_j \left(\sum_i f_j(d_i, c_i) - \sum_{i,c} Pr(c|d_i) f_j(d_i, c) \right).
\end{aligned} \tag{5.30}$$

ここで、 Λ は、 $\{\lambda_1, \dots\}$ を表す。 G を $Pr(c|d)$ について偏微分することにより、²⁶

$$\begin{aligned}
Pr(c|d) &= \frac{1}{Z(d)} \exp \left(\sum_j \lambda_j f_j(d, c) \right) \\
&= \frac{1}{Z(d)} \prod_j \mu_j^{f_j(d,c)}
\end{aligned} \tag{5.31}$$

が示せる。ここで、 $\mu_j = \exp\{\lambda_j\}$ であり、 $Z(d)$ は正規化因数である。

実際に、与えられたデータに対して分布を推定するには、次の2ステップが必要になる。

1. 事前知識から、あるいは試行錯誤により、indicator function の集合を決定する。各 indicator function には、一つのパラメータ λ が対応する。
2. パラメータを求める。

最大エントロピーモデルは、ここでやったように制約付最適化問題を解くことによっても得られるが、実は最尤推定 (MLE) の枠組でもとらえることができる。確率モデルの形を式 (5.31) のように決めたいので MLE を適用することと最大エントロピーモデルを用いることは等価であることが示されている。

単純な indicator function としては、クラスと単語のペアについて以下のようなものが考えられる：

$$f_{c',t}(d, c) = \begin{cases} 1 & \text{if } c = c' \text{ and } t \in d \\ 0 & \text{otherwise.} \end{cases} \tag{5.32}$$

²⁶未確認。

あるいは、multinomial model のように、

$$f_{c',t}(d,c) = \begin{cases} 0 & \text{if } c \neq c' \\ \frac{n(d,t)}{\sum_{\tau} n(d,\tau)} & \text{otherwise.} \end{cases} \quad (5.33)$$

でもよい。最大エントロピーモデルは、naive Bayes を精度において上回ることが多いが、常にというわけではない。データセットに依存する。むしろ、最大エントロピー法の長所とは、synthesize された素性をうまく扱える点にあるといえよう。しかし、驚くことに、そのような素性を有効的に利用した応用研究に出会ったことがない。

5.9 Discriminative Classification

ここまでは、確率的な生成モデルを見てきた。この節では、discriminative な分類器を二つ紹介する。discriminative classifier は、素性ベクトルを直接、クラスラベルに写像する。ここでは、線形分類器を扱うので²⁷、我々は、 $\text{sign}(\alpha \cdot d + b)$ が正例負例を正しく分離するような、ベクトル α とスカラー b を求めることを目標とする。

5.9.1 Linear Least-Square Regression

Naive Bayes も、Maximum entropy も、線形分類器であるが、線形分類器を構築するのにそういった確率モデルを通過する必要は必ずしもない。文書 d_i のラベルを正しく与えてくれる $\alpha \cdot d_i + b$ がわかればよい。まず、linear regression (線形回帰モデル) という枠組を見てみよう。linear regression では、二乗誤差を最小にすることを目的として分類器が構築される。二乗誤差は、 $\sum_i (\alpha \cdot d_i + b - c_i)^2$ として表される。この最小化は、よく最急降下法などを使って解かれる。たとえば、Widrow-Hoff (WH) update rule などがある。WH では、適当な初期値を α に与え、訓練事例 (d_i, c_i) を一つずつ見ながら、 $\alpha^{(i-1)} \rightarrow \alpha^{(i)}$ と更新していく。更新式は、

$$\alpha^{(i)} = \alpha^{(i-1)} + 2\eta(\alpha^{(i-1)} \cdot d_i - c_i)d_i \quad (5.35)$$

である²⁸。 η は適当な定数である。更新していく間に現れたあらゆる α の平均を、最終的な α として用いるのが普通である。

一般性を失わず、 $|\alpha| = 1$ という制約を置くことができる。このようにすると、以下のような等価な二つの解釈が可能になる：

- 分類器は、正例と負例をできるだけうまく分離する超平面である。
- 各文書は、 α の方向に射影され、一つのスカラー値として表現される。

一般に、このモデルによる文書分類は、naive Bayes より精度的に優れており、k-NN と同等の精度を出す。

5.9.2 Support Vector Machines

SVM は現在、文書の分類器としては最も精度が高いものである。

²⁷Naive Bayes も、Maximum entropy も、線形分類器であることが示せる。Maximum entropy の場合は、

$$\log Pr(c|d) = -\log Z_d + \sum_{t \in d} f_{c,t}(d,c) \log \mu_{c,t} \quad (5.34)$$

