平成24年度茨城大学工学部情報工学科卒業研究論文

分布間距離の推定を利用した語義曖昧性解消の領域適応

情報工学科

菊池裕紀（09T4020A）

指導教員　　新納浩幸 准教授

平成25年2月8日

平成 24 年度 茨城大学工学部情報工学科 卒業研究論文

分布間距離の推定を利用した語義曖昧性解消の領域適応

著者：菊池裕紀 (09T4020A)
指導教員：新納浩幸　准教授

論文要旨

従来の自然言語処理のタスクにおける機械学習では、訓練データとテストデータは同じドメインのコーパスから得たデータであることが前提であった。しかし、あるドメインのデータのラベルを識別したいにも関わらず、別のドメインのデータからしか識別規則を学習できないことも多い。そこで、あるドメイン（ソースドメイン）の訓練データから学習された分類器を、別のドメイン（ターゲットドメイン）のテストデータに適用できるようにチューニングすることを考える。これを領域適応といい、近年、転移学習の一部として活発に研究が行われている。異なるドメインに現れる単語は、意味の分布が異なる。これは語義曖昧性解消 (Word Sense Disambiguation, WSD) の領域適応の問題である。この場合、ターゲットドメインからある程度の訓練データを得ることが出来れば、領域適応の問題は発生しない。そのため領域適応の手法は、ターゲットドメインからラベル付きデータを用意するか（教師付き学習）、少量のラベル付きデータと大量のラベル無しデータを用意するか（半教師付き学習）、ラベル無しデータを用意するか（教師無し学習）の 3 通りに分けることが出来る。領域適応では、(半) 教師付き学習の手法が中心に研究されており、教師無し学習の研究事例は少ない。ただし、(半) 教師付き学習に使用する「ラベル付きデータ」がどの程度必要かは確かでない。また、教師無し学習の方が理想に近いことは明らかである。そこで本論文では、教師無し学習での実験を試みる。

本論文では、WSD をタスクに設定している。WSD に対しては、分類問題において一定の評価を得ている Naive Bayes（NB）を用いる。しかし、ターゲットドメインから訓練データを用意しない場合は領域適応の問題が生じてしまう。本論文では、ターゲットドメインの語義分布をソースドメインのコーパスとターゲットドメインのコーパス、およびソースドメインの語義分布から推測することで領域適応を行う。具体的には、ターゲットドメインの語義分布をソースドメインの語義分布と一様分布の混合分布で表す。このとき、混合比の設定が問題となる。そこでまずターゲットデータとソースデータを 4 つのグループにクラスタリングし、そこからターゲットデータとソースデータの分布間距離を求める。求めた分布間距離の逆数を混合比に設定する。

実験では、BCCWJ コーパスの新聞、Yahoo!知恵袋、書籍の 3 つのドメインのコーパスを利用し、各コーパスにある程度頻出する単語を 10 個選び、その単語について各コーパスから 50 個の用例を取り出して単語の語義ラベルを付けたものを使用した。従来の NB を用いて、ソースドメインの訓練データから学習した分類器を使って、ターゲットドメインのテストデータの語義を推定した正解率をベースラインとする。実験の結果、ベースラインからの正解率の向上はほとんど見られず、下がってしまう結果も見られた。クラスタリングの際に、クラスタ数を 4 つに固定してしまったため分布間距離の推定の精度が悪かったことが原因の 1 つであるが、それを利用した NB の補正手法を単純に設定しすぎたことも原因と考えられる。今後は、分布間距離の推定にさまざまな手法を試していき、正解率を改善していきたい。

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

従来、自然言語処理における機械学習では、訓練データとテストデータは同じドメインのコーパスからのデータであった。しかし、あるドメインのデータのラベルを識別したいにも関わらず、別のドメインのデータからしか識別規則を学習できないことも多い。そこで、あるドメイン（ソースドメイン）の訓練データから学習された分類器を、別のドメイン（ターゲットドメイン）のテストデータ用にチューニングすることを考える。これを領域適応（domain adaptation）という。領域適応は、転移学習（transfer learning）の一部であり、近年盛んに研究が行われている。本論文では、語義曖昧性解消（Word Sense Disambiguation, WSD）のタスクでの領域適応の研究を行っている。

一般に、異なるドメインに現れる単語は、意味の分布が異なることが多い。これは WSD の領域適応の問題である。ターゲットドメインからある程度の訓練データを構築できるのであれば、WSD における領域適応の問題は生じることは無い。そのため領域適応の手法は、ターゲットドメインから少量のラベル付きデータを用意するか（教師付き学習）、少量のラベル付きデータと十分な量のラベル無しデータを用意するか（半教師付き学習）、ラベル無しデータを用意するか（教師無し学習）の 3 通りに分けることができる。

従来、領域適応では、（半）教師付き学習の手法が中心に研究されており、教師無し学習の研究例は少ない。ただし、(半) 教師付き学習に使用する「ラベル付きデータ」がどれほど必要なのかは確かでない。また、教師無し学習のほうが理想に近いことは明らかである。そこで、本論文では、教師無し学習での実験を試みる。

本論文では、WSD に対する手法としてナイーブベイズ（Naive Bayes, NB）を用いる。対象単語 w の語義の集合を S とし、w の文脈を素性ベクトル $f = (f_1, f_2, \cdots, f_m)$ で表すと、NB では以下の式により w の語義を求めることができる。

$$arg \max_{s \in S} p(s) \prod_{i=1}^{m} p(f_i \mid s)$$

つまり、$p(s) \prod_{i=1}^{m} p(f_i \mid s)$ の値を最大にするクラス s に対象単語を分類することとなる。NB では、$p(s)$ と $p(f_i \mid s)$ をソースドメインの訓練データから推定する。今、ソースドメインから求めたそれぞれの分布を $p_s(s)$ と $p_s(f_i \mid s)$ とおく。またターゲットドメインにおけるそれぞれの分布を $p_t(s)$ と $p_t(f_i \mid s)$ とおく。本論文では、$p_t(f_i \mid s) = p_s(f_i \mid s)$ と仮定して、$p_t(s)$ をソースドメインのコーパスとターゲットドメインのコーパス、および $p_s(s)$ から推測することで領域適応を行う。本手法の特徴は、$p_t(s)$ の推定に $p_s(s)$ と $p_t(s)$ の分布間距離を利用する点である。

実験では、BCCWJ コーパスの新聞（PN）、Yahoo!知恵袋（OC）、書籍（PB）の 3 つのドメインのコーパスを利用した。各コーパスにある程度頻出する単語 10 個を選び、各単語について各コーパスから 50 個の用例を取り出して対象単語の語義ラベルを付けた。つまり、領域適

応としては 6 パターン存在し、各パターンにおいてソースドメインの訓練データ 50 個、ターゲットドメインのテストデータ 50 個が存在する。ソースドメインの訓練データから学習した分類器を使って、ターゲットドメインのテストデータを識別した場合の正解率がベースラインとなる。実験の結果、ベースラインから正解率の向上はほとんど見られず、逆に下がってしまう結果も見られた。分布間距離の推定精度が悪かったことが原因の 1 つであるが、それを利用した $p_t(s)$ の推定が安直過ぎたと考えている。今後は、分布間距離の推定に対して様々な手法を試し、$p_t(s)$ の推定を改善していきたい。

1.2 本論文の構成

第 2 章では、WSD に関しての一般的な解釈と手法に関して述べる。第 3 章では、領域適応に関しての問題点や関連研究についてを述べる。第 4 章では、本実験で使用するナイーブベイズ分類器やその他の分類器に関しての知識について述べる。第 5 章では、確率分布間距離の尺度の紹介と推定手法について述べる。第 6 章では、本論文において提案する手法について紹介する。第 7 章では、実験の設定方法とその結果について述べる。第 8 章では、実験の結果から考察を行う。第 9 章では、結論と今後の展望について述べていく。

第２章　語義曖昧性解消（WSD）

2.1　概要

単語には、一般に複数の意味が存在する。しかし、実際に文中で用いられる意味はそのうちの１つであり、それは文脈の情報などから決定される。例えば、英単語の「fire」には少なくとも次の４つの意味が存在する[1]。

1. uncontrolled flames, lights, and heat that destroy and damege things
2. to force someone to leave their job
3. to shoot bullets or bombs
4. to make someone feel interested in something and excited about it

それぞれの意味に対して次のような例文が考えられる[2]。

1. The fire was burning brightly.
2. He was fired from the job.
3. The enemy fire a rifle.
4. Her story fired my imagination.

これら４つの例文には「fire」という単語が現れており、それぞれ箇条書きの番号に対応した特定の意味が用いられている。

自然言語処理において、文章中に現われた単語が、どのような意味で用いられているのかを正しく判定することを語義曖昧性解消 (Word Sense Disambiguation，WSD) という。

2.2　一般的な手法

WSD では、教師付き学習（supervised)、半教師付き学習（semi-supervised)、教師無し学習（unsupervised）を用いた三種類の手法が存在する。教師付き学習（supervised）は、ソースドメインからの十分な量のラベル付きデータとターゲットドメインの少量のラベル付きデータを用いて学習を行うものである。半教師付き学習（semi-supervised）は、ソースドメインからの十分な量のラベル付きデータとターゲットドメインの十分な量のラベル無しデータを用いて学習を行うものであり、ラベルあり・無しの混在データから学習することで、ラベルありのデータだけで学習したときよりも精度を上げることを目的としている。教師無し学習（unsupervised）

[1] ロングマン現代英英辞典 [４訂新版] より
[2] ジーニアス英和辞典 [第４版] より

は、ソースドメインのラベル付きデータのみで学習する手法である。教師無し学習による WSD の流れを図 2.1 に示す。

学習には、文書から大量に集めたコーパスから得られるデータを用いる。このコーパスには、単に文書を集めたコーパスのほかに、文書中の単語に語義の情報や構文構造などの付加情報を含む自然言語処理に特化した注釈付きコーパスが存在する。注釈付きコーパスを使用することにより、高い語義識別の精度を得ることができるが、大量の文書に付加情報をつけていく作業は手間がかかる上に相当な時間とコストを必要とする。学習データとなるものには、文章中の単語に品詞をつけた POS タグや、対象単語の文脈に含まれる単語を集めたもの（bag-of-words）などがある。

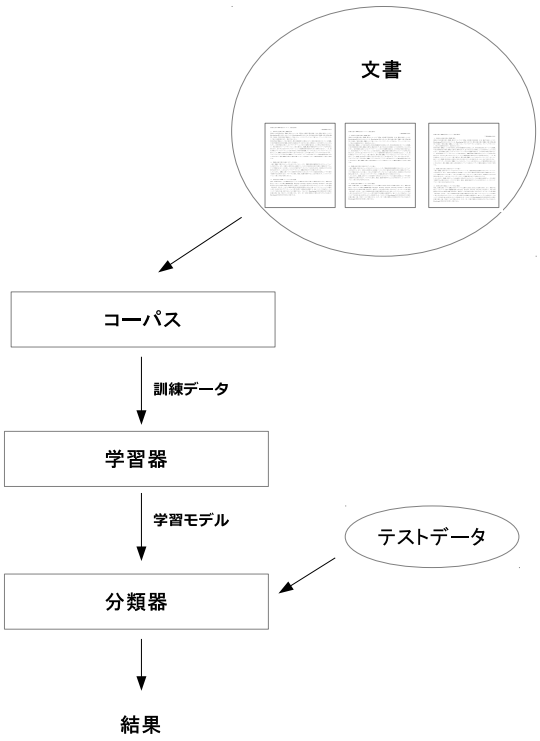

図 2.1: 教師無し学習による WSD の流れ

2.3 関連研究

WSD に関する研究は古くから行われている。先行研究の手法としては、辞書の定義文を用いる手法、用例をベースとした手法、語義のタグ付きコーパスを用いた機械学習による手法などが存在する。この 3 つのうちでは、機械学習による手法が有能であるとされている。しかし、機械学習に基づいた手法では、ソースドメインに出現する単語に対してしか語義の曖昧性を解消できない。そこで玉垣ら [13] は、さまざまな単語に対して WSD を行うためには機械学習による分類器とそれ以外の分類器を組み合わせて用いることが有効であるとし、辞書の用例を用

いた分類器、語義の出現条件を用いた分類器、語義のタグ付きコーパスを用いた分類器の三種類を組み合わせることによって読解支援システムのための WSD に関する研究を行っている。

第3章　転移学習（transfer learning）

3.1　概要

転移学習には、他にも帰納転移（inductive transfer）や領域適応（domain adaptation）、マルチタスク学習（multitask learning）などの多数の呼び名が存在する。このことからも分かるように、転移学習という単語は、かなり幅広い機械学習の枠組みに対して使われており、統一された意味を考えることは難しい。しかし、形式的ではないが、転移学習のワークショップの論文募集 [16] 中で示されている次の定義が広く受け入れられている。

> the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task
>
> 新規タスクの効果的な仮説を効率的に見つけ出すために、一つ以上の別のタスクで学習された知識を得て、それを適用する問題

つまり、ある問題を効果的かつ、効率的に解くために、別の関連した問題のデータや学習結果を再利用することが転移学習であることを示している。自然言語処理の分野に限らず、画像処理や音声認識などの研究など幅広い分野で使われる言葉であるため、共変量シフト（covariate shift）、標本選択バイアス、音声認識分野の話者適応なども転移学習のより限定された分野とみなすことができる。本論文では、自然言語処理における転移学習について考えていく。転移学習の詳しい説明は、神嶌 [2] の論文を参照することにする。

