自然言語処理
―単語ベクトル―

https://satoyoshiharu.github.io/nlp/

単語ベクトルと、100本ノックの第7章の位置づけ

- ニューラルネットを使って、単語ベクトルを作成します。どういう分かち書きを利用し、どういう単語ベクトルが欲しいかは、応用目的によるので、結局自作できるようにしていないとアプリが組めません。自作できるだけの予備知識を以下で説明します。

- 一方、100本ノックの第7章は、単語ベクトルがすでにあるとして、それで何ができるかを知る的な課題です。第7章の課題にはクラスタリングや次元圧縮といった機械学習の使用も含まれます。

自然言語処理：
なぜ単語ベクトル？

解説動画

ここでは、自然言語をニューラルネットで処理するときに、単語ベクトルというものを利用するのですが、なぜそれが必要かを見ていきます。

連続量と離散量

- 連続量
  - 例
    - 身長：170cmと170.11cmとの間に無限に取りうる値がある。
    - 体重
    - カメラから写っている物体までの距離
    - 画像のあるピクセルのRedチャネルの量(floatの場合)
    - 加速度センサーのX軸の反応量
    - スピーカーの音量
    - ・・・
- 離散量
  - 例
    - さいころの目： 1 から 6 の間の値を飛び飛びでとり、それらの間の値を取りえない。
    - 何月か
    - 学生番号
    - トランプのマーク
    - 人の性別
    - コロナ検査で陽性か陰性か
    - ・・・

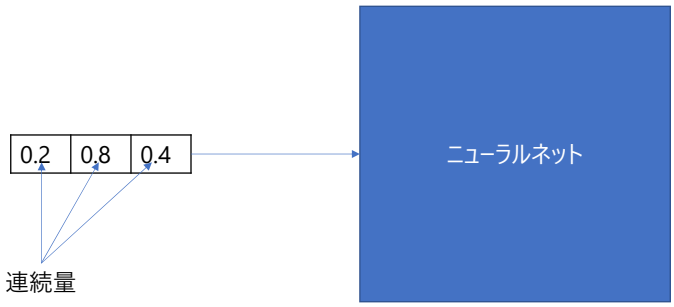
世の中の物量は、連続量と離散量とに分類できます。
例えば、身長は、170cmと170.11cmとの間に無限に取りうる値があります。
そのように値が連続的に取りうるものは連続量といいます。
一方、例えば、さいころの目は、 1 から 6 の間の値を飛び飛びでとり、それらの間の値を取りません。
そのように、値が非連続的なものを離散量といいます。

自然言語の処理単位は、離散的

- 文字：あ、い、山、川、…
- 単語：今日、明日、太郎、花子、…
- 文：今日は晴天です。明日は木曜日です。週末が楽しみです。
  …

自然言語で扱う文字、単語、文などは、どれもそれぞれユニークな値をとり、中間の値というのはありません。
つまり、離散量です。

ニューラルネットの入力は通常連続量のベクトル



ところが、ニューラルネットは通常連続量、実数のベクトルを入力とし、出力とします。
そこで、文字、単語、文などをどうデータ化するかという問題が出てきます。

One-hot-Vector

離散的なデータを、連続量（たまたま０か１）のベクトルに変換する

• 今日	1	0	0	0	0	0	…
• 明日	0	1	0	0	0	0	…
• 太郎	0	0	1	0	0	0	…
• 花子	0	0	0	1	0	0	…
• …							

⇒　ニューラルネットで取り扱いが可能になる。

最初に考えられたアプローチは、例えば単語の場合、単語数だけの長いベクトルを考え、
それぞれの単語はユニークな位置にフラグ１が立っていて、他はすべてゼロにする、という攻め方です。
これをOne-hotベクトルといいます。
１も０も実数ですので、これで、単語をニューラルネットの入出力にすることができます。

One-hot-Vector の問題点

- 文字数は、*常用漢字に限定しても*　2136 文字。
- 単語数は、通常の日常用語は 5 万語。Wikipediaなどは、数百万語～。
- 文の数は、構文的な自由度があり、再帰も可能なため、実質、無限大。

- 文字、単語、文のどれも、これらの個数だけ長いベクトルを、多数、利用するのは、メモリ効率が悪い。しかも、0 である領域が多すぎて、無駄。

ところが、単語数は膨大です。単語数だけの長いベクトルを大量に使おうとすると、
処理効率が悪くなります。ほとんど 0 なのでメモリ効率も悪くなります。

埋め込み（Embedding）
別名 Word2Vec
一般的には **単語の分散表現**

固定長の実数値ベクトルで、それぞれの単語などを表現する。

実数値は連続量なので、無限に多くの値をとりえて、さらにそのベクトルは実数を並べるので、表現力も十分にリッチ。

そこで、単語数よりもはるかに小さい、ある固定長の実数値ベクトルで、それぞれの単語を表現する、というアプローチが登場しました。
固定長のベクトルといっても、それぞれの要素は実数で無数に近い値をとりえます。それが複数あるので、
表現力に不足はありません。
この表現を、埋め込み、Embedding、別名Word2Vec、一般には単語の分散表現といいます。

単語の分散表現

ニューラルネットが処理できるように、単語を固定長の実数値ベクトルで表現したいが、その表現こそ、ニューラルネットを使って学習させる。この発想が妙、考えた人偉い。

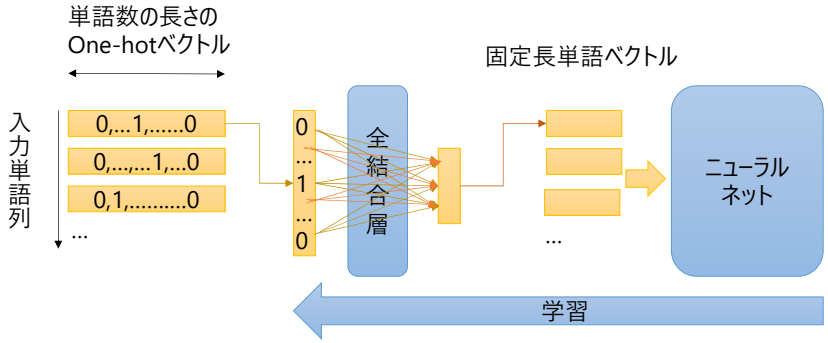
では、どのように、単語をコンパクトな埋め込み表現にするのか？
それは、ニューラルネットに単語の特徴を学ばせて、獲得します。
単語の埋め込み表現は、最初はランダムだとします。
それに対し、ある単語を与えたとき、正解ラベルがまた別の単語となるような学習問題を設定します。
その設定によって、ニューラルネットは単語の埋め込み表現を自動的に学習します。
ニューラルネットで学習させるので、埋め込み表現は、人が見ても何を表現しているのか不明です。
しかし、それらは、単語のさまざまな特徴をうまく表現できていることが知られています。
自然言語の応用をニューラルネットで書くとき、最初に単語を埋め込み表現へ変換
して処理に入ります。そのため、埋め込み表現は、必須の機能です。