である。

²⁸この式は、二乗誤差の偏微分を用いて、右辺第二項の係数を算出している。

Naive Bayes のように、線形の SVM は $\alpha_{SVM} \cdot d + b$ という値を用いて事例を分類する。しかし、SVM においては α_{SVM} はずっと慎重に選ばれる。

SVM は、訓練事例からの距離が最大になるように分離平面を構成する。これは次のように書ける：

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} \alpha \cdot \alpha \\ \text{s.t.} \quad & c_i(\alpha \cdot d_i + b) \geq 1 \quad \forall i = 1, \dots, n, \end{aligned} \quad (5.36)$$

ここで、 $\{d_1, \dots, d_n\}$ は訓練事例の集合で、 $\{c_1, \dots, c_n\}$ は対応するクラスである。

最適分離平面は、両クラスの凸包をつなぐ最短線分と直交しており、またその線分を二等分する（図 5.12）。全ての c_i に対して $c_i \in \{-1, 1\}$ なので、 α と b は、全ての訓練事例に対して

$$c_i(\alpha \cdot d_i + b) \geq 1 \quad (5.37)$$

が成り立つ。もし、 d_1 と d_2 が互いに異なるクラスの最近点を表すベクトルであれば、

$$\alpha \cdot (d_1 - d_2) = 2 \quad (5.38)$$

である。それゆえ、

$$\frac{\alpha}{\|\alpha\|} \cdot (d_1 - d_2) = \frac{2}{\|\alpha\|} \quad (5.39)$$

である。左辺は、最大の場合でも $|d_1 - d_2|$ なので、異なるクラスの事例間の距離は少なくとも $2/\|\alpha\|$ であり、マージンは、 $1/\|\alpha\|$ である。

実データは、完全に線形分離できないのが普通である。よって、slack 変数²⁹ $\{\xi_1, \dots, \xi_n\}$ を導入して、

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} \alpha \cdot \alpha + C \sum_i \xi_i \\ \text{s.t.} \quad & c_i(\alpha \cdot d_i + b) \geq 1 - \xi_i \quad \forall i = 1, \dots, n, \\ \text{and} \quad & \xi_i \geq 0 \quad \forall i = 1, \dots, n, \end{aligned} \quad (5.40)$$

と制約を少し緩和してあげる。直観的解釈は、もし、 d_i が間違って分類されると、 $\xi_i \geq 1$ になることから、結局 $\sum_i \xi_i$ は訓練事例に対するエラーの上限になっていることがわかる。エラーをどれくらい重く見るかというパラメータが、 C である。実は、このままの式ではうまく解けないので、Lagrange 法を用い、次のような双対問題に帰着させ、それを解くのが普通である：³⁰

$$\begin{aligned} \text{Maximize} \quad & \sum_i \lambda_i - \frac{1}{2} \lambda_i \lambda_j c_i c_j (d_i \cdot d_j) \\ \text{s.t.} \quad & \sum_i c_i \lambda_i = 0 \\ \text{and} \quad & 0 \leq \lambda_i \leq C \quad \forall i = 1, \dots, n. \end{aligned} \quad (5.41)$$

ここで、 $\lambda_1, \dots, \lambda_n$ は Lagrange 乗数である。

式 (5.41) は、二次計画問題である。これですっと解き易くなったのだが、問題のサイズが大きいの、まだ大変である。なので、高速化&コンパクト化のためにいろいろな工夫がなされている。例えば、この二次計画問題をイッキに解くのではなく、変数を少数もってきて、それについて最適化する（他の変数は定数とみなして）、それを繰り返して、収束してきたら、本当にグローバルに最適になっているかどうかチェックする。これ

²⁹本文では fudge variable と呼んでいる。

³⁰帰着する過程を追ってもよかったけど、このまえ学習勉強会でやったし、もういいかなと思ってやめました。

は Sequential Minimal Optimization (SMO) と呼ばれる演算手法である。また、 $d_i \cdot d_j$ を全てのペアについて計算してメモリに持つのは、大抵の場合苦しい。なので、それらはその都度計算する。でも、よく使うペアに関しては、キャッシュに持っておく。泥臭いが重要なテクニックらしい。

SVM は時間がかかるといわれるが、図 5.13 にデータサイズと計算時間の関係が示されている。オーダー的には、 n^a ($a \approx [1.7, 2.1]$) らしい³¹。

Reuters を用いた文書分類で分類精度を、naive Bayes や k-NN と比較したら、図 5.14 みたいな感じになる。SVM は他の二手法を上回っている。

別のデータセットを使った実験結果は、図 5.15 に示してある。どのデータセットについても SVM が最も良い。また、図 5.1 も SVM の優位性をサポートしている。