例として、IT 関連の雑誌の記事中に出てくる単語の品詞を特定する問題を考えてみる。品詞分類をするための規則は、その雑誌に含まれる各単語の品詞を示した文書を集めたコーパスから、機械学習の手法によって得られる。IT 雑誌記事のコーパスでは、Apple や Mac、CPU、monitor などの単語は名詞であることが多い。したがって、これらの単語から現われた場合、名詞であるという規則が得られる。一般的に、コーパスのデータが多いほど、より正確に品詞の分類をすることができる。ところが、IT 関係の雑誌の記事が見当たらず、経済関係の新聞記事しかコーパスがない状況を考えてみる。それでも、この新聞記事も同じ言語によって書かれているので品詞の推定に役立つと予測できる。しかし、単純に二つのコーパスを混ぜたデータから学習すると問題が起きてしまう。例えば、Apple や Mac、CPU といった単語は、新聞記事でも名詞として現われるが、monitor という単語は『監視する』といった意味の動詞として新聞記事では現われる。このように二つのデータの違いをうまく捉え、CPU のような場合だけに限って新聞記事のデータを活用し、IT 関係の雑誌記事の品詞を特定する規則を獲得したいというように、関連しているが異なる部分も存在するデータから、目的の問題にも利用できる情報・知識だけを抽出し、より精度の高い規則を得ることが転移学習の目標となっている。

現在、Web などから大量のテキストデータを入手することはかなり容易になっており、教示情報が必要な教師無し学習で獲得できる言語モデルなどは精度が向上している。一方、音声認識の音響モデルや文書分類モデルなどは精度が向上していない。原因としては、これらのモデルを題材としたタスクでは、教示情報付きの学習データが必要であり、そのようなデータは人

が手動で作成しなければならない点が挙げられる。こうした教示情報を大量に作成するには、費用、人的資源、時間などの制作コストが莫大にかかり、一般には困難とされている。この問題に対処するための機械学習は三種類あり、半教師付き学習、能動学習、そして転移学習である。そのため転移学習の研究は活発に行われている。

本論文では、自然言語処理の転移学習における領域適応（domain adaptation）についての研究を WSD をタスクに設定して行っている。

3.2 領域適応

3.2.1 概要

従来、自然言語処理のタスクにおける機械学習では、訓練データとテストデータは同じドメインのコーパスから得たデータであることが前提であった。しかし、あるドメインのデータのラベルを識別したいにも関わらず、別のドメインのデータからしか識別規則を学習できないことも多い。そこで、あるドメインの訓練データから識別規則を学習した分類器を、別のドメインのテストデータに適用できるようにチューニングすることを考える。このように、あるドメインで学習させ、作った分類器を違うドメインに適応する手法を領域適応という。識別規則を学習するために利用するデータのドメインをソースドメイン、適応される側をターゲットドメインと呼ぶ。例として、新聞の訓練データから学習した分類器を雑誌のテストデータに適用する流れを図 3.1 に示す。

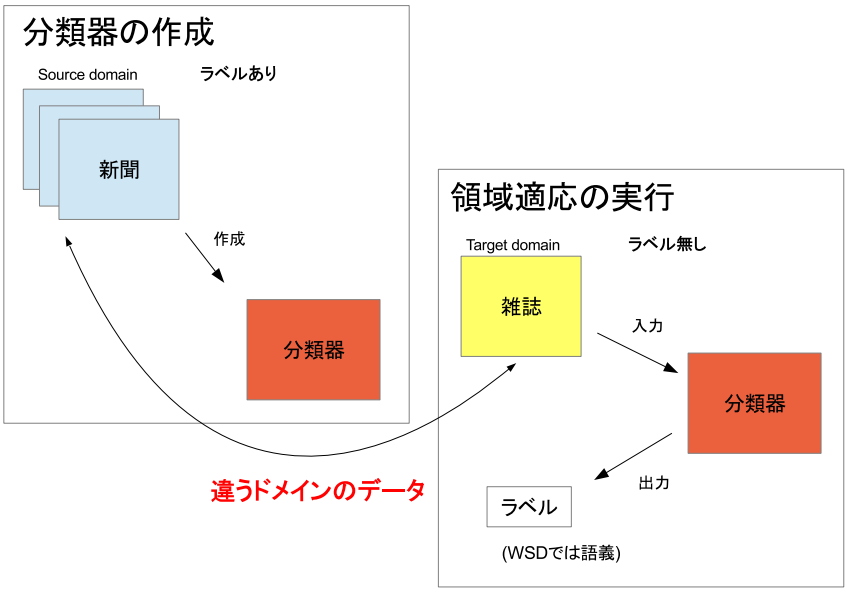

図 3.1: 領域適応時の機械学習

3.2.2 問題点

単語には、多くの意味や正しい意味を識別する過程、または文脈間での意味が存在する。単語のドメインが違う場合、語義の分布も異なることが多い。これは WSD の領域適応の問題点

である。

実際に、ソースドメインの訓練データで分類器 (Naive Bayes 分類器) を学習し、ターゲットドメインのテストデータの語義を推定させた正解率と、ソースドメインとターゲットドメインが同じ場合の正解率を表 3.1 に示す。なお、ソースドメインとターゲットドメインが同じ場合は、One-vs-Other 法の交差検定で分類を行う。

表 3.1 から読み取れるように、ソースドメインとターゲットドメインが同じ場合と異なる場合とを比べると、異なる場合では正解率が下がってしまっていることが分かる。これは前述したように、ドメイン間での語義の比率が異なることによって起こる問題である。これを如何にして向上させるかが領域適応の研究における目的である。

表 3.1: NaiveBayes 分類器による領域適応の正解率

ソースドメイン	ターゲットドメイン	正解率
Yahoo!知恵袋	Yahoo!知恵袋	82.04%
新聞		76.00%
書籍		77.40%
Yahoo!知恵袋	新聞	76.60%
新聞		84.29%
書籍		81.80%
Yahoo!知恵袋	書籍	80.60%
新聞		78.20%
書籍		83.27%

3.3 関連研究

ここで、自然言語処理の分野で行われているさまざまな領域適応の研究のうち一部を紹介する。まず最も重要な研究である (Hal Daumé(2007))[1] の研究についてである。この研究によって自然言語処理における領域適応の研究の枠組みが明確化された。Daumé は、教師付き学習（supervised）の領域適応を行っている。そこでは、ソースデータとターゲットデータを合わせて、素性空間を「ソースデータのみ」「ターゲットデータのみ」「ソースデータとターゲットデータ両方」の三倍にしてから通常の学習を行う実験を行っている。これは様々な教師付き学習に併用することが可能であり効果が高い、加えて実装が簡単である。マルチドメインへの拡張も容易であることも利点に挙げられている。筆者は、(Hal Daumé(2010))[2] において文献 [1] を半教師付き学習 (semi-supervised) のために拡張している。拡張前の利点を引き継いだ上に、ターゲットドメインのラベル無しデータを用いることで性能が向上する手法である。

WSD の領域適応を扱った研究としては古宮の一連の研究が挙げられる [9][10][11][12]。これらは全て教師付き学習に属する研究である。（古宮 (2010)[12]）では、WSD について領域適応を行った場合、最も効果的な領域適応手法はソースデータとターゲットデータの性質により異なるとし [11]、WSD の対象単語タイプ、ソースデータ、ターゲットデータの三つ組を 1 ケースで数え、そのケース毎にデータの性質から最も効果的な領域適応手法を、決定木学習によって自動的に選択する手法について述べるとともに、どのような性質が効果的な領域適応手法の決定に影響を与えたかについて考察している。

また、(Agirre and Lacalle(2008))[3] は、半教師付き学習 (semi-supervised) についての WSD における領域適応を行った。訓練データに、ターゲットドメインのテストデータを加えて行列を作成し、特異値分解 (SVD) により素性圧縮して分類器を学習するというものである。また筆者らは (Agrrie and Lacalle(2009))[4] において同じ手法で教師付き学習 (supervised) の領域適応を行っている。

(Chan and Ng(2006))[5] は、EM アルゴリズムによる Prior(意味の割合) 推定により WSD の領域適応を行っている。また、筆者らは (Chan and Ng (2007))[6] でも、EM アルゴリズムによる Prior 推定を行っているが、ここでは、Active-learning により用例をターゲットドメインから足す教師付き学習 (supervised) の領域適応を行っている。Count-marging により重要文に重みをつけてから推定を行う。

第4章　分類器（classifier）

4.1　概要

自然言語処理の分野では、データをある決められたグループに分けるといった状況に良く出会う。電子メールを例にとると、届いたメールが「業務メール」「プライベートメール」「スパムメール」のどれにあたるのかを判定したいといった状況である。あらかじめ決められたグループにデータを分けることを分類 (classification) と言い、そのグループをクラス (class) と呼ぶ。また、データを用いて何らかのモデルや処理方法を導き出し、作成したものを分類器 (classifier) という。分類器の作り方は大きく分けて、次の 2 通りが考えられる。

- 人間が分類規則を書く方法 (規則ベース手法)
- データから自動的に分類器を作成する方法

ここでは、データから分類器を自動的に作成することを考える。与えられるデータ集合を以下のように仮定する。

$$D = (d^{(1)}, s^{(1)}), (d^{(2)}, s^{(2)}), \cdots, (d^{(|D|)}, s^{(|D|)}) \tag{4.1}$$

ここで、$d^{(1)}, d^{(2)}, \cdots, d^{(|D|)}$ は事例 (文書など)、$s^{(1)}, s^{(2)}, \cdots, s^{(|D|)}$ は事例が属するクラスを意味している。このデータに付与されているクラスをラベル (label) といい、データ集合 D はラベル付きデータ (labeled data) などと呼ばれる。逆にラベルが付いていないデータをラベル無しデータ (unlabeled data) と呼ぶ。このようなデータからモデルや学習規則を導き出すことを学習といい、こうしてできた分類器にはいくつか種類があるが、本論文では、そのうちの 1 つである「ナイーブベイズ分類器」を用いて実験を行っている。

4.2　ナイーブベイズ分類器（Naive Bayes classifier）

ナイーブベイズ分類器（NaiveBayes classifier, NB classifier）は、古くからある分類器で、現在も使われており、適切な使い方をすれば高い性能を発揮することも多い。これは確率に基づいた分類器で、文章 d に対して事後確率 $P(s \mid d)$ が最大となるクラス $s \in S$ を求めることを目的としている。この事後確率の式は、ベイズの公式を利用して以下のように表すことができる。

$$P(s \mid d) = \frac{P(s)P(d \mid s)}{P(d)} \tag{4.2}$$

この右辺が最大となるクラス s に文章 d を分類することになる。ここで分母の $P(d)$ はクラス s には依存しないため、省略することができる。つまり、分子の最大化問題を解けばいい。

$$
\begin{aligned}
s_{\max} &= arg\max_{s} \frac{P(s)P(d \mid s)}{P(d)} \\
&= arg\max_{s} P(s)P(d \mid s) \qquad (4.3)
\end{aligned}
$$

式 4.3 の右辺を求めることが出来れば良い。ここで、$P(s)$ は、以下に示す式で値を求めることが可能である。

$$
P(s) = \frac{\text{クラス } s \text{ に属する文書数}}{\text{総文書数}} \qquad (4.4)
$$

式 4.4 からも分かるように、$P(s)$ の値を計算することは非常に簡単である。

しかし、ここで問題となるのが $P(d \mid s)$ の計算である。この計算は容易ではない。この $P(d \mid s)$ は、文章 d がクラス s のもとで起こる条件付確率である。文章 d において、単語の種類とその組み合わせを考えると、起こりうる文章 d は膨大な数にのぼる。全ての文章 d についてそれぞれがデータの中で何回起こるかを調べて、$P(d \mid s)$ を最尤推定で求めるのは非現実的である。

そこで問題を単純化してみる。ナイーブベイズ分類器では、この文章 d に単純化したモデルを仮定して $P(d \mid s)$ の値を求めている。モデルには以下の 2 種類が存在する。

- 多変数ベルヌーイモデル
- 多項モデル

それぞれについて解説する。

4.2.1 多変数ベルヌーイモデル（Multivariate Bernoulli model）

導入

語彙 V(単語の集合) を考える。多変数ベルヌーイモデルでは、語彙 V に含まれる各単語 w とクラス s について、ベルヌーイ分布に従う確率変数 $X_{w,s}$ を考える。

ベルヌーイ分布とは、ある事象が起こる確率を p、起こらない確率を $q = 1 - p$ として、それをモデル化したものである。つまり、2 つの事象しか現われないものをモデル化した分布と言える。各確率変数は、w が事例内で出現するときに ”1 ”となり、そうでないときに ”0 ”になるとする。各ベルヌーイ分布を規定するパラメータである「$X_{w,s}$ が 1 となる確立」は、$p_{w,s}$ で表すことにする。また、これらのベルヌーイ分布は互いに独立であると仮定する。つまり、s が与えられたときの独立なので、条件付独立である。ここで、わかりやすさのために $p_s = P(s)$ としておく。多変数ベルヌーイモデルは、p_s と $p_{w,s}$ の二つのパラメータで値が決まる。

クラス s が与えられているときに各単語 w が生起するかどうかを表す確率は、

$$
p_{w,s}^{\delta_{w,d}}(1 - p_{w,s})^{1-\delta_{w,d}}
$$

である。ここで、$\delta_{w,d}$ は単語 w が文章 d に出現したときに ”1 ”となり、そうでないときに ”0 ”となる。

文章 d の生起確率を式で表すと、

$$P(d \mid s) = \prod_{w \in V} p_{w,s}^{\delta_{w,d}} (1 - p_{w,s})^{1-\delta_{w,d}} \tag{4.5}$$

となる。よって多変数ベルヌーイモデルにおけるナイーブベイズ分類器は、

$$arg\max_s P(s)P(d \mid s) = arg\max_s p_s \prod_{w \in V} (p_{w,s}^{\delta_{w,d}} (1 - p_{w,s})^{1-\delta_{w,d}}) \tag{4.6}$$