自然言語処理：
単語ベクトルの導出ー埋め込み表、CBOW、Skipgram

解説動画

ここでは、単語などの埋め込み表現をいかにして得るかのテクニックを見ていきます。

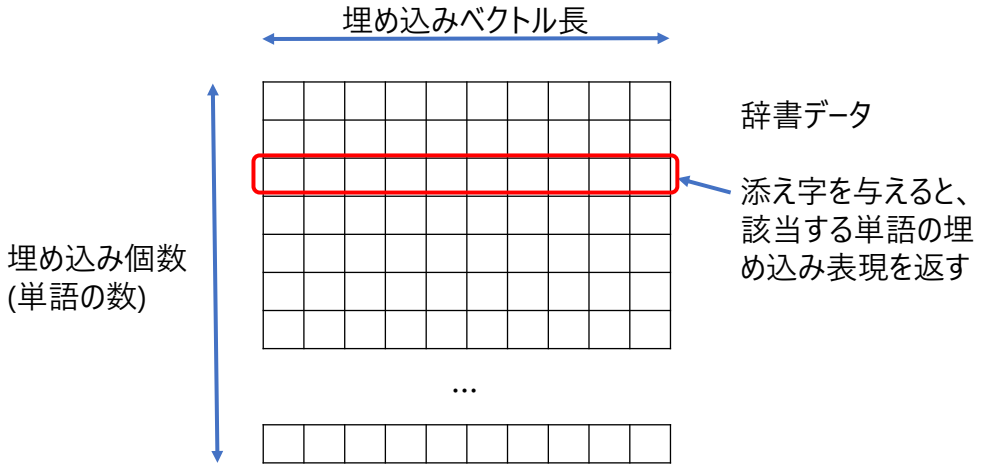
単語を固定長ベクトル(埋め込み)として学習する



結合線の組み合わせが大きい
（単語数 x 埋め込みサイズ）

単語埋め込み表現を、ニューラルネットの学習で獲得するストレートな方法は、
One-hotベクトルを入力して、全結合層を介して、固定長の埋め込み表現へ変換し、
何らかのニューラルネットに投入してトレーニングするやり方です。
出力もまた単語の埋め込み表現から、One-hotベクトルに戻します。
しかし、このやり方は、単語数と埋め込みベクトルサイズの組み合わせの重み行列が必要となります。
単語数は数百万まで行きますから、重み行列はとても大きな行列となり、効率的ではありません。

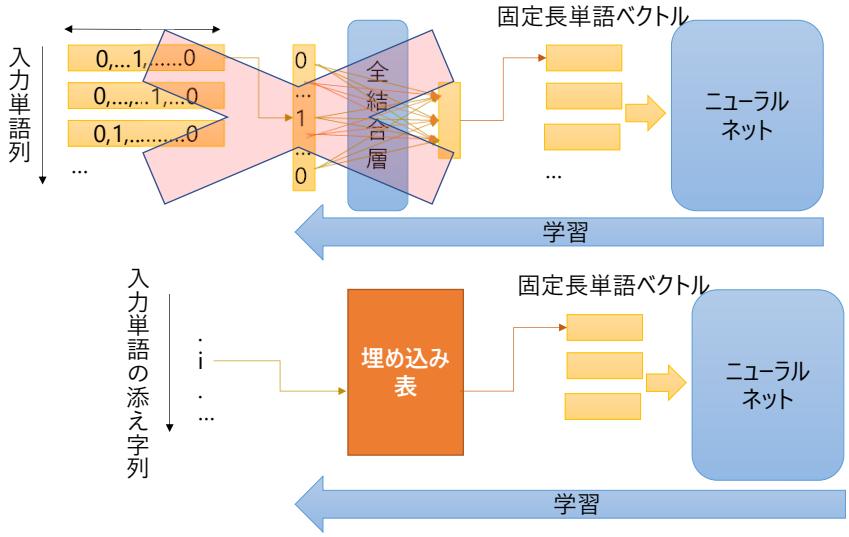
埋め込み表



PyTorch nn.Embedding（埋め込み個数、埋め込みベクトル長）

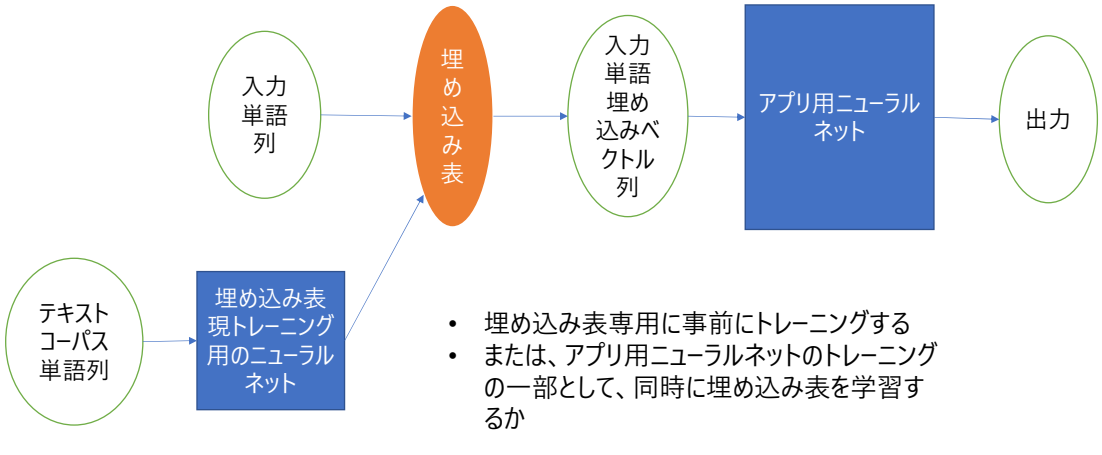
そこで、埋め込み表というものを考えます。
埋め込み表は、埋め込み表現ベクトルを単語数だけ縦に並べたものです。
それぞれの行がある単語に対応し、添え字を与えるとその埋め込みベクトルを取り出すことができます。

埋め込み表を学習すれば、単語数 x 埋め込みサイズの全結合は不要



そして、埋め込み表をニューラルネットの学習プロセスに組み込めば、
単語の埋め込み表現が自動的に得られます。
そうすれば、One-hotベクトルと全結合層という、非効率的な部分なしで、
埋め込み表現を獲得できます。

埋め込みベクトルの位置づけ



一般の自然言語を扱うニューラルネットでは、単語ベクトルの埋め込み表を事前にトレーニングして獲得しておいて、それを使って入力データを埋め込み表現に変換し、アプリ固有の処理に入ります。
また、アプリのニューラルネットのトレーニングの一部として、最初の処理に組み込んで置き、
アプリの学習の中で埋め込み表現の獲得も同時に行う場合もあります。
専用に事前にトレーニングするのが一般的なようです。

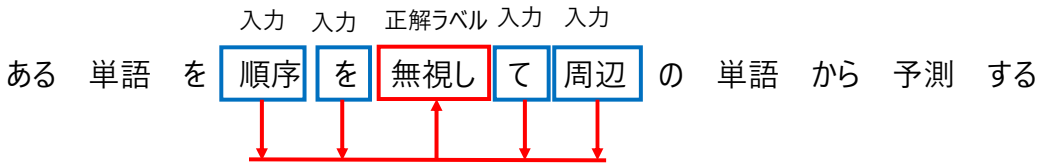
分布仮説

- 同じような文脈で使われる単語は、それらの意味が近い。
- 単語の意味の違いはその用法（周囲の単語との関係）で決められる。

  - 「日」
    - まだ日(day)が浅い
    - 日(sun)が照っている
  - 「甘・い」
    - 君はまだ甘い(naive)
    - このリンゴは甘い(sweet)よ

具体的な獲得手法を見ていきます。
自然言語処理のテクニックは、要素はコンテキストと密接に関係しているという考えに
基づくものが多いです。
これを分布仮説といいます。
同じような文脈で使われる単語は、それらの意味が近く、
単語の意味の違いはその用法（周囲の単語との関係）で決められる、
というった考え方です。
例えば、日という単語は、日にちの意味の時とお日様の意味のときがあります。これらの意味の違いは
使われるコンテキストによって決まります。

