修士学位論文

教師付き外れ値検出手法を利用した新語義の発見

平成23年度

茨城大学大学院理工学研究科

情報工学専攻

江口晃

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

本論文では、教師データの重み付きの Local Outlier Factor(LOF) による外れ値検出手法を提案して、それを新語義の検出に応用した。

自然言語処理では、多義語の語義を識別することが必要となる。文脈中の多義語の識別は、近年需要の上昇している機械翻訳で利用される。語義識別には辞書データを利用するが、辞書には載っていないような語義で使われている単語がある。このような辞書に登録されていない語義をその単語の「新語義」と呼ぶ。この新語義を検出することで、辞書に新しい語義を追加することができる。また、新語義の検出は単語の誤り検出にも応用できる。

新語義は既存の語義とは異る文脈中で使われるので、他の語義とは独自の文脈パターンを持っている。そのため検出を行う対象の新語義は、その用例集内では外れ値となっていると考えられる。

語義識別に用いるデータに対する新語義の検出では、既存の語義にはその語義の教師データが存在する。しかし従来の外れ値検出の手法は教師データの利用を考えられた手法ではない。この場合、新語義検出の教師データは正常値である既存語義のクラスが明確であり、また外れ値である新語義の定義も明確である。このように、語義識別に用いるようなデータでは、通常の外れ値検出とは異なる特徴がある。

本論文では、新語義の検出に教師データを利用した外れ値検出手法を提案した。LOF の近傍値パラメータ k から、教師データを k+1 倍したデータを作った。この k+1 倍した教師データとテストデータを合わせた全データによる教師付き LOF 手法で新語義の検出を行い、従来の外れ値検出手法より精度の高い新語義の検出を目指した。

実験では、従来の外れ値検出手法である One Class SVM、LOF、Iuenced Outlierness(INFLO)、Nearest Neighbor-base method(NN-base) と教師付き LOF による、SEMEVAL-2 の課題語に対しての新語義の検出を行い比較をした。実験の結果、一部のデータでは教師付きの効果が表れたが、検出率の向上はしなかった。

今後の課題は、新語義が近傍データに教師データを含む最適な近傍パラメータ k の設定である。またデータの局所密度比による閾値で新語義が検出できるような拡張も必要である。

1.2 本論文の構成

第 2 章では、新語義を検出する手法として、WSD の手法について述べる。

第 3 章では、新語義を検出する手法として、外れ値検出の手法について述べる。Local Outlier Factor(LOF)、Influenced Outlierness(INFLO)、Nearest Neighbor-base method(NN-base) についての説明をする。

第 4 章では、教師付きの外れ値検出手法として、教師付き LOF について説明する。

第 5 章では、各手法による実験を行う。まず人工的なデータによる外れ値検出の予備実験を行う。次に SEMEVAL-2 の課題語を使った新語義の検出実験を行う。

第 6 章では、実験の考察を行う。

第 7 章では、結論と今後の課題を述べる。

第2章　WSDによる検出手法

語義識別 (Word Sense Descrimination：WSD) とは、対象となる単語を既存の語義中から何れかの語義で使われているかを識別するタスクである。語義識別では通常、対象となる単語は既存の何れかの語義に識別されてしまう。そこで、WSD の手法を利用した新語義の検出を考えてみる。

WSD の手法としてサポートベクターマシン (Support Vector Machine：SVM) がある。サポートベクターマシンは、線形入力素子を利用して、2 クラスのパターン識別器を構成する手法である。訓練サンプルから、各データ点との距離が最大となる分離平面（超平面）を求めるマージン最大化という基準で線形入力素子のパラメータを学習する。

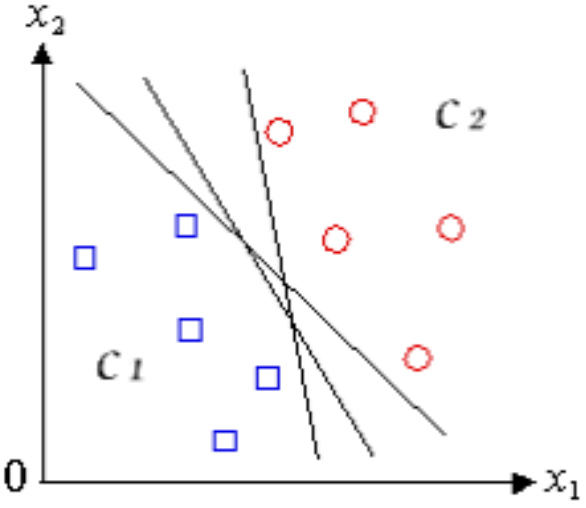

図 2.1: 2 クラスの識別をする超平面

図 2 は 2 つのクラス C_1、C_2 がある。2 つのクラスを識別する超平面はたくさんある。SVM ではいくつかの点を分離することができる幾多の候補平面の中で、マージンが最大になる超平面を探すことができる。

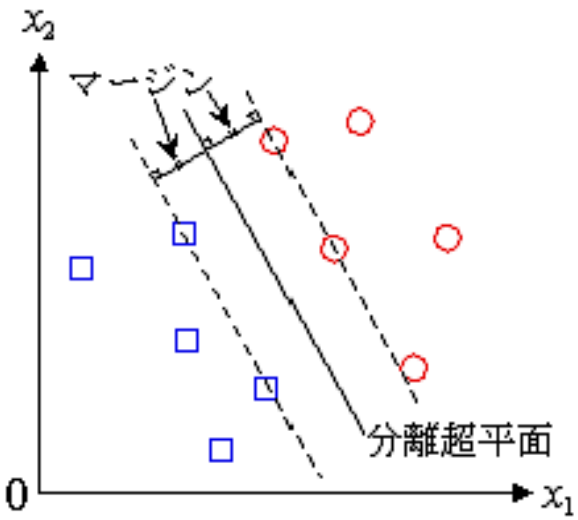

図 2.2: 超平面のマージン

この SVM による 2 クラスの識別手法を利用して新語義を検出する。ある語義のクラスを A、B、C と仮定する。それぞれのクラスに対して SVM による 2 クラスの語義識別を行う。新語義はどの語義クラスにも属さないので、それぞれのクラスの識別で、not A、not B、not C と識別されると考えられる。全てのクラスで識別されなかったデータを新語義と見なして検出する。

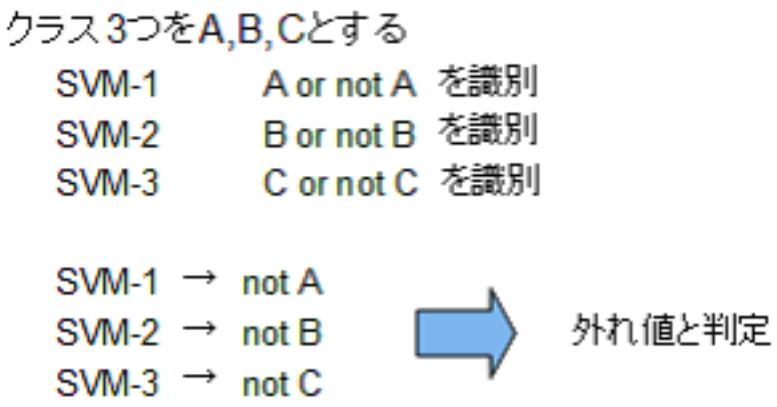

図 2.3: SVM による新語義の検出

第3章　外れ値検出の手法

3.1　One Class SVM

外れ値検出の手法として、One Class SVMがある。One Class SVMはSupport Vector Machine(SVM) を用いた教師なしクラスタリング手法である。データを多次元空間内の点とみなしたときに、データ集合の領域を求め、その領域に入っていないデータを外れ値とみなす。Gaussian カーネルを用いると入力空間で他の点から孤立している点は原点付近に写像される性質を用い、原点とデータ集合を分ける超平面を求める。このとき原点と超平面との距離が最小になるデータ点との距離 (マージン) を最小化するようにパラメータを最適化する。