を最大化するような s を出力する問題に落ち着く。こうすることで計算が困難だった $P(d \mid s)$ を $p_{w,s}$ の値の積で表すことが可能となる。$P(d \mid s)$ のときは、単語の組み合わせを考える必要があったが、$p_{w,s}$ は 1 つの単語が生起するかどうかに注目しているため、計算が簡単となる。次に、この $p_{w,s}$ をどのように計算するかを述べていく。

多変数ベルヌーイモデルにおけるパラメータの最尤推定

最尤推定を用いて与えられてデータからパラメータをどのように推定していくかを述べていく。ここで、与えられたデータ D は、式 4.1 に示された形で与えられているとする。

求めるべきパラメータは、p_s と $p_{w,s}$ である。これを最尤推定で求めるので、

$$\begin{aligned}
\log P(D) &= \sum_{(d,s) \in D} \log P(d, s) \\
&= \sum_{(d,s) \in D} \Big(\log p_s \prod_{w \in V} (p_{w,s}^{\delta_{w,d}} (1 - p_{w,s})^{1-\delta_{w,d}}) \Big) \\
&= \sum_{(d,s) \in D} \Big(\log p_s + \sum_{w \in V} (\delta_{w,d} \log p_{w,s} + (1 - \delta_{w,d}) \log (1 - p_{w,s})) \Big) \\
&= \sum_s N_s \log p_s + \sum_s \sum_{w \in V} N_{w,s} \log p_{w,s} + \sum_s \sum_{w \in V} (N_s - N_{w,s}) \log (1 - p_{w,s})
\end{aligned}$$

を最大化することになる。ここで、未定義の変数についての解説を加える。

$$N_{w,s} \quad : \quad \text{クラス } s \text{ であり、} w \text{ を含むような訓練文書数} \tag{4.7}$$

$$N_s \quad : \quad \text{クラス } s \text{ であるような訓練文書数} \tag{4.8}$$

各文書は複数の異なる単語を含みえるので、$N_s = \sum_w N_{w,s}$ は通常は成立しない。

なお、p_s については制約があるため、この最大化問題は、つぎのような制約付き条件最適化問題で表すことが出来る。

$$\begin{aligned}
max. \quad & \log P(D) \\
s.t. \quad & \sum_s p(s) = 1
\end{aligned}$$

この問題は、ラグランジュの未定乗数法（束縛条件の下で最適化を行う方法）により解くことが可能である。未定乗数 λ を導入して、次のようにラグランジュ関数 $L(\theta, \lambda)$ を定義する。

$$L(\theta, \lambda) = \log P(D) + \lambda\Big(\sum_s p_s - 1\Big) \tag{4.9}$$

ただし、θ は求めたいパラメータの集合 $(p_{w,s_{w\in V, s\in S}}, p_{s_{s\in S}})$ である。各パラメータに関する偏微分を計算してみると、

$$\frac{\delta L(\theta,\lambda)}{\delta p_{w,s}} = \frac{N_{w,s}}{p_{w,s}} - \frac{N_s - N_{w,s}}{1 - p_{w,s}}, \quad \frac{\delta L(\theta,\lambda)}{\delta p_s} = \frac{N_s}{p_s} + \lambda$$

となる。これらをそれぞれ ”0 ”として、$\sum_s p_s = 1$ と合わせて考えると、

$$p_{w,s} = \frac{N_{w,s}}{N_s}, \quad p_s = \frac{N_s}{\sum_s N_s} \tag{4.10}$$

となる。つまり、

$$p_{w,s} = \frac{\text{クラス } s \text{ に属する訓練文書のうち } w \text{ を含む文書数}}{\text{クラス } s \text{ に属する訓練文書数}}, \tag{4.11}$$

$$p_s = \frac{\text{クラス } s \text{ に属する訓練文書数}}{\text{訓練文書数}} \tag{4.12}$$

と推定していることとなる。多変数ベルヌーイモデルでは、パラメータ p_s、$p_{w,s}$ の推定に文書数を用いていることが分かる。

4.2.2 多項モデル（multinominal model）

導入

次に多項モデルの解説をする。多変数ベルヌーイモデルでは、各単語が生起するかしないかをモデル化した。多項モデルでは、文書中の各位置についてどんな単語が起こるかをモデル化する。

文書 d 内の単語数を $|\, d \,|$ で表すとする。多項モデルでは、語彙 V の中から 1 つの単語を選ぶ操作を $|\, d \,|$ 回繰り返すことで文書を作成する。つまり、$P(d \mid s)$ を多項分布でモデル化することとなる。

クラスが s であるとき、単語 w が選ばれる確率を $q_{w,s}$ で表す。つまり、W を単語とする確率変数、S をクラスの値とする確率変数とすると、$q_{w,s} = P(W = w|, S = s)$ である。文書 D 内で、単語 w がそれぞれ $n_{w,d}$ 回起こる確率は、次のように表すことが出来る。

$$\frac{\Big(\sum_w n_{w,d}\Big)!}{\prod_{w\in V} n_{w,d}!} \prod_{w\in V} q_{w,s}^{n_{w,d}}$$

$|\, d \,| = \sum_w n_{w,d}$ である。ただし、厳密には試行する回数を決定しなければならない。ここで、文書の長さはクラスに依存しないという仮定を設けて考えると、

$$p(d \mid s) = P\Big(K = \sum_w n_{w,d}\Big) \frac{\Big(\sum_w n_{w,d}\Big)!}{\prod_{w \in V} n_{w,d}!} \prod_{w \in V} q_{w,s}^{n_{w,d}} \tag{4.13}$$

となる。K は、文書の長さを表す確率変数であり、$P(K = \sum_w n_{w,d})$ は長さが $\sum_w n_{w,d}$ であるような文書が起こる確率である。よって、多項モデルにおけるナイーブベイズ分類器は、

$$P(s)P(d \mid s) = p_s P(\sum_w n_{w,d}) \frac{\Big(\sum_w n_{w,d}\Big)!}{\prod_{w \in V} n_{w,d}!} \prod_{w \in V} q_{w,s}^{n_{w,d}} \tag{4.14}$$

を最大化するような s を出力する。なお、$p_s = P(s)$ である。

$$\begin{aligned} P(s)P(d \mid s) &= arg\max_s p_s P(\sum_w n_{w,d}) \frac{\Big(\sum_w n_{w,d}\Big)!}{\prod_{w \in V} n_{w,d}!} \prod_{w \in V} q_{w,s}^{n_{w,d}} \\ &= arg\max_s p_s \prod_{w \in V} q_{w,s}^{n_{w,d}} \end{aligned} \tag{4.15}$$

であるので、最大となる s を見つけるためには、$p_s \prod_{w \in V} q_{w,s}^{n_{w,d}}$ さえ分かればいい。

多項モデルにおけるパラメータの最尤推定

データ D が式 4.1 の形で与えられているとして、多項モデルにおけるパラメータの推定の仕方を述べていく。求めるべきパラメータは、p_s と $q_{w,s}$ であるので。$\mid d \mid = \sum_w n_{w,d}$ と表すと、最尤推定により、

$$\begin{aligned} \log P(D) &= \sum_{(d,s) \in D} \log P(d, s) \\ &= \sum_{(d,s) \in D} \log \Big(\frac{P(\mid d \mid) \mid d \mid!}{\prod_{w \in V} n_{w,d}} p_s \prod_{w \in V} q_{w,s}^{n_{w,d}} \Big) \\ &= \sum_{(d,s) \in D} \log \frac{P(\mid d \mid) \mid d \mid!}{\prod_{w \in V} n_{w,d}!} + \sum_{(d,s) \in D} N_s \log p_s + \sum_{(d,s) \in D} \sum_{w \in V} n_{w,d} \log q_{w,s} \\ &= \sum_{(d,s) \in D} \log \frac{P(\mid d \mid) \mid d \mid!}{\prod_{w \in V} n_{w,d}!} + \sum_s N_s \log p_s + \sum_{(d,s) \in D} \sum_{w \in V} n_{w,d} \log q_{w,s} \end{aligned}$$

を最大化することになる。なお、N_s は、多変数ベルヌーイモデルを解説する際に定義した式 4.8 と同じである。

多項モデルにおいては、$\sum_{s \in S} p_s = 1$ となる制約に加えて、任意の s について $\sum_{w \in V} q_{w,s} = 1$ となる制約がある。したがって、この最大化問題は、次のような制約付き最適化問題で表すことが出来る。

$$\begin{aligned} max. \quad & \log p(D) \\ st. \quad & \sum_{s \in S} p_s = 1 \\ & \sum_{w \in V} q_{w,s} = 1; \forall s \in S \end{aligned}$$

これも多変数ベルヌーイモデル同様、ラグランジュ未定乗数法により解くことが可能である。未定乗数を $\beta s_{s\in S}$、γ を導入して（簡単のために $\beta s_{s\in S}$ を β で表す）、次のようにラグランジュ関数 $L(\theta, \beta, \gamma)$ を定義する。

$$L(\theta, \beta, \gamma) = \log P(D) + \sum_{s\in S} \beta_s \Big(\sum_{s\in S} q_{w,s} - 1 \Big) + \gamma \Big(\sum_{s\in S} p_s - 1 \Big)$$

ここで、θ は求めたいパラメータの集合 $q_{w,s}{}_{w\in V, s\in S}$, $p_s{}_{s\in S}$ である。p_s やそれに対応する γ は多変数ベルヌーイモデルの場合と同じである。等式制約付きの凸計画問題（最大化問題）に対するラグランジュ未定乗数法では、$q_{w,s}$ に関する偏微分が 0 になれば良い。

これを偏微分すると、

$$\frac{\partial L(\theta, \beta, \gamma)}{\partial q_{w,s}} = \frac{n_{w,s}}{q_{w,s}} + \beta_s$$

となり、これが 0 となればよいので、$\sum_{w\in V} = 1$ と合わせて考えると、

$$q_{w,s} = \frac{n_{w,s}}{\sum_w n_{w,s}}$$

が得られる。なお、p_s に関しては多変数ベルヌーイモデルで述べた式 4.12 と同じである。つまり、$q_{w,s}$ は以下のように表すことが出来る。

$$q_{w,s} = \frac{(\text{クラス } s \text{ に属する訓練文書全体での } w \text{ の出現回数})}{(\text{クラス } s \text{ に属する訓練文書全体での全単語の出現回数})}$$

以上のように推定していることとなる。

4.2.3 2 つのモデルの比較

簡単にまとめると以下のように、多変数ベルヌーイモデルでの p_s と $p_{w,s}$、多項モデルでの p_s と $q_{w,s}$ を以下のように定義していることとなる。

$$p_s = \frac{\text{クラス } s \text{ に属する訓練文書数}}{\text{訓練文書数}}, \tag{4.16}$$

$$p_{w,s} = \frac{\text{クラス } s \text{ に属する訓練文書のうち } w \text{ を含む文書数}}{\text{クラス } s \text{ に属する訓練文書数}} \tag{4.17}$$

多変数ベルヌーイモデルにおけるパラメータ推定

$$p_s = \frac{\text{クラス } s \text{ に属する訓練文書数}}{\text{訓練文書数}}, \tag{4.18}$$

$$q_{w,s} = \frac{(\text{クラス } s \text{ に属する訓練文書全体での } w \text{ の出現回数})}{(\text{クラス } s \text{ に属する訓練文書全体での全単語の出現回数})} \tag{4.19}$$

多項モデルにおけるパラメータ推定

2つのモデル間で p_s の定義は同じであるが、$p_{w,s}$ と $q_{w,s}$ の定義は異なる。ここが2つのモデルの違いである。多変数ベルヌーイモデルでは、単語 w が文書 d で生起したか否かが分類に影響を与えていたのに対して、多項モデルでは、単語 w が文書 d で生起した回数が分類に影響を与えている。また、多変数ベルヌーイモデルでは、生起しなかった単語があった場合は $1 - p_{w,s}$ を考慮して計算した。一方、多項モデルでは、生起しなかった単語については無視している。多変数ベルヌーイモデルは生起しなかったことをモデルに取り入れているのに対して、多項モデルは生起した単語にだけ着目しているのがわかる。

4.2.4 ゼロ頻度問題の処理（スムージング）

ここで、ゼロ頻度問題について言及する。確率的言語モデルにおいて、ある単語 w の生起頻度によって出現確率を求めるとき、学習データに存在しない単語の出現確率は ”0 ”となってしまう。この問題をゼロ頻度問題といい、これを解消する方法をスムージング (smoothing) と呼ぶ。具体的には、最大事後確率推定という、0.00 に近い値となる確率が非常に小さいような事前分布を与えて確率分布を均すことを行う。多変数ベルヌーイモデルと多項モデルでは、ディリクレ分布を与えることで、ゼロ頻度問題を解消することが出来る。ここで、最大事後確率推定とディリクレ分布について解説する。

最大事後確率推定（Maximum A posteriori estimation）

最大事後確率推定（maximum a posteriori estimation）は、その頭文字をとって MAP 推定（MAP estimation）とも呼ばれている。あらかじめパラメータがどのような値をとりやすいかが分かっている場合、具体的には、パラメータ θ の確率分布 $P(\theta)$ が分かっている場合である。これをパラメータの事前確率分布 (prior distribution) と呼ぶ。一方、データ D が与えられたときのパラメータ θ の確率分布 $P(\theta|D)$ を事後確率分布 (posterior distribution) と呼ぶ。MAP 推定では、事後確率 $P(\theta|D)$ が最大になるようにパラメータを決定する。

事後確率の最大化は次のような式に変形することができる。

$$\begin{aligned} arg\max_{\theta} P(\theta \mid D) &= arg\max_{\theta} \frac{p(\theta \cdot P(D \mid \theta)}{P(D)} && (4.20) \\ &= arg\max_{\theta} P(\theta) \cdot P(D \mid \theta) && (4.21) \end{aligned}$$