文書分類では、一般に使われるデータはほとんど完全に線形分離可能である。よって、文書分類では線形の SVM で充分だと考えられている。

5.10 Hypertext Classification

ここまでは、hypertext ならではの素性についてまったく議論してこなかった。この節では、hypertext の supervised classification について考える。HTML は、テキスト部分以外に、多くの異なる素性を持つ。しっかりと書かれた HTML は、tree-structured Document Object Model (DOM) と呼ばれるモデルによって記述された階層構造を持つ。

DOM tree では、内部ノードは要素 (UL, LI など) で、葉ノードのいくつかはテキストになっている。また別のノードは他の Web page へのリンクになっていたりする。

ここでは、そういった多様性のある素性を supervised learning にいかにして使用するかについて考える。hypertext を記述するのに、「関係」を使いたくなることがしばしばである。しかし、これまででできた supervised classifier は、関係をうまく記述できない。よって、この節の最後に新たに、inductive learning (帰納学習) の説明をする。

5.10.1 Representing Hypertext for Supervised Learning

サーチエンジンでは、いくつかの特定のタグにヒューリスティックな重みを割り当てるということがよくなされる。例えば、TITLE, H1, STRONG, EM などがある。そういうタグに着目することは、supervised learning でも重要である。次の例は少し人為的ではあるが、タグ情報がいかに重要かを示している。XML のようなフォーマットで書かれた履歴書は次のような記述を含んでいるかもしれない：

```
<resume>
  <publication>
    <title>Statistical models for Web-surfing</title>
  </publication>
  <hobbies>
    <item>Wind-surfing</item>
  </hobbies>
</resume>
```

ここで “surfing” という単語が二回出現しているが、これを区別することは非常に重要である。そのような区別は、DOM tree を root から辿ってそれぞれの surfing を発見する過程を素性に組み込むことによって、簡単

³¹理論的な最悪計算量は n^3 だったと思う。

に実現できる (resume.publication.title.surfing、resume.hobbies.item.surfing といった具合に)。そのパスの一部分だけ使ってもよい。

このような簡単な方法でも、分類精度を向上させることがある。Yi and Sundaresan の実験では、この手法を用いて、エラー率が 70% から 24% に減少した。

しかし、この方法は、まだ ad hoc と言わざるをえない。また、柔軟性に欠ける。例えば、hyperlink から得られた情報をいかにして素性に組み込むかは明らかでない。「関係」は、hypertext の情報を素性化する一般的な枠組であるといえよう。以下に例を挙げる。

```
contains-text(domNode, term)
part-of(domNode1, domNode2)
tagged(domNode, tagName)
links-to(srcDomNode, dstDomNode)
contains-anchor-text(srcDomNode, dstDomNode, term)
classified(domNode, label)
```

この後、多くの関係の集合から、規則を発見できる inductive classifier について勉強する。例えば、次のような規則を発見できるかもしれない：

```
classified(A, facultyPage) :-
  contains-text(A, professor), contains-text(A, phd),
  links-to(B, A), contains-text(B, faculty).
```

ここで、プロローグの記法を用いた (:- は “if” を表し、コンマは “かつ” を表す)。

5.10.2 Rule Induction

簡単のため、2 クラス分類 (正例集合 D_+ 、負例集合 D_-) で説明をする。rule finder は、predicate rule (述語規則) の集合を返す。もし、ある評価事例に対して一つの規則が true と言え、その事例は正例と分類され、そうでなければ負例と分類される。各規則は、述語規則の conjunction として表現される。

図 5.16 が、pseudocode を示す。これは、有名な FOIL (first-order inductive logic) に似たアルゴリズムである。外側のループは、新しい規則を一つずつ生成し、カバーされた正例を削除する。新しい規則ができると、負例も一緒に取り込んでしまうことがあるので、内側のループは、その規則を書き換えることにより (具体的には、literal を付け加えることにより) 負例の取り込みを回避する。literal の選び方も図 5.16 に書かれている (直観的には、なるべく多くの事例を良い精度で分類するような rule を作る literal が選ばれる。要 confirm)。

relational learning の長所は、ページ間の関係を学習できることである。例えば、事例を次のようにコード化することができる：

```
member(homePage, department),
teaches(homePage, coursePage),
advises(homePage, homePage),
writes(homePage, paper).
```

拡張として面白そうなのは、これを統計的な分類器と組み合わせることである。

ラベル付け関係は次のように再帰的にできる：

```
classified(A, facultyPage) :-
  links-to(A, B), classified(B, studentPage),
```

```
links-to(A, C), classified(C, coursePage),  
links-to(A, D), classified(D, publicationsPage).
```

このような再帰的な手法は、非常に興味深いですが、同時にミスを伝搬する可能性もある。これを回避する手法は、6章で学ぶ。