データ集合のクラスを+1、原点のクラスを-1 として、分離する超平面を求める。原点のクラスに分類されたデータを外れ値として、新語義であるとみなす。

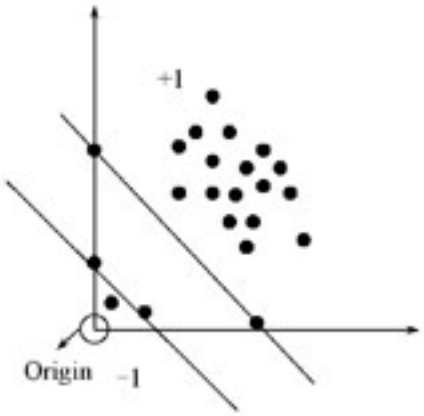

図 3.1: One Class SVM

3.2 Local Outlier Factor

Local Outlier Factor(LOF)[1]はデータ間の局所的な密度分布を用いて外れ値を検出する手法である。あるデータから k 個の最近傍データの距離密度を局所密度とする。この局所密度を k 個の最近傍データそれぞれの局所密度と比較し、周辺の密度より低い密度の場合、このデータは外れ値である可能性が高くなる。

図 3.2 は疎なクラスタ $C1$ と密なクラスタ $C2$、アウトライヤー O_1、O_2 がある。距離ベースによる外れ値検出の場合、アウトライヤー O_1 は検出できるが、密なクラスタ C_2 の近くにある O_2 は検出できない。しかし LOF では局所的なアウトライヤーを決定できるので、O_1 も O_2 も検出できるという利点がある。

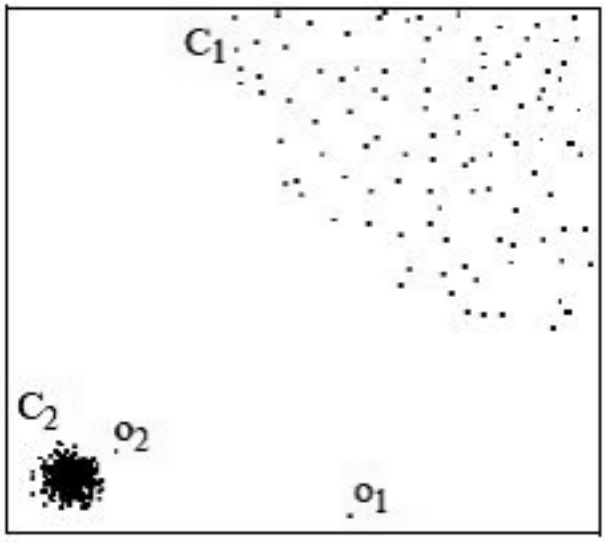

図 3.2: 密度の異なるクラスタの外れ値

[1]Wikipedia : Local outlier factor $http://en.wikipedia.org/wiki/Local_outlier_factor$

図 3.2 は $k = 3$ のときのデータ A の局所密度と最近傍データの局所密度を表したものである。データ A から近い順にデータ B、データ C、データ D となっている。データ A の最近傍データまでの距離は、A の最近傍データ三点の最近傍データまでの距離に比べて大きいことがわかる。最近傍データまでの距離が大きな値になるほど、局所密度は疎であると言える。

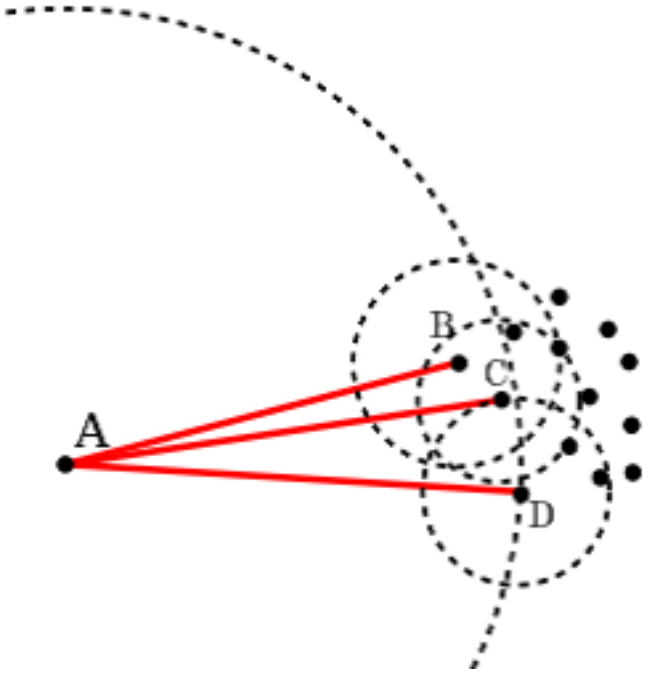

図 3.3: Local Outlier Factor

図 3.4 は $k = 3$ のときの局所密度を表したものである。$k = 3$ 番目までのデータを $N_k(A)$ とすると、データの局所密度は以下の式で得られる。

$$lrd(A) = 1/(\frac{\sum_{B \in N_k(A)} reachability - distance_k(A, B)}{|N_k(A)|})$$

最近傍データそれぞれの局所密度も同様に計算できる。LOF の数値はこれらの局所密度を用いて以下の式で得られる。

$$LOF_k(A) = \frac{\sum_{B \in N_k(A)} lrd(B)}{|N_k(A)|} / lrd(A)$$

LOF はデータの局所密度と最近傍データの局所密度の平均の平均による比で求められる。LOF が 1 に近い値のときは近傍のデータと密度が同じであることを表し、1 よりもはるかに大きい値の時にはそのデータは外れ値である可能性が高いといえる。

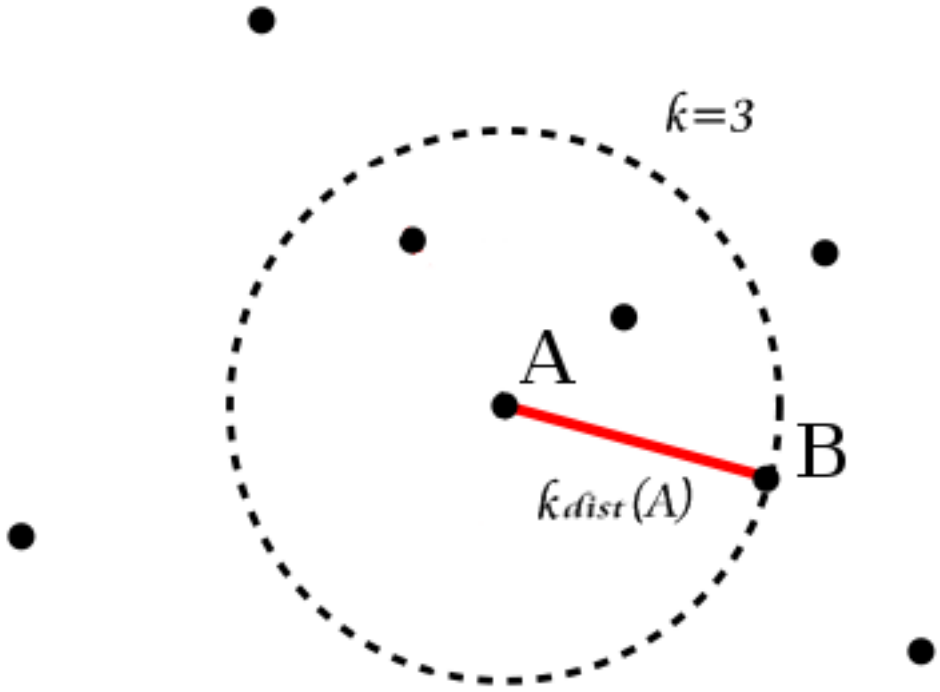

図 3.4: k=3 のときの局所密度

3.3 Influenced Outlierness

Influenced Outlierness(INFLO)[2] は、LOF の欠点を補った最近傍データの局所密度を用いた密度ベースの外れ値検出手法である。測定するデータの最近傍データ (Nearest Neighbors : NNs) と、測定するデータを NNs に含むデータ (Reverse Nearest Neighbors : RNNs) の局所密度の比によって外れ値を比較する。