なお、式 4.20 の分母 $P(D)$ は最大化に関係ないので式 4.21 にて省略した。ここで、計算の簡単化のために対数をとる。

$$\log P(\theta) \cdot P(D \mid \theta) = \log P(\theta) + \sum_{x^{(i)} \in D} \log P(x^{(i)} \mid \theta) \quad (4.22)$$

これを最大化する θ を選ぶこととなる。

ディリクレ分布（Dirichlet distribution）

ディリクレ分布は、$x_i \leq 0$ 、$\sum_i x_i = 1$ であるような $x = (x_1, x_2, \cdots, x_n)$ に対して確率を与える分布である。確率密度関数は、

$$p(\boldsymbol{x};\alpha) = \frac{1}{\int \prod_i x_i^{\alpha_i - 1} dx} \prod x_i^{\alpha_i - 1}$$

である。$\alpha_1, \cdots, \alpha_n$ という n 個のパラメータを持っている。大事な点は、確率密度が $\prod_i x_i^{\alpha_i - 1}$ に比例している点である。分母は積分すると ”1 ”になるように導入されている。

ディリクレ分布の性質を理解するため、$x = (x_1, x_2)$ として二次元ディリクレ分布を考える。パラメータ α については、$\alpha_1 = 2, \alpha_2 = 2$ とする。$\sum_i x_i = 1$ という制限から、$x_2 = 1 - x_1$ である。このとき、

$$p(\boldsymbol{x};\alpha) = \frac{1}{\int_0^1 x_1(1 - x_1) dx_1} x_1(1 - x_1)$$

である。また、分母は、

$$\int_0^1 x_1(1 - x_1) dx_1 = \Big[\frac{1}{2}x_1^2 - \frac{1}{3}x_1^3\Big]_0^1 = \frac{1}{6}$$

であるので、

$$p(\boldsymbol{x};\alpha) = 6x_1(1 - x_1)$$

となる。x 軸を横にとってこのグラフを図示したものを図 4.1 に示す。

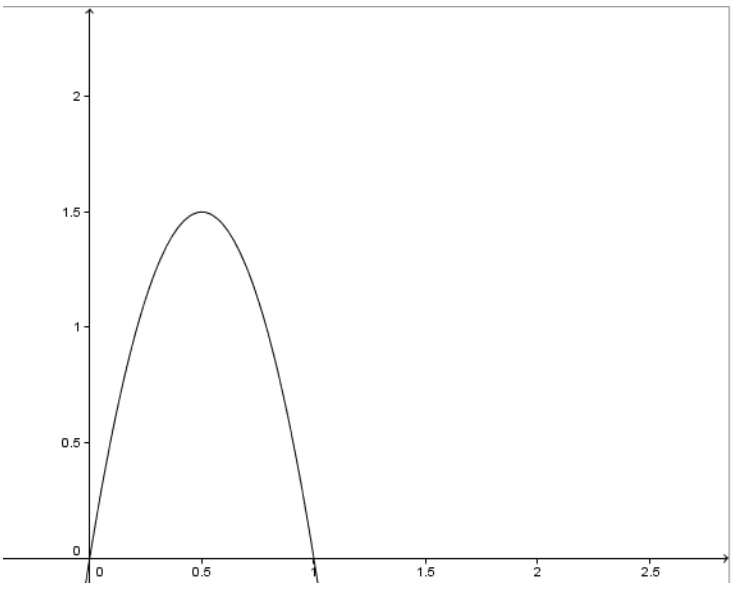

図 4.1: 二次元ディリクレ分布のグラフ

図 4.1 のグラフからも分かるように、中央が盛り上がっているグラフである。つまり、x_1 と x_2 の値が近いような x は比較的高い確率を保有しているが、$x_1 = 0.99$、$x_2 = 0.01$ のように

偏った x は非常に低い確率を持つ。いずれかの x_i が 0 もしくは 1 に近づくような x は確率が非常に小さくなる。このように、ディリクレ分布に従う確率変数は極端な値をとりにくいことで知られている。

この分布をナイーブベイズ分類器における二つのモデルに与えることで確率分布をならし、極端な値に対して補正を行う。

多変数ベルヌーイモデルにおける MAP 推定

多変数ベルヌーイモデルでは、式からもわかるように、$p_{w,s}$ の値を積の形で表している。このため、ソースドメインで出現しない単語がターゲットドメインに出現した場合、値が 0 となってしまう。この値が 0 とならないようにパラメータを推定することを考えていく。ここでディリクレ分布を与えることとなる。

多変ベルヌーイモデルにおける MAP 推定の目的関数となるのは、

$$
\begin{aligned}
& \log P(\theta) + \log P(D) \\
= \; & \log\Big(\prod_s p_s^{\alpha-1}\Big) \times \Big(\prod_{w,s} (p_{w,s}^{\alpha-1}(1-p_{w,s})^{\alpha-1})\Big) + \sum_{(d,s)\in D} \log P(d,s) + (\text{定数}) \\
= \; & (\alpha-1)\sum_s \log p_s + (\alpha-1)\sum_{w,s} (\log p_{w,s} + \log(1-p_{w,s})) \\
& + \sum_{(d,s)\in D} \log\Big(p_s \prod_{w\in V} (p_{w,s}^{\delta_{w,d}}(1-p_{w,s})^{1-\delta_{w,d}})\Big) + (\text{定数})
\end{aligned}
$$

であり、これを $\sum_s p(s) = 1$ となる制約のもとで最大化する。

最尤推定の場合とほとんど同じように計算が出来る。ラグランジュ関数は、

$$
L(\theta,\lambda) = \log P(\theta) + \log P(D) + \lambda\Big(\sum_s p_s - 1\Big)
$$

となる。偏微分を計算してみると、

$$
\begin{aligned}
\frac{\partial L(\theta,\lambda)}{\partial p_s} &= \frac{(\alpha-1)}{p_{w,s}} - \frac{(\alpha-1)}{1-p_{w,s}} + \frac{N_{w,s}}{p_{w,s}} - \frac{N_s - N_{w,s}}{1-p_{w,s}}, \\
\frac{\partial L(\theta,\lambda)}{\partial p_s} &= \frac{(\alpha-1)}{p_s} + \frac{N_s}{p_s} + \lambda
\end{aligned}
$$

となる。これをそれぞれ 0 とおき、$\sum_s p_s = 1$ と合わせると、

$$
p_{w,s} = \frac{N_{w,s} + (\alpha-1)}{N_s + 2(\alpha-1)}, \quad p_s = \frac{N_s + (\alpha-1)}{\sum_s N_s + \mid C \mid (\alpha-1)}
$$

が得られる。ここで、$\mid C \mid$ はクラスの数を表す。

たとえば、$\alpha = 2$ のときを考えてみると、

$$
p_{w,s} = \frac{N_{w,s} + 1}{N_s + 2}, \quad p_s = \frac{N_s + 1}{\sum_s N_s + \mid C \mid}
$$

となる。これは、すべての w と s に対して、生起回数に 1 を足して、パラメータを計算していることとなる。生起回数に 1 を足す手法は一般的に用いられる手法であり、ラプラススムージング (Laplace Smoothing) と呼ばれている。

多項モデルにおける MAP 推定

多項モデルにおいてもディリクレ分布で確率分布を均すことを考える。MAP 推定の目的関数となるのは、

$$\begin{aligned} & \log P(\theta) + \log P(D) \\ = \ & \Big(\prod_s p_s^{\alpha-1}\Big) \times \Big(\prod_{w,s} q_{w,s}^{\alpha-1}\Big) + \sum_{(d,s)\in D} \log P(d,s) + (\text{定数}) \\ = \ & (\alpha-1)\Big(\sum_s \log p_s + \sum_{w,s} \log q_{w,s}\Big) + \sum_{(d,s)\in D)} \log\Big(\frac{P(\mid d \mid) \mid d \mid!}{\prod_{w\in V} n_{w,d}!} p_s \prod_{w\in V} q_{w,s}^{n_{w,s}}\Big) + (\text{定数}) \end{aligned}$$

であり、これを $\sum_s p(s) = 1$、$\sum_w q_{w,s} = 1$ の制約の元で最大化する。

最尤推定の場合と同様にラグランジュ乗数 β と γ を導入し、ラグランジュ関数を、

$$L(\theta,\beta,\gamma) = \log P(\theta) + \log P(D) + \sum_{s\in S} \beta_s\Big(\sum_{w\in V} q_{w,s} - 1\Big) + \gamma\Big(\sum_s p_s - 1\Big)$$

となる。p_s と γ に関しては、多変数ベルヌーイモデルと同様である。$q_{w,s}$ について偏微分を計算してみると、

$$\frac{\partial L)(\theta,\beta,\gamma)}{\partial q_{w,s}} = \frac{(\alpha-1)}{q_{w,s}} + \frac{n_{w,s}}{q_{w,s}} + \beta_s$$

となる。これを 0 とおき、$\sum_{w\in V} q_{w,s} = 1$ と合わせて考えると、

$$q_{w,s} = \frac{n_{w,s} + (\alpha-1)}{\sum_w n_{w,s} + \mid W \mid (\alpha-1)}$$

が得られる。ここで $\mid W \mid$ は単語の種類を表している。

例えば、$alpha = 2$ を考えてみると、

$$q_{w,s} = \frac{n_{w,s}+1}{\sum_w n_{w,s} + \mid W \mid}, \quad p_s = \frac{N_s + 1}{\sum_s N_s + \mid C \mid}$$

となる。多変数ベルヌーイモデルと同じようにすべての生起回数に 1 を加えて計算を行っていることとなる。

4.2.5 アンダーフロー対策

プログラムでナイーブベイズ分類器を実装する場合、注意しなければならいな点が 1 つある。多変数ベルヌーイモデルでの $p_{w,s}$ と多項モデルでの $q_{w,s}$ は非常に小さな値であり、なおかつ文書中にはたくさんの単語が出現するため乗算がアンダーフローを起こしてしまう可能性が出てくる。この問題を解決するために対数をとることで対処する。多変数ベルヌーイモデルの式 4.6 を変形すると、

$$arg\max_s p_s \prod_{w\in V} (p_{w,s}^{\delta_{w,d}}(1-p_{w,s})^{1-\delta_{w,d}}) = arg\max_s \Big(\log p_s \sum_{w\in V} \log p_{w,s}^{\delta_{w,d}}(1-p_{w,s})^{1-\delta_{w,d}}\Big)$$

となる。多項モデルの式 4.15 を変形すると、

$$arg\max_s \log\Big(p_s \prod_{w\in V} q_{w,s}^{n_{w,d}}\Big) = arg\max_s \Big(\log p_s \sum_{w\in V} \log q_{w,s}^{n_{w,d}}\Big)$$

となる。こうすることでアンダーフローを防ぐことが出来る。

4.3 サポートベクトルマシン (Support Vector Machine)

サポートベクトルマシン（Support Vector Machine, SVM）は、線形二値分類器であり、クラス数が 2 つであるような問題に用いられる。2 つのクラスは、それぞれ正クラス（positive class）と負クラス（negative class）と呼ばれている。正クラスに属している事例を正例（positive example）、負クラスに属している事例を負例（negative example）という。訓練データが、

$$D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(|D|)}, y^{(|D|)})\} \tag{4.23}$$

の形で与えられているとする。$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(|D|)}$ は事例の素性ベクトルを表しており、$y^{(1)}, y^{(2)}, \cdots, y^{(|D|)}$ は事例のクラスラベルを表している。正例のクラスラベルは +1、負例のクラスラベルは −1 である。線形分類器であるので、分離平面の方向ベクトル $\boldsymbol{w}$ と切片 b をパラメータとして、SVM は、

$$f(\boldsymbol{x}) = \boldsymbol{w}\cdot\boldsymbol{x} - b \tag{4.24}$$

という関数を用いて表される。事例 $\boldsymbol{x}$ を $f(\boldsymbol{x}) \le 0$ ならば正クラス、$f(\boldsymbol{x}) < 0$ ならば負クラスに分類する。

4.3.1 マージン最大化

上で述べたパラメータ $\boldsymbol{w}, b$ の求め方を説明する。訓練データが図 4.2 のように分布しているとする。この分布を分類する平面を考えていく。この平面のことを分離平面と呼ぶ。方向ベクトルを $\boldsymbol{w}$、切片を b で表すと、分離平面は $\boldsymbol{w}\cdot\boldsymbol{x} = b$ を満たす点 $\boldsymbol{x}$ の集合となる。

ここでポイントとなるのが、どちらのクラスからもなるべく遠い位置で分ける平面を考えることである (図 4.3)。この方法をマージン最大化（margin maximization）と呼ぶ。分離平面のマージンとは、最も近い訓練事例との距離と定義されている。