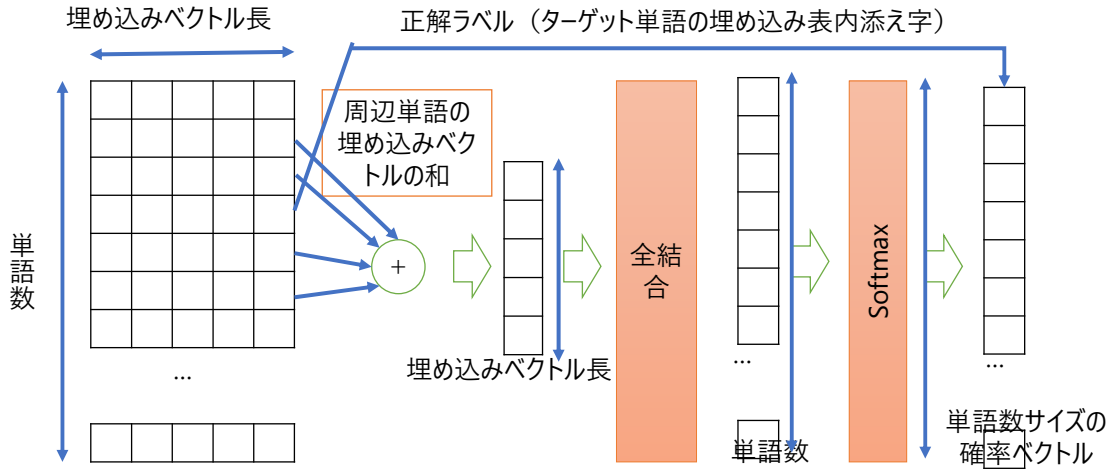
CBOW (Continuous Bag-Of-Words)



その学習は、教師なし学習っぽく、正解ラベルを
文データ自体から自動的に持ってきます。

分布仮説を使って、埋め込み表現を得るテクニックとして、CBOWとSkipGramがあります。
まずCBOWのほうは、文章を取り上げて、ある特定の単語をその前後の単語から推定するものです。
ニューラルネットに、前後の単語を入力し、中心の単語を予測させます。
正解ラベルは中心単語で、文章データがあれば、ラベルを人の手で作る必要はありません。

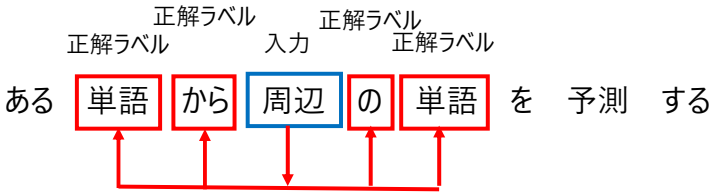
CBOWネットワーク構成



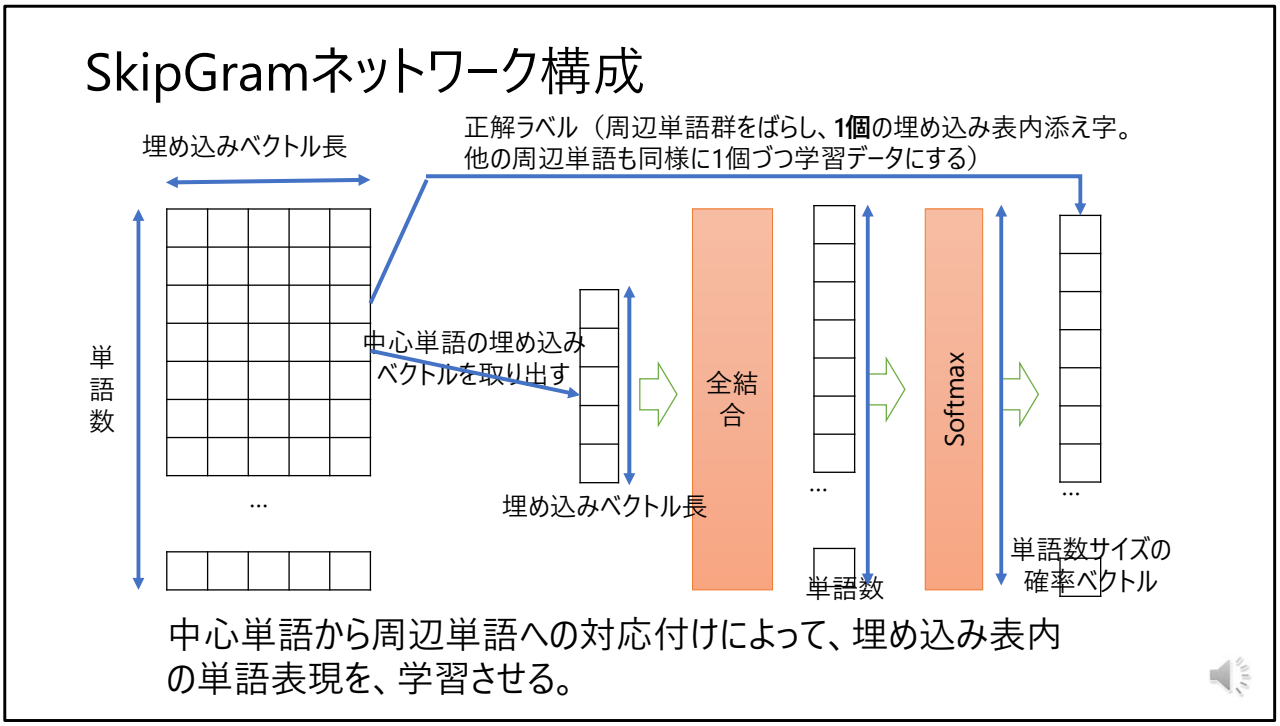
周辺単語の情報からターゲット単語への対応付けによって、埋め込み表内の単語表現を、学習させる。

CBOWの実装です。
複数の周辺単語は、中心単語のコンテキストになります。
コンテキストは、埋め込みベクトルの和で、表現します。
それに与えて、中心単語を予測させます。
今、特定の単語の予測は、全結合とSoftmaxで、実装するとします。
この構成で、学習を繰り返すと、周辺単語からターゲット単語を予測するのに最適な埋め込み表が形成されます。

SkipGram



次に、CBOWと逆の発想のテクニックがSkipGramです。
SkipGramは、中心単語から周辺単語を予測させるタスクで、埋め込み表現を学習させます。



SkipGramの実装は、ちょっとしたひねりがあります。
中心単語から周辺単語を予測するのですが、正解・不正解を判定するには
出力をまとめて予測するよりも、1個づつやるほうが単純です。
周辺単語を1個づつばらして、中心単語から周辺単語の一つを予想する、という
学習設定にします。

CBOWはあるコンテキストからある単語を予測させ、SkipGramはある単語からコンテキストを予測させます。
そのようなタスクでトレーニングすることで、埋め込み表現を学習します。
いずれも、単語とコンテキストは密接な関係があるという分布仮説に基づいています。

ただ、ここまで、CBOWもSkipGramの出力層に、全結合が存在することに留意してください。
単語数サイズの全結合やSoftmaxは効率が悪いので、対策があります。
それは別の単語ベクトル導出の動画でご紹介します。