図 3.3 は密なクラスタ C_1 と疎なクラスタ C_2、クラスタ C_2 のデータ p、アウトライヤーのデータ q、r、u がある。$k = 3$ の LOF の場合、q はアウトライヤーとして検出できるが、疎なクラスタ C_2 内にあるデータ p なども誤って外れ値として検出してしまうことになる。またデータ r を外れ値として検出しようとした場合、最近傍データに同じく外れ値のデータ u を含んでいるために、上手く検出できない恐れがある。

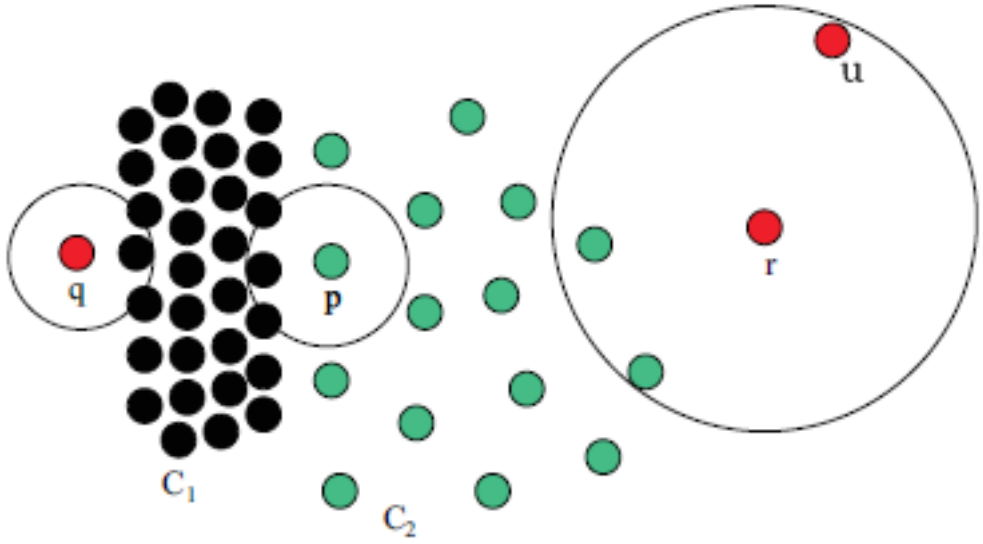

図 3.5: LOF の問題点

そこで、測定したいデータを最近傍に含むデータに注目する。データ q を最近傍に含むデータは存在しないが、データ p を最近傍に含むデータは同じクラスタ内に s と t が存在する。またデータ r を最近傍に含むデータは同じく外れ値のデータ u が存在する。

このようなデータを最近傍データ (Nearest Neighbors : NNs) に対し Reverse Nearest Neighbors(RNNs) と言う。NNs と RNNs それぞれの局所密度を使うことで、LOF で誤検出を起こすデータでも正しく検出できるようになる。