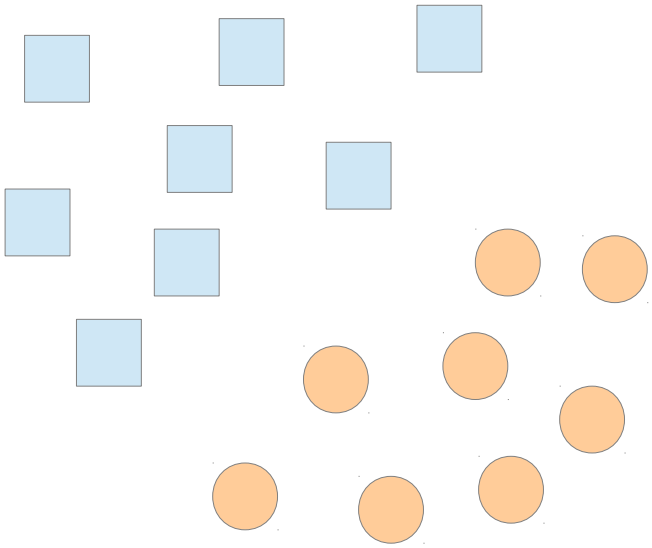

図 4.2: 訓練データの分布

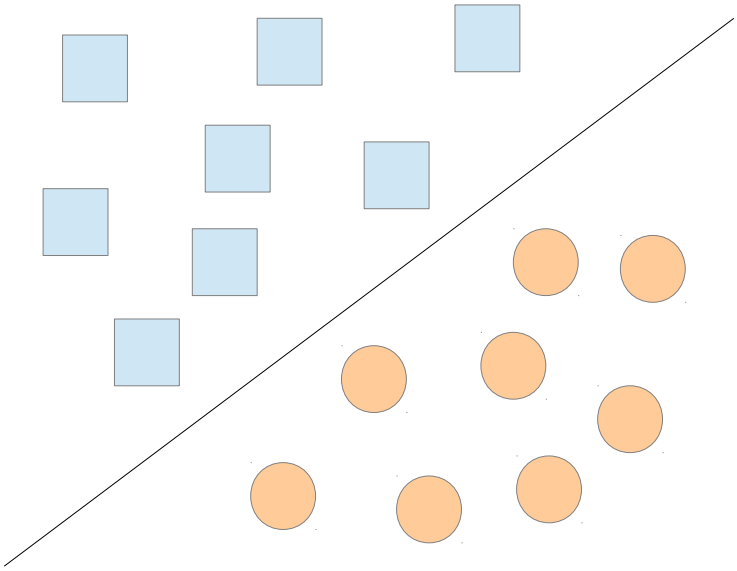

図 4.3: 分離平面の例

分離平面に最も近い正例を $\boldsymbol{x}_+$ で表すことにする。また $\boldsymbol{x}_+$ と分離平面を結ぶ垂線の足を x_* で表す。すなわち、マージンは $|\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ |$ で表すことができる。$\boldsymbol{w}$ と $\boldsymbol{x}_+ - \boldsymbol{x}_*$ は同じ方向を向いているので、$\boldsymbol{w} \cdot (\boldsymbol{x}_+ - \boldsymbol{x}_*) = |\ \boldsymbol{w}\ ||\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ |$ が成り立つ。

ここで、分離平面 $\boldsymbol{w} \cdot \boldsymbol{x} = b$ は、式全体を定数倍しても変わらない。逆に言えば、うまく定数倍することができれば、$\boldsymbol{w} \cdot \boldsymbol{x}_+ - b = 1$ とすることができる。また、x_* は、分離平面上にあるので、当然ながら $\boldsymbol{w} \cdot \boldsymbol{x}_* = b$ である。よって、

$$\begin{aligned} \boldsymbol{w} \cdot (\boldsymbol{x}_+ - \boldsymbol{x}_*) &= \boldsymbol{w} \cdot \boldsymbol{x}_+ - \boldsymbol{w} \cdot \boldsymbol{x}_* \\ &= (b+1) - b = 1 \end{aligned} \tag{4.25}$$

となる。$\boldsymbol{w} \cdot (\boldsymbol{x}_+ - \boldsymbol{x}_*) = |\ \boldsymbol{w}\ ||\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ |$ と合わせると、$|\ \boldsymbol{w}\ ||\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ | = 1$ であることがわかる。したがって、

$$|\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ | = \frac{1}{|\ \boldsymbol{w}\ |} \tag{4.26}$$

を導き出せる。ここで、$|\ \boldsymbol{x}_+ - \boldsymbol{x}_*\ |$ はマージンを表しているので、結局、分離平面のマージンは、$1/|\ \boldsymbol{w}\ |$ で表されることとなる。これを最大化すればよい。計算の簡単化のために、値を二乗し、逆数をとった $\boldsymbol{w}^2$ を最小化する問題に置き換えて考える。

4.3.2 厳密制約下の SVM モデル

訓練事例を正しく分類することを考えていく。$y^{(i)} = +1$ であるような訓練事例は $\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - b \leq 1$ であれば良い。$y^{(i)} = -1$ の訓令事例は $\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - b \geq 1$ であれば良い。この 2 つの条件をまとめると、

$$y^{(i)}(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - b) \leq 1$$

よって、これを制約とした最適化問題を解くこととなる。

$$\begin{aligned} min. \quad & \frac{1}{2}\boldsymbol{w}^2 \\ s.t. \quad & y^{(i)}(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - b) - 1 \leq 0;\ \forall i. \end{aligned}$$

ここで、目的関数につけた係数 1/2 は計算を分かりやすくするためである。これは省略してもかまわない。

この不等式制約付き最適化問題は、凸計画化問題であるので、ラグランジュ法を用いて解くこととする。ラグランジュ未定乗数 $\alpha_i (\leq 0)$ を導入すると、ラグランジュ関数は、

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^2 - \sum_i \alpha_i \Big(y^{(i)}(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - b) - 1 \Big)$$

と表せる。これをそれぞれのパラメータで偏微分する。

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \alpha) = \boldsymbol{w} - \sum_i \alpha_i y^{(i)} \boldsymbol{x}^{(i)}, \tag{4.27}$$

$$\frac{\partial L(\boldsymbol{w}, b, \alpha)}{\partial b} = \sum_i \alpha_i y^{(i)} \tag{4.28}$$

となる。これらを 0 とおいて、

$$\boldsymbol{w}^* = \boldsymbol{w} - \sum_i \alpha_i y^{(i)} \boldsymbol{x}^{(i)}, \tag{4.29}$$

$$\sum_i \alpha_i y^{(i)} = 0 \tag{4.30}$$

を得る。1 つ目の式 4.29 は、分離平面の方向ベクトル $\boldsymbol{w}^*$ は訓練事例ベクトルの線形和で表されることを意味している。この式 $\boldsymbol{w}^* = \sum_i \alpha_i y^{(i)} \boldsymbol{x}^{(i)}$ を分離平面の式 $\boldsymbol{w} \cdot \boldsymbol{x} = b$ に代入すると、

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x} - b \tag{4.31}$$

となる。これで α_i と b を求められれば、分離平面を得ることができる。

そこで、これらの式を元のラグランジュ関数に代入することを考える。まず、$\boldsymbol{w}^* = \sum_i \alpha_i y - (i)\boldsymbol{x}^{(i)}$ を用いると、

$$L(\boldsymbol{w}^*, b, \alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} \tag{4.32}$$

$$- \sum_i \alpha_i y^{(i)} \Big(\sum_j \alpha_j y^{(j)} \boldsymbol{x}^{(j)} \cdot \boldsymbol{x}^{(i)} - b \Big) + \sum_i \alpha_i \tag{4.33}$$

$$= -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} + b \sum_i \alpha_i y^{(i)} + \sum_i \alpha_i \tag{4.34}$$

と変形できる。ここで 2 つ目の式 4.30 より $\sum_i \alpha_i y^{(i)} = 0$ なので、

$$\boldsymbol{w}^*, b, \alpha) = -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} + \sum_i \alpha_i \tag{4.35}$$

となる。これで元々の変数 $\boldsymbol{w}$ と b がラグランジュ関数から消去された。

ここでラグランジュ関数の鞍点（saddle point）について説明を加える。ラグランジュ関数 $L(\boldsymbol{x}, \lambda)$ において、不等式 $L(\boldsymbol{x}^*, \lambda^*) \geq L(\boldsymbol{x}^*, \lambda^*) \geq L(\boldsymbol{x}, \lambda)$ を満たす点 $L(\boldsymbol{x}^*, \lambda^*)$ のことを鞍点という。これは $\boldsymbol{x}$ に関しては最大となり、λ に関しては最小となる点のことである。

今回のラグランジュ関数に置き換えて考えてみると、$L(\boldsymbol{w}, b, \alpha)$ を最大化する α_i を求めれば良い。ただし、$\sum_i \alpha_i y^{(i)} = 0, \alpha_i \leq 0$ の制約の下での最大化である。

最適解 α_i^* が求まれば、$\boldsymbol{w}^*$ が求まり、$\boldsymbol{x}_+$ を用いて、$b = \boldsymbol{w}^* \cdot \boldsymbol{x}_+ - 1$ として切片も求まる。

4.3.3 緩和制約下の SVM モデル

上記で導出した SVM は、実際のデータに対してはなかなかうまくいかない。原因としては、全ての訓令事例を正確に分類しなければならないと言う制約 $y(i)(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}-b)$ があるからである。訓練データの中には例外的な事例が存在することがあり得る。こうした事例によって分類平面が大きく影響を受けてしまう。極端なケースでは、訓練データが線形関数でうまく分類できずに、制約を満たす解が存在しないことにもなってしまう。

そこで、制約を少し緩めて考えることにする。新たな変数 $\xi_i(\leq 0)$ を導入して、

$$y^{(i)}(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}-b)-1\leq\xi_i \tag{4.36}$$

という制約に書き換える。ξ_i は、i 番目の訓練事例がうまく分けられない度合いを表す。つまり、x_i が小さいほうが良い。そこで、これを目的関数に加えることにする。新しい最適化問題はつぎのようになる。

$$min.\quad \frac{1}{2}\boldsymbol{w}^2+C\sum_i\xi_i \tag{4.37}$$

$$s.t.\quad y^{(i)}(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}-b)\leq 1-\xi_i;\ \forall i \tag{4.38}$$

ここで、C は正の定数であり、この値が大きいほどきちんと分類できるようになる。逆にこの値が小さいと例外的な訓練事例をほぼ無視した分類器ができる。

この最適化問題をラグランジュ h 法を用いて解いてみる。ラグランジュ未定乗数 $\alpha_i(\leq 0)$ を導入すると、ラグランジュ関数は、

$$\begin{aligned}L(\boldsymbol{w},b,\xi,\alpha,\beta) &= \frac{1}{2}\boldsymbol{w}^2+C\sum_i\xi_i\\ &\quad -\sum_i\alpha_i\Big(y^{(i)}(\boldsymbol{w}\boldsymbol{x}^{(i)}-b)-1+\xi_i\Big)-\sum_i\beta_i\xi_i\end{aligned}$$

と表せる。これをそれぞれのパラメータで偏微分する。$\boldsymbol{w}$ と b については厳密制約下の場合と同じで、$\boldsymbol{w}=\sum_i\alpha_i y^{(i)}\boldsymbol{x}^{(i)}$ と $\sum_i\alpha_i y^{(i)}=0$ が得られる。ξ_i に関しては、

$$\frac{\partial L(\boldsymbol{w},b,\xi,\alpha,\beta)}{\partial\xi_i}=C-\alpha_i-\beta_i \tag{4.39}$$

であるので、$C=\alpha_i+\beta_i$ を得る。双対ラグランジュ関数は、

$$\begin{aligned}L(\boldsymbol{w}^*,b,\xi,\alpha,\beta) &= -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(j)}+\sum_i\alpha_i\\ &\quad +C\sum_i\xi_i-\sum_i\alpha_i\xi_i-\sum_i\beta_i\xi_i\end{aligned}$$

となるが、最後の 3 つの項は $C=\alpha_i+\beta_i$ を使うことで消去できるため、結局は厳密制約下の場合と同じになる。これを $\alpha_i\leq 0,\beta_i\leq 0,\xi_i\leq 0$ なる条件の下で最大化する。ただし、まず ξ_i は双対ラグランジュ関数に登場しないため、とりあえずは放っておく。また、β_i も双対ラグラ

ンジュ関数には登場しないので、$\beta_i = C - \alpha_i$ が満たされていれば良い。結局、$\alpha_i \leq 0$ と合わせて、$0 \geq \alpha_i \geq C$ の条件の下で、

$$L(\boldsymbol{w}^*, b, \xi, \alpha, \beta) = -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(j)} + \sum_i \alpha_i$$

を最大化することとなる。

b は $0 < \alpha_i < C$ の事例を 1 つ持ってきて、$b = \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} - y^{(i)}$ とすることで計算できる。

4.3.4 関数距離

SVM は事例 $\boldsymbol{x}$ を、$f(\boldsymbol{x}) \leq 0$ なら正クラスに、$f(\boldsymbol{x}) < 0$ なら負クラスに分類する。この $f(\boldsymbol{x})$ を関数距離（functional distance）と呼ぶ。

分類する際、$f(\boldsymbol{x}) = 0.00001$ である $\boldsymbol{x}$ も、$F(\boldsymbol{x}) = 1000$ である $\boldsymbol{x}$ も正クラスに分類される。これはこれで良いが、前者のように関数距離 $f(\boldsymbol{x})$ が 0 に非常に近いものは、その分類結果が誤りであることが多い。また後者のように関数距離 $f(\boldsymbol{x})$ が非常に大きいものは、実際に正クラスに属していることが多い。負クラスについても同じことが言える。

このように、関数距離は分類結果の信頼度となっている。つまり、関数距離が大きい事例だけを集めれば、高い確率で正例となる事例を集めることができる。また関数距離が 0 に非常に近い事例を集めれば、分類結果が誤りである事例を集めることができる。誤り発見の重要な技術であると同時に、エラー分析の際にも利用できる。

第5章　分布間距離の推定

空でない集合 X を考える。$\forall x, y \in X$ に対して、ある実数関数値 $d(x, y)$ が存在するとき、次を満たすと仮定する。

- 非負性：$\forall x, y,\ d(x, y) \geq 0$
- 非退化性：$d(x, y) = 0 \Leftrightarrow x = y$
- 対象性：$\forall x, y,\ d(x, y) = d(y, x)$
- 三角不等式：$\forall x, y, z,\ d(x, z) \leq d(x, y) + d(y, z)$

この関数 $d(x, y)$ を「集合 X 上の距離関数」といい、X に d を導入した (X, d) を「距離空間」と呼ぶ。なお、上に示した 4 つを「距離の公理」という。

本章では、機械学習に有用な距離尺度と関連研究について紹介する。

5.1　確率分布間の距離

確率分布間の距離は、さまざまなデータ解析タスクで重要な働きをする基礎的な概念である。ここでは、確率変数が x である二つの確率分布 p, q が与えられたときの確率分布間距離の尺度をいくつか説明する。