自然言語処理:
ニューラルネット、Sigmoid関数

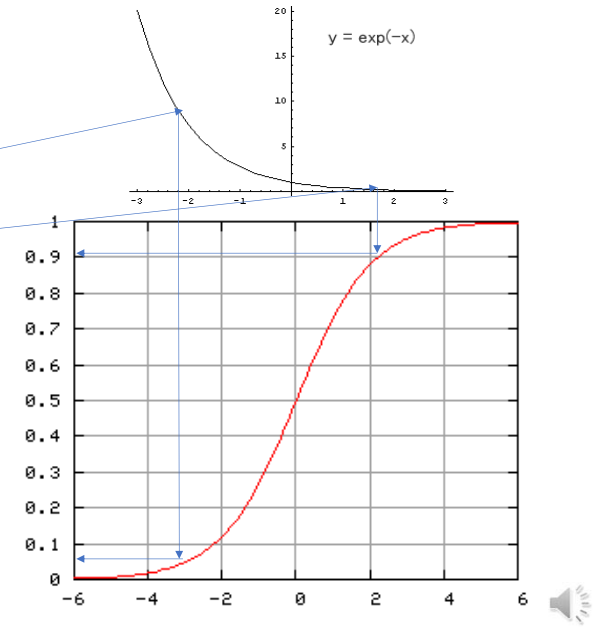
解説動画

ここでは、活性化関数や出力層で使われるSigmoid関数を見ていきます。

Sigmoidグラフ

- 入力xが負数だとEXP(-x)は大きくなり、Sigmoidは0に近くなる。
- 入力xが正数だと、EXP(-x)は小さくなり、Sigmoidは 1 に近くなる。
- つまりSigmoidは、任意の実数値を、0..1の範囲の値に変換する。

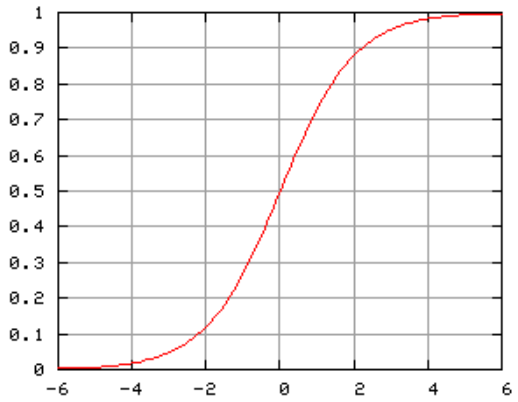
$$S = \frac{1}{1 + e^{-x}}$$



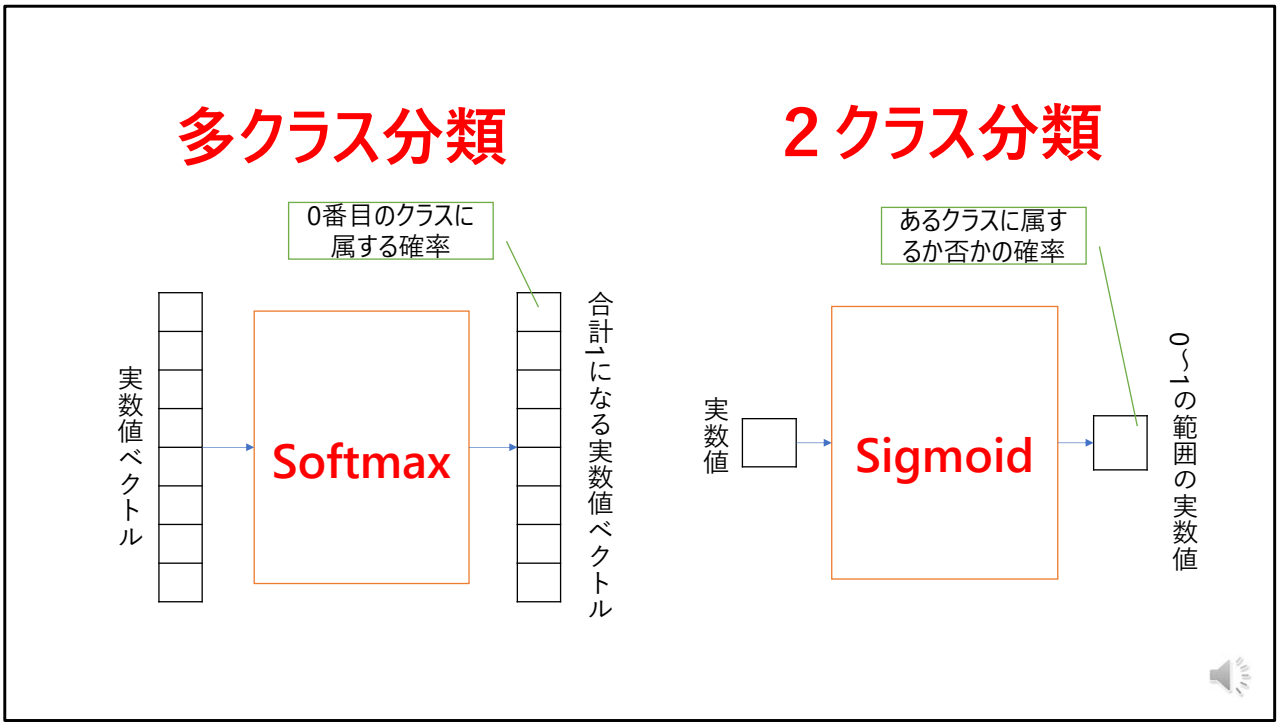
Sigmoid関数は、ここに示したような式で表されます。
オイラー数の-x乗のグラフと、それに対応したSigmoidのグラフを右に示します。
まず右上のグラフ：
入力 x が負数で、 x がマイナス方向に大きくなるほど、オイラー数の-x乗は正数でより大きくなります。
入力が正数だで、より大きくなるほど、オイラー数の-x乗は正数でより0に近づきます。
次に右下のグラフ：
それに対応して、Sigmoid関数の値は、 0 から 1 までの間を動きます。
x は一無限大から＋無限大の実数の範囲を動く間、Sigmodは0から 1 の値をとります。

Sigmoid
機能

0..1の範囲へ変換する
＝
YESである確率を得る。



このためSigmoidは、0..1の値にしたいときに利用されます。
0..1の値は、YES/NO判定でYESである確率などとして解釈できます。
また、画像を2つの領域にクラス分けする時などでも利用できます。
また、Sigmoidは、中間層で、何か他の値の重みづけなど、0..1の値を利用したいときに利用されます。