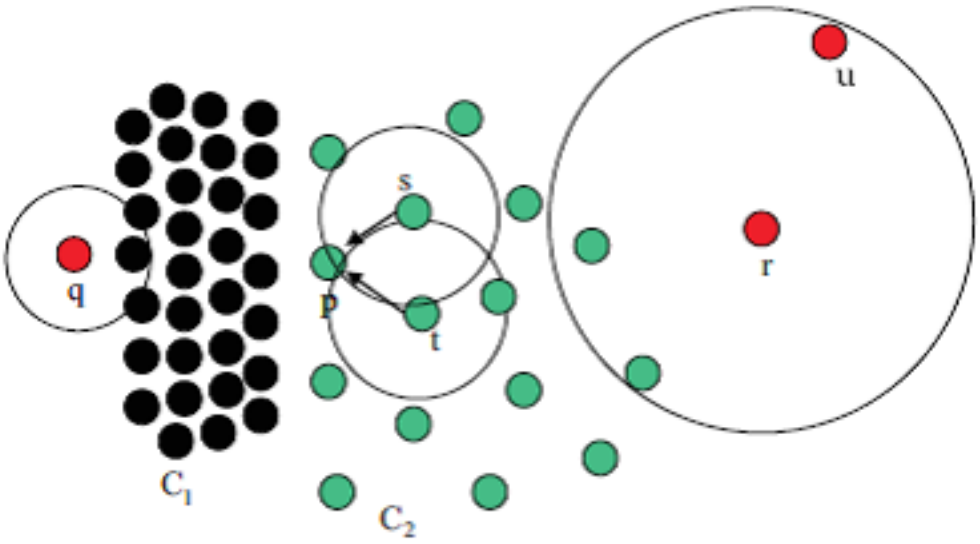

図 3.6: Influenced Outlierness

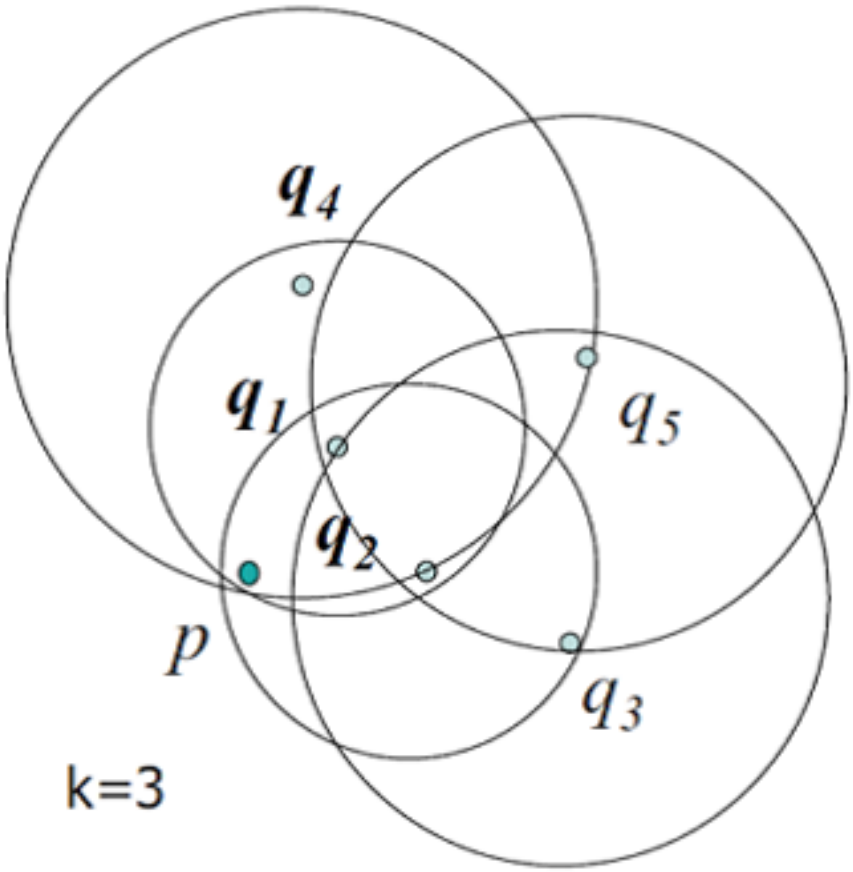

図 3.7: RNN と Influence Space

図3.3はデータp、q_1、q_2、q_3、q_4、q_5と、k=3のときのNNsである。それぞれ$NN_k(q_1) = p, q_2, q_4$、$NN_k(q_2) = p, q_1, q_3$、$NN_k(q_3) = q_1, q_2, q_5$、$NN_k(q_4) = p, q_1, q_2, q_5$、$NN_k(q_5) = q_1, q_2, q_3$となっている。$p$のRNNsは、$RNN_k(p) = q_1, q_2, q_4$となる。

データpのNNsの集合とRNNs集合の和をInfluence Space(IS)で表す。

$$IS_k(p) = NN_k(p) \bigcup RNN_k(p)$$

3.3ではpのNNsとRNNsは、$NN_k(p) = q_1, q_2, q_4$、$RNN_k(p) = q_1, q_2, q_4$なので、$IS_k(p) = q_1, q_2, q_4$となる。図3.3では$NN_k(p) = RNN_k(p)$となっているが、NNsとRNNsは必ずしも同じになるとは限らない。

p の局所密度を $den(p)$ とすると、$den(p)$ は k 番目のデータまでの距離の逆数で求められる。

$$den(p) = 1/k_{dist}(p)$$

外れ値の度合いを測る数値 INFLO は以下の式で求められる。

$$INFLO_k(p) = \frac{den_{avg}(IS_k(p)}{den(p)}$$

このとき

$$den_{avg}(IS_k(p)) = \frac{\sum_{o \in IS_k(p)} den(o)}{IS_k(p)}$$

である。

INFLO は NNs の局所密度と IS の局所密度の平均による比で求められる。INFLO が 1 に近い値のときは近傍のデータと密度が同じであることを表し、1 よりもはるかに大きい値の時にはそのデータは外れ値である可能性が高いといえる。

3.4 Nearest Neighbor-base method

Nearest Neighbor-base method(NN-base)[1] の手法は教師データを利用した、対象データの局所密度と教師データの局所密度を用いた密度ベースの外れ値検出手法である。トレーニングデータとの局所密度の比を取ることで、データが与えられたトレーニングデータに属するのか、別のクラスに属するのかの確率を数値で求めることができる。

図 3.4 ではデータ x とトレーニングデータに含まれる t、t' がある。データ x からの最近傍のデータは t で、x と t の距離は d_{xt} である。データ t からの最近傍のデータは同じトレーニングデータ内の t' で、t と t' の距離は $d_{tt'}$ である。

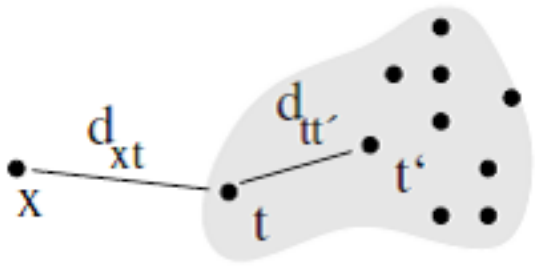

図 3.8: Nearest Neighbor-base

NN-base の値は最近傍データの距離の比によって以下の式で求められる。

$$pNN(x) = \frac{d_{xt}}{d_{tt'}}$$

$pNN(x)$ の値によりデータが外れ値であるかを判定する。$pNN(x)$ の最も低い閾値は $pNN(x) = 1$ であり、値が 1 より大きいほどそのデータは外れ値である可能性が高いといえる。

第4章　教師付き外れ値検出の手法

教師付き LOF とは教師データの重みを変えた、LOF による密度ベースの外れ値手法である。通常の LOF に教師データの情報を利用することで、検出率の向上を目指した。

通常の LOF による外れ値検出を用いる場合、教師データとテストデータによるデータの区別はなく、同じデータとして扱うことになる。教師データとテストデータを合わせた全データより局所密度を求めることによって LOF を計算する。しかし新語義の検出で、予め既存の語義のテストデータがある場合、この手法ではせっかくの教師データの情報を全く利用することができない。

図 4 では、教師データもテストデータと同様に扱っている。データ A を見ると、データ A は最近傍データに教師データを含んでいるのだが、k=3 の近傍データまで k_{dist} は大きくなる。

図 4.1: 教師データを利用しない LOF

新語義の検出に用いる教師データには予め語義のクラスラベルが付いているので、外れ値ではないことがわかっているデータである。そこで、教師データを $k+1$ 倍して予め重み付けしたデータによる LOF 手法を提案する。教師データの重みを変えることによって、教師データ近傍のデータの LOF 値を下げることができると考える。図 4 では、データ A の近傍に教師データ B が含まれる。教師データ B は k+1 倍されているので、同じ点が k+1 個あることになる。よってデータ A の k_{dist} は教師データまでとなる。教師データを NNs に含むような教師データの場合、教師データまでの距離が小さいデータの場合、LOF の値は低くなると予想できる。

また、全ての教師データは NNs に自身と同じデータを持つ。よって局所密度の値は全て等しくなり、LOF の値は LOF=1 となり外れ値より除外される。

図 4.2: 教師データを利用した LOF

第5章　実験

5.1　予備実験

5.1.1　実験概要

予備実験では人工的なデータに対しての従来の外れ値検出手法と教師付きLOF手法での外れ値検出実験を行った。また、SVMによる外れ値の検出数は2個と少なかったが、外れ値困難であることを示した。それぞれの手法での外れ値検出率の比較を行い動作を確認した。

実験には以下のモデルを利用してデータを作成した。各次元は独立で、分散共分散行列は対角行列である。

$$\int(x) = \frac{1}{(2\pi)^{n/2}|\sum|^{1/2}} exp - \frac{1}{2}(x-\mu)^t \sum (x-\mu)$$

$$\sum = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix}$$

３つの５次元正規分布を用意して、各モデルからそれぞれ200個のデータを生成した。うち各20個を教師データとした。

生成された全データ内の最小値、最大値を範囲とする一様分布からデータを20個生成し、外れ値データとした。

以上の教師データ60個（各クラス20個)、テストデータ560個（各クラス180個、外れ値20個）による外れ値検出実験を行った。

5.1.2 実験結果

実験結果は表 5.1.2 のようになった。評価には以下に定義した適合率、再現率、F 値を用いた。

$$\text{適合率 (Precision)} = \frac{\text{正解数}}{\text{抽出数}}$$

$$\text{再現率 (Recall)} = \frac{\text{正解数}}{\text{検出目標のデータ数}}$$

$$\text{F 値} = \frac{2}{\frac{1}{\text{再現率 (Recall)}} + \frac{1}{\text{適合率 (Precision)}}}$$

予備実験では LOF、INFLO、NN-base、教師付き LOF では数値の高い上位 20 個のデータを外れ値として検出した。密度ベースの外れ値手法において高い適合率、再現率を確認できた。