5.1.1　カルバック・ライブラー距離（KL 距離）

定義される式は以下である。

$$KL(p \,||\, q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{5.1}$$

KL 距離は相対エントロピーとも呼ばれており、最尤推定との相性が良く、密度比推定によって精度良く近似することができる尺度である。また、KL 距離には以下の性質がある。

$$\begin{aligned} KL(p \,||\, q) &\leq 0 \\ KL(p \,||\, q) = 0 &\Leftrightarrow p = q \\ KL(p \,||\, q) &\neq KL(q \,||\, p) \end{aligned}$$

性質からも分かるように、KL 距離は非対称である。また、外れ値の影響を受けやすい欠点もある。

5.1.2 IR 距離

定義される式は以下である。

$$IR(p,q) = \sum_i \Big[p(x_i) \log \frac{2p(x_i)}{p(x_i)+q(x_i)} + q(x_i) \log \frac{2q(x_i)}{p(x_i)+q(x_i)} \Big]$$

IR 距離は以下の性質を持っている。

$$\begin{aligned} IR(p,q) &\leq 0 \\ IR(p,q) = 0 &\Leftrightarrow p = q \\ IR(p,q) &= IR(q,p) \end{aligned}$$

IR 距離は、KL 距離を対象性を満たすように拡張したものである。

5.1.3 f 距離

定義される式は以下である。

$$f(p \mid\mid q) = \int q(x) f(\frac{p(x)}{q(x)}) dx$$

なお、$f(t) = t \log t$ とおけば、KL 距離と一致する。これは $f(1) = 0$ を満たす凸関数である。

5.1.4 ピアソン距離

定義される式は以下である。

$$PE(p \mid\mid q) = \int q(x) \Big(\frac{p(x)}{q(x)} - 1 \Big)^2 dx$$

ピアソン距離は、最小二乗法と相性が良く、密度比推定によって精度良く近似することができる。しかし、外れ値の影響を受けやすく、対象性と三角不等式を満たさない欠点もある。

ここで、密度比 $p(x)/q(x)$ について言及する。密度比は、場合によっては発散してしまう可能性がある。このため KL 距離やピアソン距離は外れ値の影響を受けやすくなっている。そこで相対的な密度比 $p(x)/q_\beta(x)$ を考える。

$$\frac{p(x)}{q_\beta(x)} < \frac{1}{\beta}$$

なお、$0 \leq \beta \leq 1$、 $q_\beta(x) = \beta p(x) + (1-\beta) q(x)$ とする。こうすることで発散を防ぐことができる。

5.1.5 相対ピアソン距離

上で説明した相対密度比を使うことで、ピアソン距離を改良したものを相対ピアソン距離という。

$$rPE(p \,||\, q) = PE(p \,||\, q_\beta) = \int q_\beta(x)\Big(\frac{p(x)}{q_\beta(x)} - 1\Big)^2 dx$$

相対ピアソン距離は、最小二乗法と相性が良く、外れ値の影響を受けにくい。また相対密度比によって精度良く近似することができる。

5.1.6 L^2 距離

定義される式は以下である。

$$L^2(p, q) = \int \Big(p(x) - q(x)\Big)^2 dx$$

この距離の尺度は、距離の公理を全て満たす点で優れている。最小二乗法と相性が良く、密度差推定によって精度良く近似することができる。また、外れ値の影響を受けにくい。

5.2 KL 距離の推定

データ $\{x_1, x_2, \cdots, x_n\}$ から KL 距離を推定することを考える。KL 距離の定義式は式 5.1 に示した。この式を変形すると以下のようになる。

$$\int_{i=1}^{m} p(x_i) \log \frac{p(x_i)}{q(x_i)} = \int_{i=1}^{m} p(x_i) \log p(x_i) - \underline{\int_{i=1}^{m} p(x_i) \log q(x_i)} \tag{5.2}$$

この右辺第 1 項は、仮定した真の分布にのみ依存する。そのため下線を引いた第 2 項を大きくする確率モデル $q(\boldsymbol{x})$ を導き出すことが目的となる。$\log q(x_i) = a_i$ とおいて、

$$\int_{i=1}^{m} p(x_i) \log q(x_i) = \int_{i=1}^{m} p(x_i) a_i = E[\boldsymbol{a}] \tag{5.3}$$

なお、$E[\boldsymbol{a}]$ は $\boldsymbol{a}$ の期待値とする。
ここで、確率変数 a の試行を $n \longrightarrow \infty$ 回繰り返す。その平均値が $E[\boldsymbol{a}]$ の値に収束する。

$$\overline{\boldsymbol{a}} = \frac{a_{x_1} + a_{x_2} + \cdots + a_{x_n}}{n} = \frac{1}{n} \int_{j=1}^{n} a_{x_j} \tag{5.4}$$

大数の法則により、$n \longrightarrow \infty$ のとき、$E[\boldsymbol{a}]$ と一致する。

$$\int_{i=1}^{m} p(x_i) \log q(x_i) \longrightarrow \frac{1}{n} \int_{i=1}^{n} \log a(x_i) \tag{5.5}$$

こうして推定が可能である。

第6章　提案手法

6.1　素性ベクトル

本実験で使用する素性ベクトルを構成する要素は、以下の 9 種類である。クラスを識別する対象となる単語を w とし、その直前の単語を w_{-1}、直後の単語を w_1 とする。領域は OC（Yahoo! 知恵袋)、PN(新聞)、PB(書籍) とする。

g ：領域（OC or PN or PB）

e0 ：単語 w の表記

e1 ：単語 w の品詞

e2 ：単語 w_{-1} の表記

e3 ：単語 w_{-1} の品詞

e4 ：単語 w_1 の表記

e5 ：単語 w_1 の品詞

e6 ：w の周辺にある自立語を表記

e7 ：e6 の分類語彙表[1]における番号

例えば、ある文書中の「聞く」という単語から素性ベクトルを作成すると、

g=OC, e0=聞き, e1=動詞-一般,
e2=とても, e3=副詞,
e4=づらい, e5=接尾辞,
e6=放送, e6=ニッポン, e6=うち,
e7=1312, e7=13123

といったベクトルが作成される。

6.2　手法

本実験では、教師無し学習（unsupervised）の手法によるナイーブベイズ分類器（Naive Bayes classifier）を用いる。対象単語 w の語義の集合を S として、w の文脈を素性ベクトル $f = (f_1, f_2, \cdots, f_m)$ で表すと、ナイーブベイズ分類器では、以下の式により w の語義を推定する。

$$arg\max_{s \in S} p(s) \prod_{i=1}^{m} p(f_i \mid s)$$

[1]語を意味によって分類・整理した辞書やデータベース

ナイーブベイズ分類器では $p(s)$ と $p(f_i \mid s)$ をソースドメインの訓練データから推定する。今、ソースドメインから求めたそれぞれの分布を $p_s(s)$ と $p_s(f_i \mid s)$ とおく。またターゲットドメインにおけるそれぞれの分布を $p_t(s)$ と $p_t(f_i \mid s)$ とおく。本論文では、$p_t(f_i \mid s) = p_s(f_i \mid s)$ と仮定し、$p_t(s)$ をソースドメインのコーパスとターゲットドメインのコーパス、および $p_s(s)$ から推定することで領域適応を行う。

そこで、$p_t(s)$ を $p_s(s)$ と一様分布 $u(s)$ との混合分布で近似することを考える。

$$p_t(s) = \alpha p_s(s) + (1 - \alpha)u(s)$$

ここで、混合比 α が問題となる。本稿では、α を $p_s(s)$ と $p_t(s)$ との確率分布間距離の逆数とする。距離が 0 であれば、$\alpha = 1$ とし、距離が大きくなるほど $\alpha = 0$ に近づく。

そして、この分布間距離を測るために以下の操作を行う。（図 6.1）

1. 対象単語 w に対するソースドメインの用例 50 個とターゲットドメインの用例 50 個を合わせた 100 個の用例をクラス数 4 に固定してクラスタリングを行う。得られたクラスタを S_1, S_2, S_3, S_4 とする。

2. クラスタ S_i に含まれるソースドメインの用例数を s_i、ターゲットドメインの用例数を t_i とする。$s_1 + s_2 + s_3 + s_4 = 50$、$t_1 + t_2 + t_3 + t_4 = 50$ が成立する。

3. $p_s(s)$ を代用する分布を $p = (s_1/50, s_2/50, s_3/50, s_4/50)$ とし、$p_t(s)$ を代用する分布を $q = (t_1/50, t_2/50, t_3/50, t_4/50)$ として、分布 p, q の IR 距離を測り、それを分布間距離とする。

以上の操作で求まった分布間距離の逆数を α に代入して、混合比に設定する。

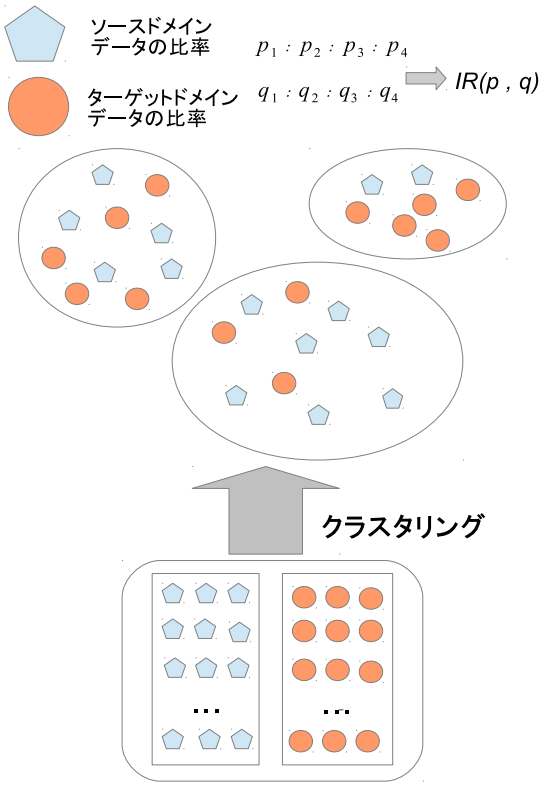

図 6.1: IR 距離の算出仮定

第7章　実験

本実験では、BCCWJ コーパスの新聞（PN)、Yahoo!知恵袋（OC)、書籍（PB）の 3 つのドメインのコーパスを利用した。また以下に示す単語 10 個のそれぞれについて、各コーパスから 50 用例を取り出した。各領域における 各単語の 50 用例が訓練データとテストデータになる。

聞く、言う、時間、自分、出る

場合、前、見る、持つ、考える

領域の組み合わせは次の 6 通りが考えられる。

表 7.1: 領域適応の組み合わせ

領域適応のタイプ	ソースドメイン	ターゲットドメイン
OC-PN	Yahoo!知恵袋	新聞
OC-PB	Yahoo!知恵袋	書籍
PN-OC	新聞	Yahoo!知恵袋
PN-PB	新聞	書籍
PB-OC	書籍	Yahoo!知恵袋
PB-PN	書籍	新聞

ソースドメインの訓練データのみから学習した NB により、ターゲットドメインのテストデータを識別させる手法を Normal-NB とする。Normal-NB の正解率がベースラインとなる。またターゲットドメインの訓練データに対して、NB を用いて One-vs-Other 法の交差検定から識別させる手法を Cross-NB とする。この正解率が上限となる。Normal-NB と Cross-NB、および提案手法の実験結果を表 7.2 に示す。

表 7.2: 実験結果（識別の平均正解率）

領域適応のタイプ	Normal-NB (ベースライン)	Cross-NB (上限値)	提案手法
OC-PN	79.60%	84.29%	79.40%
OC-PB	80.60%	83.27%	76.80%
PN-OC	76.00%	82.04%	75.20%
PN-PB	78.20%	83.27%	76.80%
PB-OC	77.40%	82.04%	75.40%
PB-PN	81.80%	84.29%	80.40%

また本論文では、$p_s(s)$ と $p_t(s)$ の分布間距離の推定も行っている。ソースドメインとターゲットドメインの両方のデータを用いて得られる $p_s(s)$ と $p_t(s)$ から実際の IR 距離と、本論文

で推定した分布間距離の比較を表 7.3 にまとめた。

表 7.3: 距離の比較

領域対	実際の距離	推定した距離
OC-PN (聞く)	0.011339827412828	0.00916588879739783
OC-PN (言う)	0.00183913520880368	0.342149297744164
OC-PN (時間)	0.0530257600677961	0.166244747755612
OC-PN (自分)	0.0807333466930704	0.0448151805294378
OC-PN (出る)	0.0427239459232424	0.0211038135066326
OC-PN (場合)	0.0545361451935044	0.0242527666823595
OC-PN (前)	0.0786393586844086	0.0685306301386311
OC-PN (見る)	0.299507525943364	0.291848736193869
OC-PN (待つ)	0.166673456748398	0.206078839081979
OC-PN (考える)	0.028134078825291	0.0328296437306609
PN-PB (聞く)	0.000507445678299093	0.0119095811545379
PN-PB (言う)	0.0110173090051153	0.25184968696276
PN-PB (時間)	0.100051514808951	0.22072967507211
PN-PB (自分)	0 0.0367662962873135	0.0367662962873135
PN-PB (出る)	0.0614456789150188	0.0541643621492555
PN-PB (場合)	0.000784811146700247	0.012196314137299
PN-PB (前)	0.105396414869774	0.105961606926621
PN-PB (見る)	0.0386089158212089	0.0312871017070503
PN-PB (待つ)	0.071289349538833	0.100764004583094
PN-PB (考える)	0.00350107534562855	0.0073710310682139
PB-OC (聞く)	0.0166178714493244	0.0127889249544648
PB-OC (言う)	0.00386421647571343	0.0105441273811344
PB-OC (時間)	0.00802055978441636	0.0274917509362417
PB-OC (自分)	0.0807333466930704	0.0721459726646436
PB-OC (出る)	0.0324641389635975	0.0207896003028752
PB-OC (場合)	0.067366535139215	0.0206413805608184
PB-OC (前)	0.0880421563618802	0.115263371355251
PB-OC (見る)	0.222519188294599	0.0385505526684944
PB-OC (待つ)	0.0468545662273027	0.0551300320288762
PB-OC (考える)	0.0139639554299628	0.0164076416058136