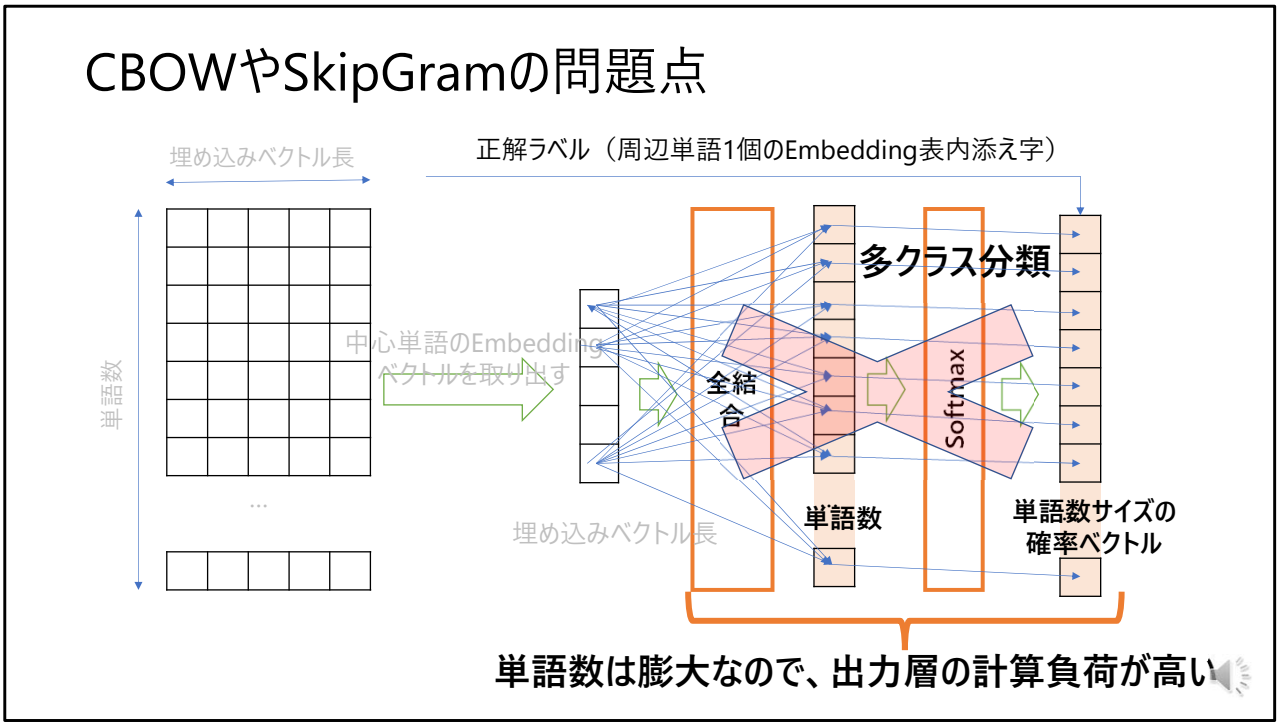


Softmaxは、足して1になる値へ変換するので、
他クラス分類の出力層に利用されます。
一方、Sigmoidは、YES/NO判定に利用できる性質から、
2値問題、2クラス分類問題の出力層に利用されます。

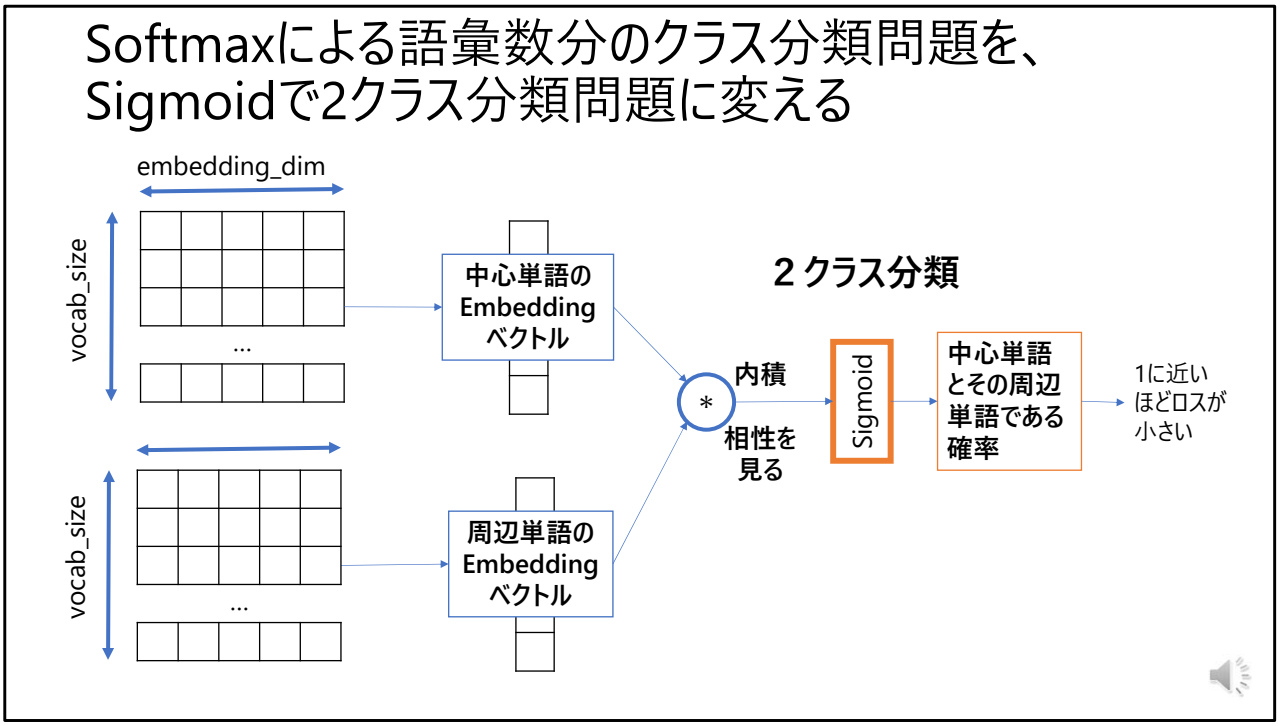
自然言語処理:
単語ベクトルの導出ー
Negative Sampling

解説動画

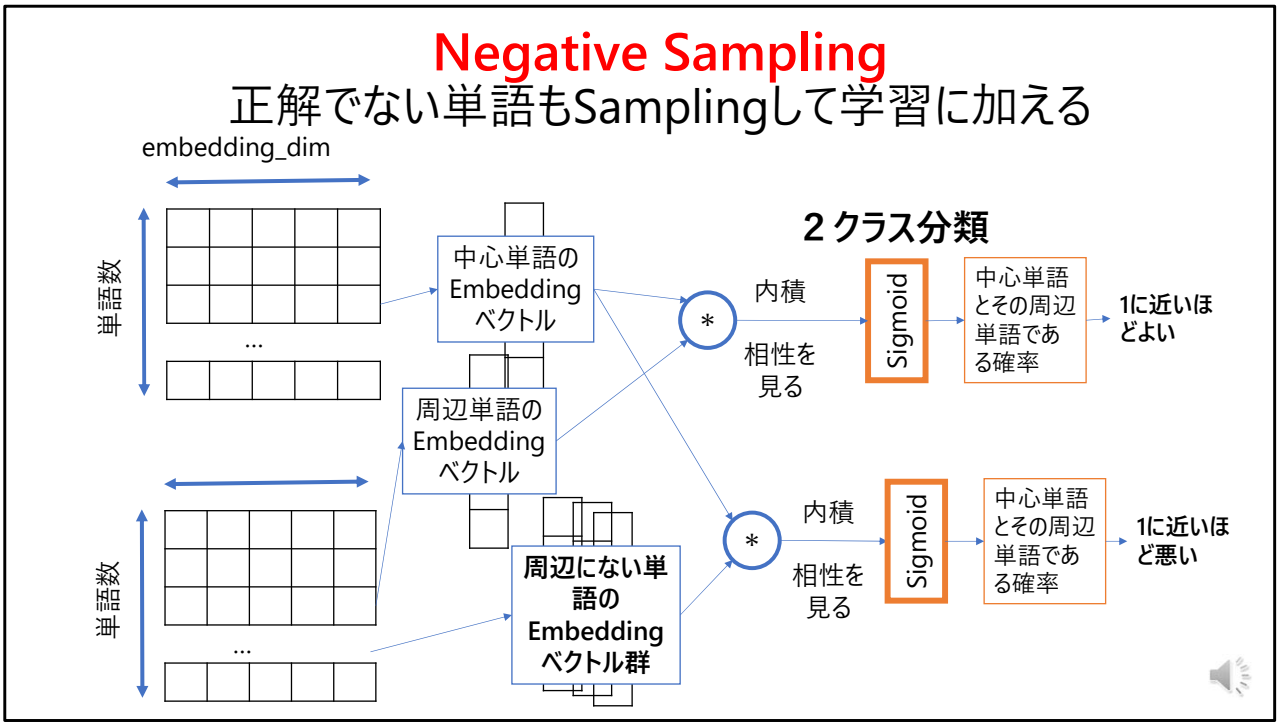
単語ベクトルの導出には、分布仮説に基づいたCBOWやSkipGramが利用されました。
ここでは、それらの出力層で用いられるテクニックであるNegative Samplingについて紹介します。