SVM による検出では、適合率は高いものの、正解数はわずか 2 個と低い再現率になった。SVM による外れ値以外のテストデータ 540 個に対する正解率は 100%であった。外れ値の各クラスでの識別結果を見ると、多くの外れ値は何れかのクラスに高い信頼率で識別されていた。これは、SVM による 2 クラスの識別が各クラスの分離平面までの距離で決まるためだと考えられる。信頼度の高い WSD による識別で、外れ値の検出は可能であった。さらに検出数を上げるために工夫が必要である。

教師付き LOF による検出では、LOF に比べて教師データの重みを増やしたことによって LOF の値の幅が広がり、外れ値の LOF 値増加が顕著になった。上位 20 個のデータに含まれなかった外れ値に関しても、数値の大幅な上昇が見られたデータがあった。

表 5.1: 予備実験の実験結果

手法	抽出数	正解数	適合率	再現率	F 値
SVM	2	2	1	0.1	0.182
One-Class-SVM	30	12	0.400	0.600	0.480
LOF	20	18	0.900	0.900	0.900
INFLO	20	19	0.950	0.950	0.950
NN-Base	20	16	0.800	0.800	0.800
教師付き LOF	20	17	0.850	0.850	0.850

5.2 実験

5.2.1 実験概要

実験では実際の文書データに対してそれぞれの手法で外れ値検出を行い、新語義が検出できるか実験した。それぞれの手法での検出率の比較を行った。

実験にはSemEval-2の日本語辞書タスクで課題とされた名詞・形容詞・動詞48単語を使用した。このタスクは各単語に対してそれぞれ50用例の教師データと50用例のテストデータがある。これらの単語は語義の頻度分布のエントロピーを考慮して選定されており、語義判別が容易なものから困難なものまでバランスよく選定されている。

課題語の「可能」と「入る」は教師データに未知語がある特殊なデータの為除外した。

表 5.2: SemEval-2の課題単語

名詞…21単語				
相手	意味	関係	技術	経済
現場	子供	時間	市場	社会
情報	手	電話	場合	はじめ
いれる	関係	いきる	たかい	はじめる
場所	一	文化	他	前
もの				
動詞…22単語				
あう	あげる	与える	生きる	入れる
教える	考える	すすめる	する	出す
たつ	出る	とる	乗る	はじめる
開く	見える	認める	見る	持つ
求める	やる			
形容詞…5単語				
大きい	高い	強い	早い	いい

実験は課題語 48 単語の各単語に対してモデルを作って行い、それぞれの単語での数値の合計で F 値を求めた。LOF、INFLO、NN-Base、教師付き LOF では各単語の上位 5 個のデータを外れ値と見なして検出している。
テストデータ 2400 用例のうち 8 単語 16 用例が新語義となっている。新語義のある単語は表の通りである。

表 5.3: 検出目標の新語義

名詞「意味」…1 用例
-このような、互いの意識の開きが、ある**意味**で、科学技術と社会に関する世論調査…
名詞「手」…3 用例
-その他、入院収益、その他医業収益等は**手**入力 -・以前は、本部での集約も手入力、… -経理コンピュータへの予算入力も**手**入力で、ミスもありましたが…
名詞「前」…7 用例
-ランチ＝**前**十一時半～後３時。 -二十三日月祝、二十四日火、**前**十時～後７時（二十四日は８時まで)、… -来年３月二十日木までの**前**十時～後十時、東京都豊島区東池袋… -大沢悠里のゆうゆうワイド（ＴＢＳ＝**前**８・三十）　竹下景子をゲストに… -三十日水までの**前**十一時半～後２時半、… -十九日土～十二月二十日火、**前**十一時半～後２時半、… -**前**十時半と後６時、本館１階正面口で「ふるまいボレーション」を開催。
動詞「もとめる」…1 用例
-…送電網の未整備などインフラ不安に要因を**求め**、その強化対策を打ち出している。
動詞「あげる」…2 用例
-シンガポールは六十五年の独立後、国を**挙げて**緑化を進めた。 -国を**あげて**緑化に取り組んだシンガポールは、街中に花があふれる。
動詞「はじめる」…2 用例
-ただし、動作などをあらわす「**はじめる**・はじまる」は「初」でなく、 -「**始める**・始まる」と書きます。

5.2.2 実験結果

実験結果は表5.2.2のようになった。LOF、INFOL、NN-base、教師付きLOFによる検出は、各単語における上位5個のデータとした。LOFとINFOLは教師データを差し引いたデータの上位５つの数値を括弧内に示した。教師データの重みを増やすことで教師データのLOF値が低くなり、一部のデータで外れ値の検出率の向上が見られた。

表 5.4: 新語義検出の実験結果

手法	抽出数	正解数	適合率 (Precision)	再現率 (Recall)	F 値
One-Class-SVM	755	2	0.003	0.125	0.005
LOF	240	0(2)	0.000(0.008)	0.000(0.125)	0.000(0.016)
INFLO	240	0(2)	0.000(0.008)	0.000(0.125)	0.000(0.016)
NN-Base	240	5	0.021	0.313	0.039
教師付き LOF	240	2	0.008	0.125	0.016

第6章　考察

実験ではLOF、INFLO、NN-Base、教師付きLOFの外れ値を、各単語で数値の大きい上位５個のデータに設定した。しかし全単語に新語義があるわけではないため、これを最適なLOF値による閾値に設定できれば、さらに高い検出精度が得られる可能性がある。また他の新語義検出データに応用しやすいはずである。しかしLOF値は各単語のデータにより大きく変化するため、LOFの数値による設定は難しい。

実験結果で教師付きLOFの正解検出を見てみると、検出できた2用例は新語義は動詞「あげる」の新語義であった。「あげる」の新語義はNNsに教師データを含んでいるので、LOFの数値が高くなり検出された。教師付きLOFの場合、NNsに教師データを含む未知語はLOFの数値が高くなりやすいので、この部分においては本手法の効果があったといえる。

教師付きLOFで検出できなかった新語義を見てみると、名詞「前」ではLOF値が全ての新語義で低く、LOF値が全データで一番低い未知語もあった。これらの未知語のNNsを見てみると、NNsに教師データが含まれていなかった。

新語義以外でLOF値の高かった誤検出データを見てみると、誤検出データはNNsに教師データを含んでいて、尚且つ教師データとの距離が大きいデータであることがわかった。

同じく教師データを利用しているNN-Baseでは、教師付きLOFでは抽出できなかった新語義を検出している。NN-Baseでは、教師付きLOFでLOF値の低かった名詞「前」の新語義を検出している。これは新語義の近傍データにテストデータが多かったため、LOF値が上手くでなかったためだと考えられる。

実験結果での未検出の原因として、kの値が最適でなかったことが考えられる。教師付きLOFを用いる際には新語義のNNsに教師データが含まれている必要がある。kが小さいとNNsに教師データを含むデータだけが数値が大きくなってしまい、外れ値を抽出できないことがある。また、外れ値と正常値のデータ間の距離にあまり差が出なかったことが考えられる。予備実験では偏ったパターンのデータを均等に準備していたが、SEMEVAL-2のデータでベクトル化したときの素性の取り方や、データ数自体の少なさに問題があった。新語義の抽出にはベクトル化する際の素性の取り方や距離の測り方に工夫が必要である。

第7章　結論

本研究では、新語義の検出に外れ値検出手法を利用し、教師データをk+1倍したデータによるLOF手法を提案した。また、実際に新語義の検出を行い他の外れ値検出手法と比較をした。