第8章　考察

提案手法は、本実験では有効に機能しなかった。原因としては2つ考えられる。

8.1　混合分布のモデル化

今回、ターゲットドメインの語義分布 $p_t(s)$ をソースドメインの語義分布 $p_s(s)$ と一様分布との混合分布と仮定したが、その混合比 α を単純に $p_s(s)$ と $p_t(s)$ の IR 距離の逆数 $IR^{-1}(p,q)$ に設定してしまった点である。確かに α と $IR^{-1}(p,q)$ との間には正の相関関係があるが、その値を直接用いて $p_t(s)$ の補正をしてもうまくいかない。$p_t(s)$ を一様分布に近似することは、確率分布 $p_t(s)$ が $p_s(s)$ とかなり異なる分布をしていないと有効に機能しない。今回の $p_t(s)$ と $p_s(s)$ の分布は、そこまでの差異がなかった。現実的には、$p_s(s)$ と $p_t(s)$ がかなり異なったときのみに、小さな α 値で $p_t(s)$ を構成したほうが適切だったと予測している。

8.2　確率分布間距離の推定手法

$p_s(s)$ と $p_t(s)$ の距離の算出方法が適切でなかったと考えている。本論文のポイントは、$p_t(s)$ を直接推定せずに、$p_s(s)$ との距離を測ることで WSD の領域適応を行うことであった。つまり $p_t(s)$ を正確に推定する必要がないため、簡易な方法で距離を測ってしまった。例えば、今回はクラスタリングを行って、クラスタに属しているデータ数から分布間距離を求めているが、その際のクラスタ数は 4 個に固定している。クラスタ数は対象単語の語義数に指定するのが自然であるが、距離さえ求まれば良いのでクラスタ数はいくつでも良いと判断し、4 個に固定したが安易過ぎたかもしれない。また距離を図る際に $p_s(s)$ の分布が既知であるのに、その情報をまったく利用しないことは大きな問題であると考える。このため、本手法ではまったくランダムにクラスタリングした場合、理論的には $p_s(s)$ と $p_t(s)$ との距離が 0 になってしまうという明らかな不備がある。

8.3　提案手法のポイント

提案手法に不備があるが、本稿のアイデアとしては妥当であったと考えている。本稿のポイントは以下の 2 点である。

1. WSD の領域適応では、ターゲットドメインの語義分布を推定することが鍵である。
2. ターゲットドメインの語義分布を直接求める必要は無く、ソースドメインとの距離から求めることが可能である。

特に 2 点目に関しては、機械学習の分野で近年、研究が活発な分布間距離の推定の研究が応用できると考えている。また、領域適応は転移学習の一部であるため、転移学習で有効に機能している手法が領域適応でも応用できると考えている。それらの研究を応用して本実験の改良をしていきたい。

第9章　おわりに

本論文では、WSD のタスクでの領域適応に対する教師無し学習の手法を提案した。提案手法では、Naive Bayes を基本としている。Naive Bayes では、ターゲットドメインの語義分布 $p_t(s)$ と事後分布の要素である $p_t(f \mid s)$ が必要だが、$p_t(f \mid s)$ はソースドメインに対する $p_s(f \mid s)$ で代用し、$p_t(s)$ をソースドメインの語義分布 $p_s(s)$ と一様分布の混合分布として推定している。そして、その混合比を $p_s(s)$ と $p_t(s)$ の分布間距離から得る。実験では、ソースドメインとターゲットドメインの組み合わせ 6 パターンに対して、10 単語に関する WSD を行い、その平均正解率を調べた。結果的には、提案手法の有効性は確認できなかった。ただし、アイデア的には有望だと考えられるので、今後は分布間距離の推定法の改良をすることで手法の有効性を示すとともに、転移学習の研究成果を応用して結果の改善をしていきたい。

謝辞

本研究を進めるにあたり、多大なご指導ご協力を頂いた新納浩幸准教授に心から感謝いたします。また、日常の議論を通じて多くの知識や示唆を頂いた新納研究室の皆様に感謝します。

参考文献

[1] Daumé：” Frustratingly Easy Domain Adaptation ”, *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp 256-263(2007).

[2] Daumé：” Frustratingly Easy Semi-Supervised Domain Adaptation ”, *In Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing in ACL 2010*, pp 2359(2010).

[3] Agirre,E. and de Lacalle, O.L.：” On Robustness and Domain Adaptation using SVD dor Word Sense Disambiguation ”, *In Proceeding of the 22nd International Conference on Computational Linguistics*, pp 17-24(2008).

[4] Agirre, E. and de Lacalle, O.L.:” Supervised Domain Adaptation for WSD ”, *In Proceeding of the 12th Conference of the European Chapter of the Association of Computational Linguistics*, pp 42-50(2009).

[5] Chan, Y. S. and Ng. H. T.：” Estimating Class Priors in Domain Adaptation for Word Sense Disambiguation ”, *In Proceedings of the 21th International Confetence on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp89-96(2006).

[6] Chan, Y. S. and Ng, H. T.：” Domain Adaptation with Active Learning for Word Sense Disambiguation ”, *In Proceedings of the 45th Annual Meeting of the Associatoion of Computational Linguistics*, pp49-56(2007).

[7] Marthinus Christoffel du Plessis and MAsashi Sugiyama：” Semi-Supervised Learning of Class Balance under Class-Prior Change by Distribution Matching ”, *In Proceeding of 29th International Conference on Machine Learning (ICML2012)*, pp 823-830(2012).

[8] NIPS 2005 Workshop-Inductive Transfer: 10 Years Later, http://iitrl.acadiau.ca/itws05/ (2005).

[9] Kanako Komiya and Manabu Okumura：” Automatic Domain Adaptation for Word Sense Disambiguation Based on Comparison of Multiple Classifiers ”, *International Joint Conference on Natural Language Processing(IJCNPL 2011)*, pp 1107-1115(2011).

[10] Kanako Komiya and Manabu Okumura：” Automatic Domain Adaptation for Word Sense Disambiguation Based on Comparison of Multiple Classifiers ”, *PACLIC-2012*, pp 75-85(2012).

[11] 古宮嘉那子，奥村学：” 語義曖昧性解消のための領域適応手法の自動選択 ”, *ALC 2008*, pp 101-102(2008).

[12] 古宮嘉那子, 奥村学：”語義曖昧性解消のための領域適応手法の決定木学習による選択－三手法からの決定－ ”, 言語処理学会　第１８回年次大会発表論文集, pp 1288-1291(2012).

[13] 玉垣隆幸、白井清昭：” 読解支援システムのための語義曖昧性解消に関する研究 ”, 自然言語処理学会年次大会発表論文集, pp 481-484(2003).

[14] 神嶌敏弘：” 転移学習 ”, 人工知能学会誌 vol.25 no.4, pp 572-580(2010).

[15] 高村大也：” 自然言語処理シリーズ１　言語処理のための機械学習入門 ”, コロナ社, 211pp(2010).

[16] 奥村学：” 自然言語処理の基礎 ”, コロナ社, 155pp(2010).

付 録A　ソースリスト

```
#!/usr/bin/perl
use strict;
use warnings;

my (%number, %count); #語義の番号、語義 i の数を格納する連想配列
my $n = 0;          #語義の数をカウントする変数

#ソースドメインの訓練データを読み込む
open(IN, "< ../semebal/$ARGV[0]/$ARGV[1]")
    or die "Cannot open $ARGV[0]/$ARGV[1]: $!";

while(<IN>){#訓練データを読み終わるまで実行
    chomp($_);
    my @sepa = split(/\s/, $_);
    $count{$sepa[$#sepa]} += 1;    #語義 i の数をインクリメント
}
close(IN);

#書き込み用のファイルをオープン
open(OUT, "> test_data/ps.txt")
    or die "Cannot open data.txt: $!";
foreach my $key (keys %count){#値を書き込む
    $n++;
    $number{$key} = $n;
    print OUT "$key\t$number{$key}\t$count{$key}\n";
}
close(OUT);
```

ソースコード A.1: 語義分布を調べるプログラム

```
#!/usr/bin/perl
use strict;
use warnings;

my (%number, %count); #語義の番号、語義 i の数を格納する連想配列
my $n = 0;          #語義の数を格納する変数

#ps.txt(学習データ1)を読み込む
open(IN, "< test_data/ps.txt")
  or die "Cannot open ps.txt: $!";
while(<IN>){#ファイルを読み終わるまで実行
  my @sepa = split(/\t/ , $_);
  if($sepa[0] ne "EOS"){
    $number{$sepa[0]} = $sepa[1]; #語義識別の番号を読み込む
    $n++;               #語義の数をインクリメント
  }
}
```

```
close(IN);

#訓練データからpsx.txt(学習データ2)を作成
open(IN, "< ../semebal/$ARGV[0]/$ARGV[1]")
  or die "Cannot open: $!";
while(<IN>){
  chomp($_);
  my @sepa = split(/\s/ , $_);

  #まず文章dに対して各組成を読み込み
  #クラスc1~ckまでの数をカウントするための土台をつくる
  foreach my $key (keys %number){
    for(my $i = 1; $i < $#sepa; $i++){
      if($sepa[$i] ne ""){
        if(exists$count{"$key/$sepa[$i]"} != 1){
            $count{"$key/$sepa[$i]"} = 0;
        }
        }
    }
  }
    #クラスcの文章dの組成f1~fnをカウント
    for(my $i = 1; $i < $#sepa; $i++){
    if($sepa[$i] ne ""){
      $count{"$sepa[$#sepa]/$sepa[$i]"}++;
    }
    }
}
close(IN);

#学習データ2をテキストファイルに書き込む
open(OUT, "> test_data/psx.txt")
  or die "Cannot open $ARGV[0]: $!";
foreach my $key2 (keys %count){
    my @sepa = split(/\// , $key2);
    print OUT "$sepa[1]\t$number{$sepa[0]}\t$count{$key2}\n";
}
close(OUT);
```

ソースコード A.2: 素性の数を調べるプログラム

```
#!/usr/bin/perl
use strict;
use warnings;

my (%num, %count, %sim);  #語義番号、
    train_Dで語義iが出てきた数、推定が正解かどうかの判定用
my %word;
my $words;            #総語彙数を格納する変数
my $N     = 50;         #用例の数
my $K     =  0;         #語義の数
my $prior =  0;         #test_Dの正解数を格納する(正解率)
my @p_s;           #learn_Dから求めたP(s)の値を格納する変数
my $IR = 0;

my @count;
my $sum;
my @hose;
```

```
17
18 #語義番号とその数を記録したファイルからデータを読み込む
19 open(IN, "< test_data/ps.txt")
20   or die "Cannot open ps.txt: $!";
21 while(<IN>){
22   chomp($_);
23   my @sepa = split(/\t/, $_);
24   $num{$sepa[0]}   = $sepa[1];     #語義番号を格納
25   $count{$sepa[1]} = $sepa[2];     #語義 i が出てきた数を格納
26   $sim{$sepa[1]}   = $sepa[0];     #判定のため格納
27   $K++;          #語義の数をインクリメント
28 }
29 close(IN);
30
31 #P(s)の値を計算する
32 for(my $i = 1; $i <= $K; $i++){$p_s[$i] = log(($count{$i} + 1)/($N +
       $K));}
33
34 #IR距離を読み込む
35 open(IN, "< ./distance")
36   or die "Can't open distance: $!";
37 while(<IN>){
38   chomp($_);
39   $IR = $_;
40   if($IR != 0){$IR = 1/$IR;}
41 }
42 close(IN);
43
44 #テストデータから学習したのもを読み込む
45 open(IN, "< test_data/psx.txt")
46   or die "Cannot open psx.txt: $!";
47 while(<IN>){
48   chomp($_);
49   my @sepa = split(/\t/, $_);
50
51   #出現するクラス s_i である単語を含む文章の数を格納
52   $word{"$sepa[1]/$sepa[0]"} = $sepa[2];
53 }
54 close(IN);
55
56
57 #p(s_i)の値を調整するための処理
58 for(my $i = 1; $i <= $K; $i++){$p_s[$i] = ($IR * $p_s[$i]) + (1 - $IR
       )*$K};
59
60
61 #nbの計算を行う
62 open(IN, "< ../semebal/$ARGV[0]/$ARGV[2]")
63   or die "Cannot open $ARGV[2]: $!";
64 while(<IN>){
65   my ($max, $pre);     #nbの値が大きいほうを格納、予測された語義番号
66   my @p_fs;        #P(f|s)の値を入れる配列
67
68   chomp($_);
69   my @sepa = split(/\s/, $_);
70
```

```
71    #語義 i に対して
72    for(my $i = 1; $i <= $K; $i++){
73      my $adjust = 0;
74      for(my $j = 1; $j < $#sepa; $j++){
75        my $c = 1;
76        if($sepa[$j] eq ""){next;}
77
78        # test_D の語義 i における nb の値を計算する
79        #該当する組成がある場合
80        if(exists$word{"$i/$sepa[$j]"} == 1){
81          $p_fs[$i] += log(($word{"$i/$sepa[$j]"} + 1)/($N + 2));
82        }else{ #ゼロ頻度問題の対応(スムージング)
83          $p_fs[$i] += log(1/($N + 2));
84        }
85      }
86      #最終的な nb の値
87      $p_fs[$i] = $p_s[$i] + $p_fs[$i];
88    }
89
90    # nb の最大値を求めて、語義(クラス)を予測する
91    $max = $p_fs[1];
92    for(my $i = 1; $i <= $K; $i++){
93      if($max <= $p_fs[$i]){
94        $max = $p_fs[$i];
95        $pre = $num{$sim{$i}};
96      }
97    }
98
99    #語義の予測が当たっていた場合
100   if(!$num{$sepa[$#sepa]}){}
101   elsif($pre == $num{$sepa[$#sepa]}){
102     $prior++;    #正解数をインクリメント
103   }
104 }
105 close(IN);
106
107 #正解率を表示
108 if($prior != 0){$prior = ($prior/$N) * 100;}
109 print "$prior\n";
110
111
112 my @s1 = split(/-/, $ARGV[1]);
113 my @s2 = split(/-/, $ARGV[2]);
114 chomp($s1[1]);
115 chomp($s2[1]);
116 open(OUT, ">> result/$s1[1]-$s2[1].txt")
117   or die "Cannot open $s1[1]-$s2[1].txt: $!";
118 print OUT "$prior\n";
119 close(OUT);
```