単語ベクトルの導出の前編でご紹介したCBOWやSkipGramは、出力層のところで、
埋め込みベクトルから全結合で単語のOne-hotベクトルに戻し、
さらにSoftmaxに通すという処理を行っていました。
ここで、単語数が数百万のオーダーだと、全結合の組み合わせは膨大になります。
また、Softmaxの計算にも負荷がかかります。
それを解決します。



そのポイントは、Softmaxによる語彙数分のクラス分類問題を、Sigmoidで2クラス分類問題に変える、
という点です。
埋め込み表から中心単語と周辺単語の埋め込みベクトルを取り出します。
それらベクトルの内積をとれば、ベクトル間の相性・近さがスカラーで量化できます。
より大きければ、相性が良い、近いということになります。
そのスカラーを、Sigmoidにとおし、YES/NO判定します。
中心単語と周辺単語の関係の単語の場合は、より相性が高ければ、Sigmoidを通せば1に近づきます。
この値をロスとして扱うために、Sigmoidの結果を1から引くなどのなどの加工をします。
すると、相性が大きければ、Sigmoidが1に近く、ロスが 0 に近くなります。
このような設定をしてトレーニングすれば、全結合なしで、中心単語と周辺単語の関係を
ネットワークおよび埋め込み表に学習できそうです。



他クラス分類を2クラス問題に転換したうえで、
中心単語と周辺単語ではない関係も、学習で仕込む必要があります。
そのためには、トレーニングデータとして、正のデータ、つまり中心単語と実際に周辺に合った単語、
および負のデータ、つまり、中心単語とその周辺にはきそうもない単語、双方を与えます。
負の単語は、周辺以外からランダムに高頻度単語をピックアップするなどして選びます。
ロスの計算時には、正のデータと負のデータの扱いを少し変えて、
正のデータならロスが小さく、負のデータならロスを大きくなるように設定します。
このようなテクニックをNegative Samplingといいます。

gensimパッケージ

gensim パッケージ

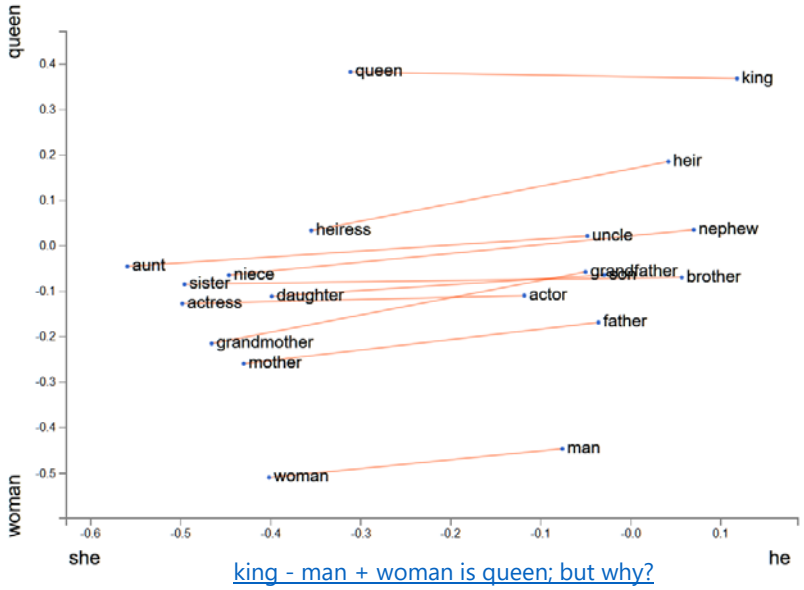
- 単語から文書まで、ベクトル化して、モデル作成できる Python Package。
  - ホームページ　https://radimrehurek.com/gensim/
  - 単語ベクトルを作ってくれる　https://techacademy.jp/magazine/30591
  - 100本ノックの課題では、出来合いの単語ベクトルをロードして、いろいろ操作するメソッドを利用します。

単語ベクトルの効果

単語ベクトルの性質

- 単語埋め込み表現は、ある単語を、周辺単語の何らかの特徴群を抽出することで、表現する。それら特徴は、その単語の形態的、構文的、意味的な属性をいろいろ含む。
- 100本ノックの課題の中に、単語ベクトルの足し引きをして、単語ベクトルが単語の意味を表現していることを体験するものがあります。Enjoy。

king - man + woman = queen



king - man + woman is queen; but why?

参考資料

- オリジナルペーパ:　"Efficient Estimation of Word Representations in Vector Space"
- 解説動画
  - Deep Learningで行う自然言語処理入門
  - 【レクチャー: word2vecの概要】自然言語処理とチャットボット: AIによる文章生成と会話エンジン開発
- king - man + woman is queen; but why?
- Python 実装
  - word2vec(Skip-Gram with Negative Sampling)の理論と実装2
  - https://github.com/theeluwin/pytorch-sgns/blob/master/model.py
- genism の Word2Vec パッケージ
  - gensimのWord2Vecを使ってみる。
  - gensimを使ってWikipediaの全日本語記事からWord2Vecを作る