実験ではSEMEVAL-2の辞書タスク48単語の語義識別問題を題材に、本手法と従来の外れ値検出手法であるLOF、INFLO、NN-Baseによる新語義検出を行い新語義検出の効果を調べた。本手法では教師データの重みを変えることにより、一部の単語データで従来の外れ値検出手法では検出できなかった新語義検出の検出は見られたが、検出率の向上は見られなかった。これはNNsに教師データの含まない新語義データのLOF値が低くなり抽出できないためだと考えられる。

今後の課題は、新語義がNNsに教師データを含むkのパラメータの設定である。また今回はLOFの数値による上位５個のデータを検出データとしたが、LOF値による閾値で新語義が検出できるような拡張も必要である。

関連図書

[1] Katrin Erk : Unknown word sense detection as outlier detection , HLT-NAACL '06 Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, 2006,

[2] Wen Jin, Anthony K. H. Tung, Jiawei Han, and Wei Wang : Ranking Outliers Using Symmetric Neighborhood Relationship. Lecture Notes in Computer Science, 2006, Volume 3918/2006, 577-593,

[3] 新納浩幸, 佐々木稔 : 教師付き外れ値検出による新語義の発見, 言語処理学会第 17 回年次大会, E3-7 , 2011,

付 録A　プログラムソースリスト

■ベクトル間の距離を計算するプログラム\\

```
#!/usr/bin/perl
#   @ARGV 配列に引数格納    $ARGV[0] に数字が格納
$code=$ARGV[0];
$v_n=100;#vector の個数
open(IN,"vlist-$code.xml");
open(OUT,">>dist-$code.xml");

$max_v=0;
$i=0;
##ファイルから一行ずつ読み込んで配列に入れる
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$vp=3;

while($sp_line[$vp] ne ""){
$find1 = index($sp_line[$vp],":");
$v = substr($sp_line[$vp],0, $find1);
$n = substr($sp_line[$vp],$find1+1);
$vector[$i][$v] = $n;
if($v > $max_v){$max_v = $v;}
$vp++;
}
}


##各データ間の距離のベクトルを作る
```

```
for($i=1; $i<=$v_n; $i++){
print OUT "$i ";

for($t=1; $t<=$v_n; $t++){
#ベクトル距離計算
for($n=1; $n<=$max_v; $n++){
$va[$n]=$vector[$i][$n];
$vb[$n]=$vector[$t][$n];
$dist[$i][$t]+=($va[$n]-$vb[$n])*($va[$n]-$vb[$n]);
}

#ファイルへの書き込み
$S_dist= sprintf("%.3f",$dist[$i][$t]);
print OUT "$t:$S_dist ";
}

print OUT "\n";
}

close(IN);
close(OUT);
```

■NNs リストを作るプログラム\\

```
#!/usr/bin/perl
$n=$ARGV[1];
$code=$ARGV[0];
$v_n=100;
open(FH,"dist-$code.xml");
open(OUT1,">>NN$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす

#データを配列に入れる
$findP=0; #文字列検索の開始位置
for($t=1;$t<=$v_n;$t++){
$find1 = index($line,":",$findP);
$find2 = index($line,' ',$find1+1);
$vector[$i][$t] = substr($line,$find1+1, $find2-$find1-1);
$findP=$find2;
}

}

##距離の近い順にn個のデータを選ぶ
for($i=1;$i<=$v_n;$i++){
#初期値を設定
for($p=0;$p<=$n;$p++){$NN[$p]=100;$NNP[$p]=0;}
#ファイルへの書き込み
print OUT1 "$i ";

#距離の小さい順にn個を残していく
for($t=1;$t<=$v_n;$t++){
for($p=0;$p<=$n;$p++){
if($vector[$i][$t]<$NN[$p]){ #大小比較
for($c=$n;$c>$p;$c--){$NN[$c]=$NN[$c-1]; $NNP[$c]=$NNP[$c-1];} #繰り下げ代入
```

```
$NN[$c]=$vector[$i][$t];  $NNP[$c]=$t; #比較した値を代入
last;
}
}
}

#ファイルへの書き込み
for($p=1;$p<=$n;$p++){print OUT1 "NN$p=$NNP[$p]:$NN[$p] ";}
print OUT1 "\n";
}


close(OUT1);
close(FH);
```

■LOF を計算するプログラム\\

```
#!/usr/bin/perl
$n=10;
$code=$ARGV[0];
$v_n=100;

open(FH,"NN$n-$code.xml");
open(FH2,"NN$n-$code.xml");
open(OUT1,">>LOF$n-$code.xml");

##ファイルからクラスを読み込んで配列に入れる##
open(IN,"vlist-$code.xml");
$i=0;
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$trts[$i] = $sp_line[1];
$clas[$i] = $sp_line[2];
}
close(IN);
####

##■ den(p) を計算
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$i++; #配列番号を一つ増やす

#den(p) を配列に入れる
for($t=1;$t<=$n;$t++){
$find1 = index($line,"NN$t=");
$find2 = index($line,':',$find1+4);
$find3 = index($line,' ',$find2+1);
```

```
$num = substr($line,$find2+1, $find3-$find2-1);
if($num==0){$num=0.0001;}
$den[$i] += 1 / sqrt($num);
$find1=$find3;
}
}

##■ NNk の den(p) を計算、表示
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH2>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす


for($k=1;$k<=$n;$k++){
#データを配列に入れる
$find = index($line,"NN$k=");
$find1 = index($line,"=",$find);
$find2 = index($line,':',$find1);
$num = substr($line,$find1+1, $find2-$find1-1);

$den_k[$i] += $den[$num];
}

$lof = ($den_k[$i] / $n) / $den[$i];
print OUT1 "$i $trts[$i] $clas[$i] LOF=$lof den=$den[$i] den_k=$den_k[$i] \n";
}

close(OUT1);
close(FH1);
close(FH2);
```

■LOF をソートするプログラム\\

```
#!/usr/bin/perl
$n=10;
$code=$ARGV[0];
$v_n=100;
open(FH1,"LOF$n-$code.xml");
open(OUT1,">>LOFRank$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH1>)){
chomp($line);
$i++; #配列番号を一つ増やす

#一行を配列に入れる
$in_line[$i] = $line;

#INFLO を配列に入れる
$find1 = index($line,"LOF=");
$find2 = index($line,' ',$find1+4);
$inflo[$i] = substr($line,$find1+4, $find2-$find1-4);
}


#INFLO の値でソートする。
for($i=$v_n;$i>0;$i--){
for($j=2;$j<=$i;$j++){
if($inflo[$j]>$inflo[$j-1]){
$c_inflo=$inflo[$j];$inflo[$j]=$inflo[$j-1];$inflo[$j-1]=$c_inflo;
$c_line=$in_line[$j];$in_line[$j]=$in_line[$j-1];$in_line[$j-1]=$c_line;
}
}
}


for($i=1;$i<=$v_n;$i++){
print OUT1 "$in_line[$i]\n";
}
```

```
close(OUT1);
close(FH1);
```

■ IS を作成するプログラム\\

```
#!/usr/bin/perl
$n=10;
$code=$ARGV[0];
$v_n=100;
open(FH1,"NN$n-$code.xml");
open(OUT1,">>IS$n-$code.xml");


##■ IS を配列に入れる
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH1>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす

for($k=1;$k<=$n;$k++){
$find = index($line,"NN$k=");
$find1 = index($line,"=",$find);
$find2 = index($line,':',$find1);
$num = substr($line,$find1+1, $find2-$find1-1);

$IS[$i][$num]=1; #NN(p)
$IS[$num][$i]=1; #RNN(p)
}
}


##RNN 配列を書き込む
for($i=1;$i<=$v_n;$i++){
print OUT1 "$i ";

for($s=1;$s<=$v_n;$s++){
if($IS[$i][$s]==1){print OUT1 "IS=$s ";}
}
print OUT1 "\n";
}
```

```
close(OUT1);
close(FH1);
```