ソースコード A.3: Naive Bayes を実行するプログラム

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
```

```
5
6  my $N = 50;      #用例の数
7  my $t_place = 0;  #ターゲットととなる用例の位置
8  my $prior = 0;    #正解率を格納
9
10 #推定をする用例の読み込み
11 open(IN, "< ../semebal/$ARGV[0]/$ARGV[1]")
12   or die "Can't open $ARGV[1]: $!";
13 while(<IN>){
14   my $s_place = 0;    #ソースの位置を格納
15   my $n       = 0;    #語義の数をカウント
16   my (%num, %count);    #語義の番号,語義 i の用例数
17   my %sosei;          #生起する単語を格納(sosei)
18   my %sim;            #判定用のハッシュ変数
19
20   chomp($_);
21   my @target = split(/\s/, $_); #ターゲットを格納
22   $t_place++;              #ターゲットの位置をインクリメント
23
24   #ソースデータの学習
25   open(DATA, "< ../semebal/$ARGV[0]/$ARGV[1]")
26     or die "Can't open $ARGV[1]: $!";
27   while(<DATA>){
28     $s_place++;   #ソースデータの位置をインクリメント
29     if($t_place == $s_place){next;}
30     else{
31       chomp($_);
32       my @source = split(/\s/, $_);      #ソースデータを格納
33       if(exists$num{$source[$#source]} != 1){ #語義に番号がふられていない場合
34         $n++;          #語義の数をインクリメント
35         $num{$source[$#source]} = $n;    #語義に番号をふる
36         $sim{$n} = $source[$#source];    #判定用の変数に格納
37       }
38       $count{$num{$source[$#source]}} += 1; #語義の数をカウント
39       for(my $i = 1; $i < $#source; $i++){
40         if($source[$i] ne ""){
41           $sosei{"$num{$source[$#source]}/$source[$i]"}++;
42         }
43       }
44     }
45   }
46   close(DATA);
47
48   my ($max, $pre);  #
       nbの最大値を格納する変数、予測された語義の番号を格納
49   my (@p_s, @p_fs); #p(s) ,p(f|s)の値を格納する配列
50
51   #語義 iに対して
52   for(my $i = 1; $i <= $n; $i++){
53     $p_s[$i] = log(($count{$i} + 1)/(($N-1) + $n)); #p(s)の値を計算
54     #P(f|s)の値を計算
55     for(my $j = 1; $j < $#target; $j++){
56       if($target[$j] eq ""){next;}
57       if(exists$sosei{"$i/$target[$j]"} == 1){
58         $p_fs[$i] += log(($sosei{"$i/$target[$j]"} + 1)/(($N-1) + 2));
59       }else{
```

```
60            #ゼロ頻度問題の処理(スムージング)
61            $p_fs[$i] += log(1/(($N-1) + 2));
62          }
63        }
64        #最終的なp_sdの値
65        $p_fs[$i] = $p_s[$i] + $p_fs[$i];
66      }
67      $max = $p_fs[1];
68      #NBの値の最大値と予測される語義番号の判定
69      for(my $i = 1; $i <= $n; $i++){
70        if($max <= $p_fs[$i]){
71          $max = $p_fs[$i];
72          $pre = $sim{$i};
73        }
74      }
75      #語義の推定が当たっていた場合
76      if($pre eq $target[$#target]){
77        $prior++;
78      }
79  }
80  close(IN);
81
82  #正解率を計算
83  if($prior != 0){$prior = ($prior/($N-1)) * 100;}
84  print "$prior\n";
85
86  my @s1 = split(/-/, $ARGV[1]);
87  chomp($s1[1]);
88  open(OUT, ">>␣result/$s1[1]-$s1[1].txt")
89    or die "Can't␣open␣$s1[1]-$s1[1].txt:␣$!";
90  print OUT "$prior\n";
91  close(OUT);
```

ソースコード A.4: 交差検定の Naive Bayes

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  my $ave;  #平均の値を格納
7  my $count;  #値の数をカウント
8
9  #正解率の記述されたファイルをオープン
10 open (IN, "<␣result/$ARGV[0]")
11     or die "Cannot␣opne!:␣$!";
12
13 while(<IN>){
14     chomp($_);
15     $ave += $_; #値を加算
16     $count++; #値の数をインクリメント
17 }
18
19 $ave = $ave/$count; #平均値を求める
20
21 print "$ave\n";
```

```
22 unlink $str;
```

ソースコード A.5: 正解率の平均を求めるプログラム

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my %matrix;      #類似度行列を格納
6  my $h_count = 1;   #その時点での列の位置を格納
7  my $N = 100;     #類似度を計算するの用例数
8
9  #ファイルを開く
10 open(IN, "< ../semebal/$ARGV[0]/$ARGV[1]")
11   or die "Can't open $ARGV[1]: $!";
12 while(<IN>){
13   my $x;         #素性ベクトルの大きさを格納
14   my $v_count = 0;   #行の位置を格納
15   chomp($_);
16   my @sepa = split(/\s/, $_);
17
18   #分母/x/の値を計算
19   for(my $i = 1; $i < $#sepa; $i++){if($sepa[$i] ne ""){$x++;}};
20
21   open(DATA, "< ../semebal/$ARGV[0]/$ARGV[1]")
22     or die "Can't open $ARGV[1]: $!";
23   while(<DATA>){
24     $v_count++;
25
26     my $y = 0;     #素性ベクトルの大きさを格納
27     my $sim = 0;   #類似度を格納
28     chomp($_);
29     my @sepa2 = split(/\s/, $_);
30     #分母/y/の値を計算
31     for(my $i = 1; $i < $#sepa2; $i++){if($sepa2[$i] ne ""){$y++;}}
32
33     #素性が同じものがあればインクリメント
34     for(my $i = 1; $i < $#sepa; $i++){
35       for(my $j = 1; $j < $#sepa2; $j++){
36         if(($sepa[$i] ne "") && ($sepa2[$j] ne "")){
37           if($sepa[$i] eq $sepa2[$j]){$sim++;}
38         }
39       }
40     }
41     #類似度の計算（余弦定理）
42     if($sim != 0){$sim = $sim / (sqrt($x) * sqrt($y));}
43     #行列に格納
44     if(exists$matrix{"$h_count:$v_count"} != 1){
45       $matrix{"$h_count:$v_count"} = $sim;
46     }
47   }
48   close(DATA);
49
50   open(DATA, "< ../semebal/$ARGV[0]/$ARGV[2]")
51     or die "Can't open $ARGV[2]: $!";
52   while(<DATA>){
```

```
53      $v_count++;
54
55      my $y = 0;      #素性ベクトルの大きさを格納
56      my $sim = 0;    #類似度を格納
57      chomp($_);
58      my @sepa2 = split(/\s/, $_);
59      #分母/y/の計算
60      for(my $i = 1; $i < $#sepa2; $i++){if($sepa2[$i] ne ""){$y++;}}
61
62      for(my $i = 1; $i < $#sepa; $i++){
63        for(my $j = 1; $j < $#sepa2; $j++){
64          if(($sepa[$i] ne "") && ($sepa2[$j] ne "")){
65            if($sepa[$i] eq $sepa2[$j]){$sim++;}
66          }
67        }
68      }
69      if($sim != 0){$sim = $sim/(sqrt($x) * sqrt($y));}
70      if(exists$matrix{"$h_count:$v_count"} != 1){
71        $matrix{"$h_count:$v_count"} = $sim;
72      }
73    }
74    close(DATA);
75    $h_count++;
76  }
77  close(IN);
78
79  open(IN, "< ../semebal/$ARGV[0]/$ARGV[2]")
80    or die "Can't open $ARGV[2]: $!";
81  while(<IN>){
82    my $x;
83    my $v_count = 0;
84    chomp($_);
85    my @sepa = split(/\s/, $_);
86    #分母/x/の値を計算
87    for(my $i = 1; $i < $#sepa; $i++){if($sepa[$i] ne ""){$x++;}}
88
89    open(DATA, "< ../semebal/$ARGV[0]/$ARGV[1]")
90      or die "Can't open $ARGV[1]: $!";
91    while(<DATA>){
92      $v_count++;
93
94      my $y = 0;
95      my $sim = 0;
96
97      chomp($_);
98      my @sepa2 = split(/\s/, $_);
99      #分母/y/の値を計算
100     for(my $i = 1; $i < $#sepa2; $i++){if($sepa2[$i] ne ""){$y++;}}
101
102     for(my $i = 1; $i < $#sepa; $i++){
103       for(my $j = 1; $j < $#sepa2; $j++){
104         if(($sepa[$i] ne "") && ($sepa2[$j] ne "")){
105           if($sepa[$i] eq $sepa2[$j]){$sim++;}
106         }
107       }
108     }
```

```
109     if($sim != 0){$sim = $sim / (sqrt($x) * sqrt($y));}
110     if(exists$matrix{"$h_count:$v_count"} != 1){
111       $matrix{"$h_count:$v_count"} = $sim;
112     }
113   }
114   close(DATA);
115
116
117   open(DATA, "< ../semebal/$ARGV[0]/$ARGV[2]")
118     or die "Can't open $ARGV[2]: $!";
119   while(<DATA>){
120     $v_count++;
121     my $y = 0;
122     my $sim = 0;
123
124     chomp($_);
125     my @sepa2 = split(/\s/, $_);
126     #分母/y/の値を計算
127     for(my $i = 1; $i < $#sepa2; $i++){if($sepa2[$i] ne ""){$y++;}}
128
129     for(my $i = 1; $i < $#sepa; $i++){
130       for(my $j = 1; $j < $#sepa2; $j++){
131         if(($sepa[$i] ne "") && ($sepa2[$j] ne "")){
132           if($sepa[$i] eq $sepa2[$j]){$sim++;}
133         }
134       }
135     }
136     if($sim != 0){$sim = $sim / (sqrt($x) * sqrt($y));}
137     if(exists$matrix{"$h_count:$v_count"} != 1){
138       $matrix{"$h_count:$v_count"} = $sim;
139     }
140   }
141   close(DATA);
142   $h_count++;
143 }
144 close(IN);
145
146 my $n = keys(%matrix);
147 print ("$N $n\n");
148 for(my $i = 1; $i <= $N; $i++){
149   for(my $j = 1; $j <= $N; $j++){
150     print (" $j ");
151     print $matrix{"$i:$j"};
152     if($j == $N){print ("\n");}
153   }
154 }
```

ソースコード A.6: 分布間の類似度行列を作成するプログラム

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $row = 1;        #行の位置を格納
6  my @cluster1 =(0,0,0,0);   #それぞれのクラスタの数を計算
7  my @cluster2 =(0,0,0,0);   #それぞれのクラスタの数を計算
```

```
 8 my ($count1, $count2);  #合計の数を計算
 9 my $IR;          #IR距離を格納
10
11 my @sepa = split(/([\-\.])/, $ARGV[1]);
12
13 open(IN, "<␣./sim_research/$ARGV[0]/$ARGV[1]")
14   or die "Can't␣open␣$ARGV[1]:␣$!";
15 while(<IN>){
16   chomp($_);
17
18   if($row <= 50){
19     $cluster1[$_]++;
20     $count1++;
21   }elsif($row >=51){
22     $cluster2[$_]++;
23     $count2++;
24   }
25   $row++;
26 }
27
28
29 #クラスタの分布を求める
30 for(my $i = 0; $i <= 2; $i++){
31   if($cluster1[$i] != 0){
32     $cluster1[$i] = $cluster1[$i]/$count1;
33   }
34 }
35
36 #クラスタの分布を求める
37 for(my $j = 0; $j <= 2; $j++){
38   if($cluster2[$j] != 0){
39     $cluster2[$j] = $cluster2[$j]/$count2;
40   }
41 }
42
43
44 #IR距離を求める
45 for(my $i = 0; $i <= 2; $i++){
46   if($cluster1[$i] == 0){
47     $IR += 0;
48   }elsif($cluster2[$i] == 0){
49     $IR += ($cluster1[$i] * log((2 * $cluster1[$i])/($cluster1[$i])));
50   }else{
51     $IR += ($cluster1[$i] * log((2 * $cluster1[$i])/($cluster1[$i] +
         $cluster2[$i])));
52   }
53
54   if($cluster1[$i] == 0){
55     $IR += ($cluster2[$i] * log((2 * $cluster2[$i])/($cluster2[$i])));
56   }elsif($cluster2[$i] == 0){
57     $IR += 0;
58   }else{
59     $IR += ($cluster2[$i] * log((2 * $cluster2[$i])/($cluster1[$i] +
         $cluster2[$i])));
60   }
61 }
```

```
62
63 #IR距離をファイルに記述
64 open(OUT, ">␣../NB/distance")
65   or die "Can't␣open␣distance:␣$!";
66 print OUT ("$IR");
67 close(OUT);
```

ソースコード A.7: 分布間の IR 距離を求めるプログラム