課題に即した補足

スピアマン相関係数

- 順位の相関を求める

$$\rho = 1 - \frac{6\sum D^2}{N^3 - N}$$

で定義される。ただし

D = 対応するXとYの値の順位の差

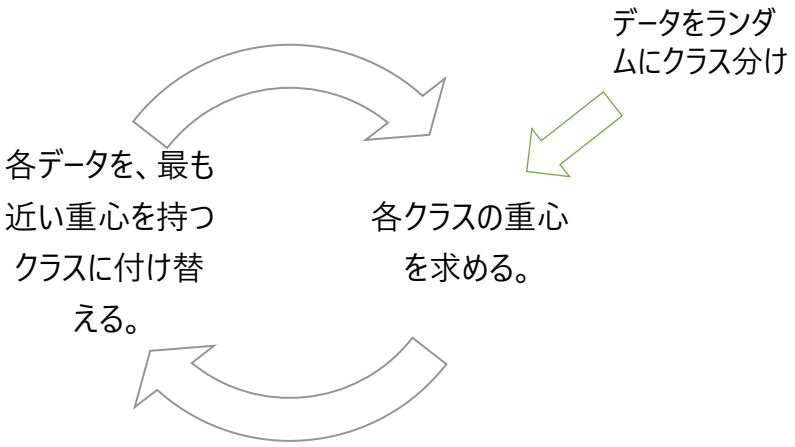
N = 値のペアの数

自然言語処理:
機械学習、クラスタリング

解説動画

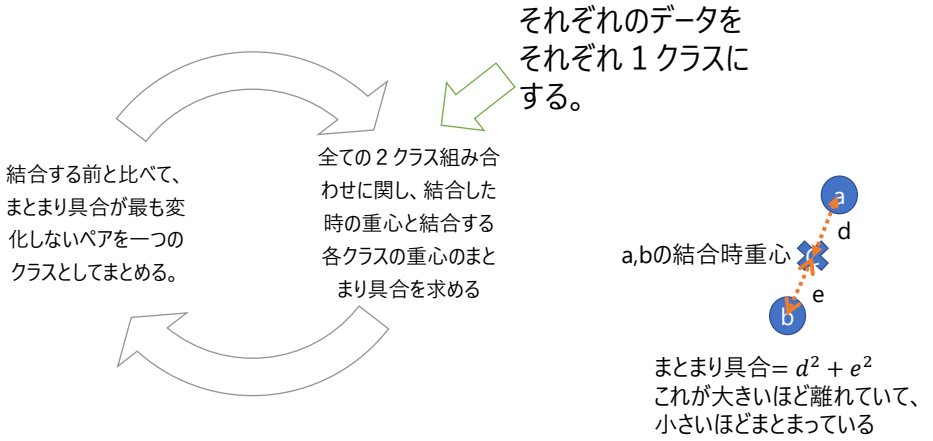
クラスタリングというのは、機械学習の1分野で、教師なし学習によって、
データポイントをいくつかのクラスに分類するタスクです。
データポイントが、何らかの数値あるいはベクトルで表現できているとき、
利用できるクラスタリングの代表として、K-MeansとWard法を紹介します。
いずれも近いものをグルーピングすることを繰り返し法で求めて行きます。

k-means クラスタリング



K-meansは、最初、すべてのデータポイントをランダムにクラス分けします。
次に、各クラスの重心を求めることと、すべてのデータポイントに関して
それぞれを最も近い重心を持つクラスに付け替えること、
を繰り返します。
その繰り返しによって、データポイントは、所定のクラス数にグルーピング
されます。

ward クラスタリング



https://www.albert2005.co.jp/knowledge/data_mining/cluster/hierarchical_clustering
https://mathwords.net/wardmethod

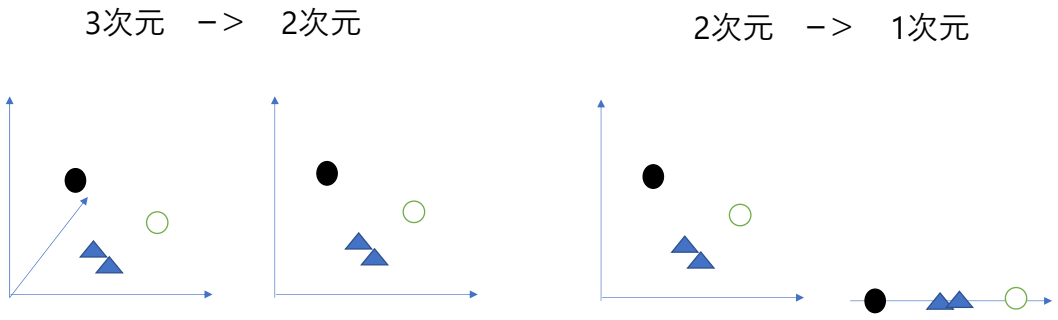
ward法は、最初、データポイントをそれぞれ1クラスと初期化します。
そして、すべての二つのクラスの組み合わせの重心を求め、そこから二つのクラスの各クラスの重心への距離の平方和を求めます。この距離の平方和というのは、統計でいう分散、いわば、ばらけ具合です。
ばらけ具合が小さいことが望ましいので、まとまり具合といったほうがよいでしょう。
次に、まとまり具合が最も変化しないペアを1クラスへと結合します。
まとまり具合の変化とは、組み合わせた後の距離の平方和から、組み合わせる前の平方和を引いたものです。
これを繰り返します。
ward法は、クラスターが階層的に得られることが特徴です。
k-Meansと比べて、あらかじめ分割するクラス数を決めなくても、階層の適当な箇所に注目することで、任意のクラス数のクラスターを得られるという違いがあります。

自然言語処理:
機械学習、t-SNE

解説動画

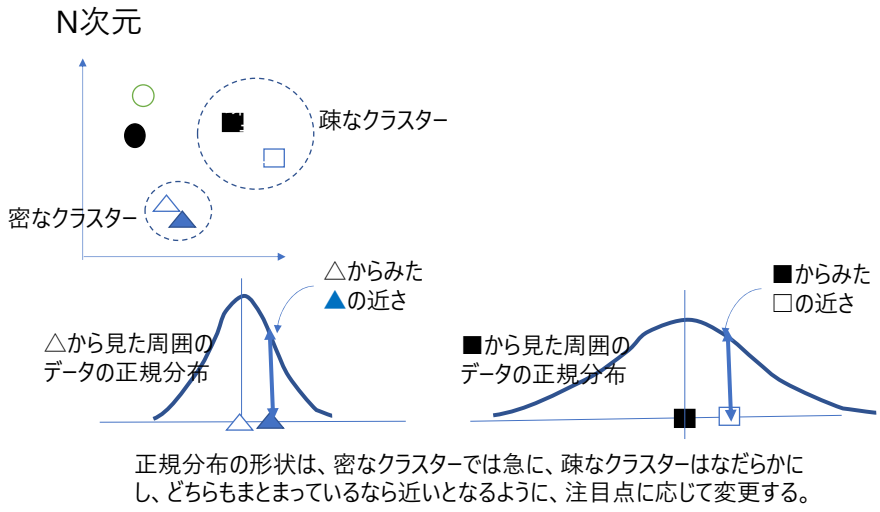
ここでは、次元圧縮の手法であるt-SNEについて解説します。

t-SNE (t-Distributed Stochastic Neighbor Embedding)：次元圧縮



t-SNEというのは、t分布で確率的に近さを表現する、とでもいうような意味です。
次元圧縮というのは、多次元データをより少ない次元に圧縮することです。
t-SNEは、特に、多次元データを人の見やすい2次元で表現するのに使われます。

t-SNE：近さの正規化

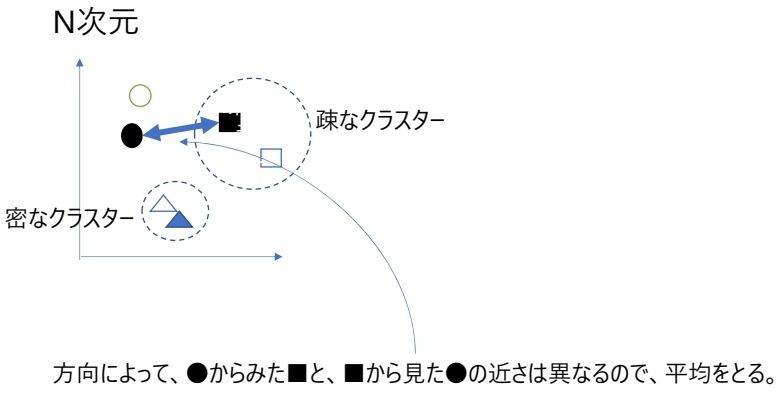


最初に多次元空間でのデータポイント間の近さの定義を見ていきます。

t-SNEは、データポイント間の近さを表現するときに、
あるデータポイントからほかのデータポイントを見た時の近さというものを考えます。
注目データポイントを中心とした正規分布グラフを利用して、別のデータポイントの位置するところのグラフの高さをもって、
注目データポイントからその別のデータポイントへの近さとします。
このとき、密なクラスターと疎なクラスターとで、どっちでも寄り集まり具合がはっきり見えるように、近さを正規化します。