■ INFLO を計算するプログラム\\

```
#!/usr/bin/perl
$n=10;
$code=$ARGV[0];
$v_n=100;

open(FH1,"NN$n-$code.xml");
open(FH2,"IS$n-$code.xml");
open(OUT1,">>Inflo$n-$code.xml");

##ファイルからクラスを読み込んで配列に入れる##
open(IN,"vlist-$code.xml");
$i=0;
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$trts[$i] = $sp_line[1];
$clas[$i] = $sp_line[2];
}
close(IN);
####

##■ den(p) を計算
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH1>)){
chomp($line);
$i++; #配列番号を一つ増やす

#den(p) を配列に入れる
$find1 = index($line,"NN$n=");
$find2 = index($line,':',$find1+4);
$find3 = index($line,' ',$find2+1);

$num = substr($line,$find2+1, $find3-$find2-1);
```

```
if($num==0){$num=0.0001;}
$den[$i]=1 / sqrt($num);
}


##■ IS の den(p) を計算、表示
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH2>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす

#初期値の設定
$IS_c=0;
$findP=index($line,"IS=");

#データを配列に入れる
while($findP>=0){
$find1 = index($line,' ',$findP);
$num = substr($line,$findP+3, $find1-$findP-3);

$den_IS[$i] += $den[$num];

$IS_c++;
$findP=index($line,"IS=",$find1+1);
}

$inflo = ($den_IS[$i] / $IS_c) / $den[$i];
print OUT1 "$i $trts[$i] $clas[$i] INFLO=$inflo den=$den[$i] IS=$den_IS[$i] IS_c=$IS_
}



close(OUT1);
close(FH1);
close(FH2);
```

■ INFOL をソートするプログラム\\

```
#!/usr/bin/perl
$n=10;
$code=$ARGV[0];
$v_n=100;
open(FH1,"Inflo$n-$code.xml");
open(OUT1,">>InfloRank$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH1>)){
chomp($line);
$i++; #配列番号を一つ増やす

#一行を配列に入れる
$in_line[$i] = $line;

#INFLO を配列に入れる
$find1 = index($line,"INFLO=");
$find2 = index($line,' ',$find1+6);
$inflo[$i] = substr($line,$find1+6, $find2-$find1-6);
}


#INFLO の値でソートする。
for($i=$v_n;$i>0;$i--){
for($j=2;$j<=$i;$j++){
if($inflo[$j]>$inflo[$j-1]){
$c_inflo=$inflo[$j];$inflo[$j]=$inflo[$j-1];$inflo[$j-1]=$c_inflo;
$c_line=$in_line[$j];$in_line[$j]=$in_line[$j-1];$in_line[$j-1]=$c_line;
}
}
}


for($i=1;$i<=$v_n;$i++){
print OUT1 "$in_line[$i]\n";
}
```

```
close(OUT1);
close(FH1);
```

■ NN-Base の NNs を作成するプログラム\\

```
#!/usr/bin/perl
$n=1;
$code=$ARGV[0];
$v_n=100;
open(FH,"Dist-$code.xml");
open(OUT1,">>NNbaseNN$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす

#データを配列に入れる
$findP=0; #文字列検索の開始位置
for($t=1;$t<=$v_n;$t++){
$find1 = index($line,":",$findP);
$find2 = index($line,' ',$find1+1);
$vector[$i][$t] = substr($line,$find1+1, $find2-$find1-1);
$findP=$find2;
}

}

##距離の近い順に n 個のデータを選ぶ
for($i=1;$i<=$v_n;$i++){
#初期値を設定
for($p=0;$p<=$n;$p++){$NN[$p]=1000;$NNP[$p]=0;}

#ファイルへの書き込み
print OUT1 "$i ";

#距離の小さい順に n 個を残していく
for($t=1;$t<=$v_n;$t++){
if($t>=51){next;}#トレーニングデータ以外のスキップ
```

```
for($p=0;$p<=$n;$p++){
if($vector[$i][$t]<$NN[$p]){ #大小比較
for($c=$n;$c>$p;$c--){$NN[$c]=$NN[$c-1]; $NNP[$c]=$NNP[$c-1];} #繰り下げ代入

$NN[$c]=$vector[$i][$t];  $NNP[$c]=$t; #比較した値を代入
last;
}
}
}

#ファイルへの書き込み
for($p=1;$p<=$n;$p++){print OUT1 "NN$p=$NNP[$p]:$NN[$p] ";}
print OUT1 "\n";
}


close(OUT1);
close(FH);
```

■ NN-Base を計算するプログラム\\

```
#!/usr/bin/perl
$n=1;
$code=$ARGV[0];
$v_n=100;
open(FH,"NNbase-NN$n-$code.xml");
open(FH2,"NNbase-NN$n-$code.xml");
open(OUT1,">>NNbase$n-$code.xml");

##ファイルからクラスを読み込んで配列に入れる##
open(IN,"vlist-$code.xml");
$i=0;
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$trts[$i] = $sp_line[1];
$clas[$i] = $sp_line[2];
}
close(IN);
####


$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$i++; #配列番号を一つ増やす

#den(p) を配列に入れる
$find1 = index($line,"NN$n=");
$find2 = index($line,':',$find1+4);
$find3 = index($line,' ',$find2+1);

$num = substr($line,$find2+1, $find3-$find2-1);
if($num==0){$den[$i]=10000;}else{$den[$i]=1 / sqrt($num);}
```

```
}


$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH2>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす


if($i<=50){next;} #トレーニングデータのスキップ
#データを配列に入れる
$find = index($line," ");
$num2 = substr($line,0, $find);

for($k=1;$k<=$n;$k++){
#データを配列に入れる
$find1 = index($line,"NN$k=");
$find2 = index($line,':',$find1+4);
$num = substr($line,$find1+4, $find2-$find1-4);

$den_k[$i] += $den[$num];


}

$lof = ($den_k[$i] / $n) / $den[$i];
print OUT1 "$num2 $trts[$num2] $clas[$num2] LOF=$lof den=$den[$i] den_k=$den_k[$i] \n"
}

close(OUT1);
close(FH1);
close(FH2);
```

■教師データを k+1 倍したデータを作成するプログラム\\

```
#!/usr/bin/perl
#   @ARGV 配列に引数格納    $ARGV[0] に数字が格納
$code=$ARGV[0];
$n=$ARGV[1];
$v_n=100;#vector の個数
$tr_n=50;

open(IN,"./vlist/vlist-$code.xml");
open(OUT,">./supv/supv$n-$code.xml");

$max_v=0;
$i=0;
##ファイルから一行ずつ読み込んで配列に入れる
while(defined($line = <IN>)){
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$vp=1;
while($sp_line[$vp] ne ""){
$sup_v[$i][$vp]=$sp_line[$vp];
$vp++;
}
}


##各データ間の距離のベクトルを作る
for($i=1; $i<=$v_n; $i++){
print OUT "$i ";

$vp=1;
while($sup_v[$i][$vp] ne ""){
print OUT "$sup_v[$i][$vp] ";
$vp++;
}
print OUT "\n";
}
```

```
#tr データの点を k 個増やす
for($k=1;$k<=$n;$k++){
for($t=1; $t<=$tr_n; $t++){
print OUT "$i ";

$vp=1;
while($sup_v[$t][$vp] ne ""){
print OUT "$sup_v[$t][$vp] ";
$vp++;
}
print OUT "\n";
$i++;
}
}


close(IN);
close(OUT);
```

■教師付き LOF のベクトル間の距離を計算するプログラム\\

```
#!/usr/bin/perl
#    @ARGV 配列に引数格納    $ARGV[0] に数字が格納
$code=$ARGV[0];
$n=$ARGV[1];
$v_n=100+50*$n;#vector の個数
open(IN,"./supv/supv$n-$code.xml");
open(OUT,">./supd/supd$n-$code.xml");

$max_v=0;
$i=0;
##ファイルから一行ずつ読み込んで配列に入れる
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$vp=3;

while($sp_line[$vp] ne ""){
$find1 = index($sp_line[$vp],":");
$v = substr($sp_line[$vp],0, $find1);
$n = substr($sp_line[$vp],$find1+1);
$vector[$i][$v] = $n;
if($v > $max_v){$max_v = $v;}
$vp++;
}
}


##各データ間の距離のベクトルを作る
for($i=1; $i<=$v_n; $i++){
print OUT "$i ";
for($t=1; $t<=$v_n; $t++){
#ベクトル距離計算
if($i>100){$ic=$i%50; if($ic==0){$ic=50;} $dist[$i][$t]=$dist[$ic][$t];}
elsif($t>100){$tc=$t%50; if($tc==0){$tc=50;} $dist[$i][$t]=$dist[$i][$tc];}
```

```
else{
for($n=1; $n<=$max_v; $n++){
$va[$n]=$vector[$i][$n];
$vb[$n]=$vector[$t][$n];
$dist[$i][$t]+=($va[$n]-$vb[$n])*($va[$n]-$vb[$n]);
}
}
#ファイルへの書き込み
$S_dist= sprintf("%.3f",$dist[$i][$t]);
print OUT "$t:$S_dist ";
}

print OUT "\n";
}

close(IN);
close(OUT);
```

■教師付き LOF の NN を作成するプログラム\\

```
#!/usr/bin/perl
$code=$ARGV[0];
$n=$ARGV[1];
$v_n=100+50*$n;
open(FH,"./supd/supd$n-$code.xml");
open(OUT1,">./supNN/supNN$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす

#データを配列に入れる
$findP=0; #文字列検索の開始位置
for($t=1;$t<=$v_n;$t++){
$find1 = index($line,":",$findP);
$find2 = index($line,' ',$find1+1);
$vector[$i][$t] = substr($line,$find1+1, $find2-$find1-1);
$findP=$find2;
}

}

##距離の近い順に n 個のデータを選ぶ
for($i=1;$i<=$v_n;$i++){
#初期値を設定
for($p=0;$p<=$n;$p++){$NN[$p]=100;$NNP[$p]=0;}
#ファイルへの書き込み
print OUT1 "$i ";

#距離の小さい順に n 個を残していく
for($t=1;$t<=$v_n;$t++){
for($p=0;$p<=$n;$p++){
#if($vector[$i][$t]<$NN[$p]){ #大小比較
if($vector[$i][$t]<$NN[$p] || ($vector[$i][$t]==$NN[$p] && ($t<51&&$t>100))){
```

```
#大小比較
for($c=$n;$c>$p;$c--){$NN[$c]=$NN[$c-1]; $NNP[$c]=$NNP[$c-1];} #繰り下げ代入

$NN[$c]=$vector[$i][$t];  $NNP[$c]=$t; #比較した値を代入
last;
}
}
}

#ファイルへの書き込み
for($p=1;$p<=$n;$p++){print OUT1 "NN$p=$NNP[$p]:$NN[$p] ";}
print OUT1 "\n";
}


close(OUT1);
close(FH);
```

■教師付き LOF の NN を作成するプログラム\\

```
#!/usr/bin/perl
$code=$ARGV[0];
$n=$ARGV[1];
$v_n=100+50*$n;

open(FH,"./supNN/supNN$n-$code.xml");
open(FH2,"./supNN/supNN$n-$code.xml");
open(OUT1,">./supLOF/supLOF$n-$code.xml");

##ファイルからクラスを読み込んで配列に入れる##
open(IN,"./supv/supv$n-$code.xml");
$i=0;
while(defined($line = <IN>)){
chomp($line);
@sp_line = split(/\s/, $line);
$i++; #配列番号を一つ増やす

#データを配列に入れる
$trts[$i] = $sp_line[1];
$clas[$i] = $sp_line[2];
}
close(IN);
####

##■ den(p) を計算
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH>)){
chomp($line);
$i++; #配列番号を一つ増やす

#den(p) を配列に入れる
for($t=1;$t<=$n;$t++){
$find1 = index($line,"NN$t=");
$find2 = index($line,':',$find1+4);
$find3 = index($line,' ',$find2+1);
```

```
$num = substr($line,$find2+1, $find3-$find2-1);
if($num==0){$num=0.0001;}
$den[$i] += 1 / sqrt($num);
#$distNN += $num;
$find1=$find3;
}
#$den[$i] += 1 / sqrt(($distNN / $n));


#den(p) を配列に入れる
#$find1 = index($line,"NN$n=");
#$find2 = index($line,':',$find1+4);
#$find3 = index($line,' ',$find2+1);

#$num = substr($line,$find2+1, $find3-$find2-1);
#if($num==0){$num=0.0001;}
#$den[$i]=1 / sqrt($num);


}

##■ NNk の den(p) を計算、表示
$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH2>)){
chomp($line);
$line=$line . " ";
$i++; #配列番号を一つ増やす


for($k=1;$k<=$n;$k++){
#データを配列に入れる
$find = index($line,"NN$k=");
$find1 = index($line,"=",$find);
$find2 = index($line,':',$find1);
$num = substr($line,$find1+1, $find2-$find1-1);

$den_k[$i] += $den[$num];
}
```

```
$lof = ($den_k[$i] / $n) / $den[$i];
print OUT1 "$i $trts[$i] $clas[$i] LOF=$lof den=$den[$i] den_k=$den_k[$i] \n";
}

close(OUT1);
close(FH1);
close(FH2);
```

■教師付き LOF をソートするプログラム\\

```
#!/usr/bin/perl
$code=$ARGV[0];
$n=$ARGV[1];
$v_n=100+50*$n;
open(FH1,"./supLOF/supLOF$n-$code.xml");
open(OUT1,">./supLOFRank/supLOFRank$n-$code.xml");

$i=0;
#一行ずつファイルを探索する。
while(defined($line = <FH1>)){
chomp($line);
$i++; #配列番号を一つ増やす

#一行を配列に入れる
$in_line[$i] = $line;

#INFLO を配列に入れる
$find1 = index($line,"LOF=");
$find2 = index($line,' ',$find1+4);
$inflo[$i] = substr($line,$find1+4, $find2-$find1-4);
}


#INFLO の値でソートする。
for($i=$v_n;$i>0;$i--){
for($j=2;$j<=$i;$j++){
if($inflo[$j]>$inflo[$j-1]){
$c_inflo=$inflo[$j];$inflo[$j]=$inflo[$j-1];$inflo[$j-1]=$c_inflo;
$c_line=$in_line[$j];$in_line[$j]=$in_line[$j-1];$in_line[$j-1]=$c_line;
}
}
}


for($i=1;$i<=$v_n;$i++){
print OUT1 "$in_line[$i]\n";
```

```
}

close(OUT1);
close(FH1);
```