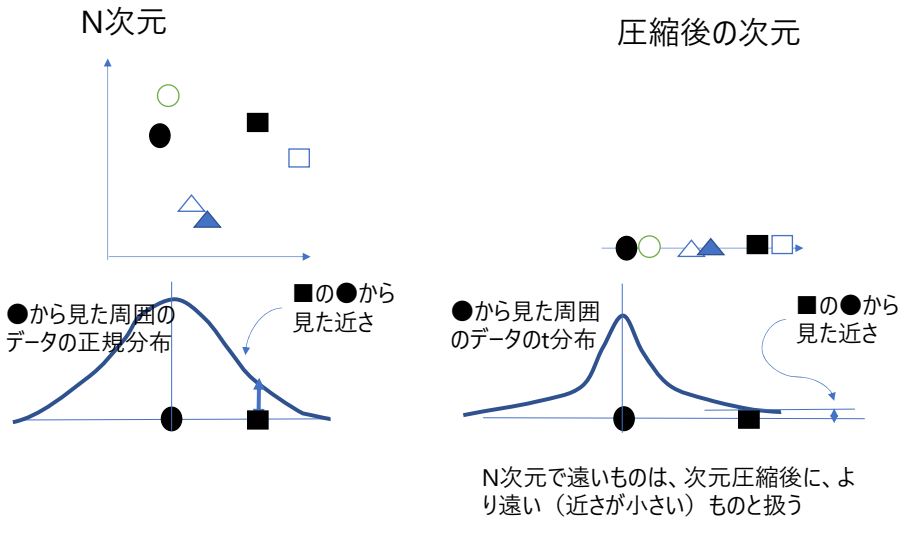
どうやるか。疎なクラスターか密なクラスターかで、利用する正規分布を変えます。
密なクラスターの場合、幅の狭い正規分布を用い、疎なクラスターの場合、幅の広い正規分布を使います。
そうすることで、密なクラスターでも疎なクラスターでも、寄り集まっているデータポイントは同じくらい近いとみなせるようにします。そうすることで、寄り集まっているデータポイントを、疎も密でも、寄り集まっているものと認識させます。

t-SNE：平均をとる



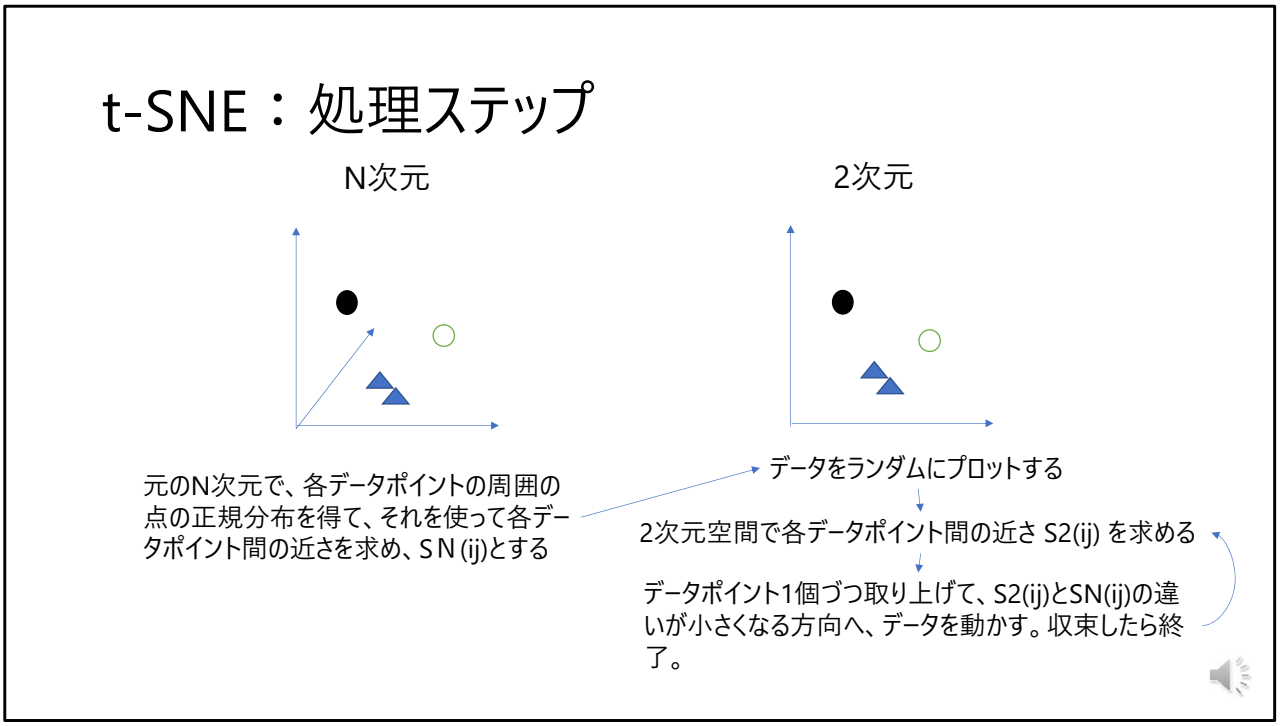
ところで、注目データポイントとから別のデータポイントの近さは、立場を変えた時の近さと異なります。
そこで、二つのデータポイントの近さは、注目点を交代したものとの平均で定義します。

t-SNE：t-分布の役割



次に、圧縮後の距離の定義を見ていきます。
圧縮前のデータポイント間定義は、注目点を中心とした正規分布の高さで定義しました。
一方、圧縮後は、正規分布の代わりに、より山が急ですそ野が広いt-分布のグラフを使います。
すると、ある程度離れたデータポイントならば、急速に近さが小さくなる、つまりより遠いものとなります。
そうすることで、ある程度離れたものは十分離れているように、次元圧縮します。

以上、圧縮前に、寄り集まっているものは寄り集まっていることを強調するように距離を定義し、
次元圧縮時には、離れたものは十分離れているように距離を定義します。



以上のテクニックを踏まえて、t-SNEのアルゴリズムは、次のように進みます。
最初に、圧縮前の多次元で、各データポイントの周囲の点の正規分布を選び、それを使って各データポイント間の近さを求めます。
次に、圧縮後のプロットを求めます。
それには、まずデータポイントをランダムにプロットし、データポイント間の距離を求めます。
そして、すべてのデータポイントについて、それぞれ1個づつ取り上げて、圧縮前の距離と圧縮後の距離とが近くなるように、
注目データポイントを移動させます。
移動させたら、再びデータポイント間の距離を求め、データポイントの移動と距離計算を繰り返します。
収束したら、終了です。

t-SNE 参考資料

- 英語だがわかりやすい
- PCAとの対照データが印象的

100本ノック第７章課題：単語ベクトルの利用60～63、機械学習66～69

- この章の課題は、単語ベクトルがすでにあるものとして、使ってみるものです。前半の60～63で、単語ベクトルがどういうものか、その効果が興味深く示されます。後半の66～69は、単語ベクトルを利用してできること、面白いネタが並んでいます。「NLP、単語ベクトル.ipynb」というノートをコピーし、サンプルコードを完成させ、実行してください。ノートには解説やコメントを入れています。

オプション課題：CBOW, SkipGram, NegativeSampling コード読解

- cbow_skipgram_negativesampling.ipynb に、cbow, skipgram, negative sampling で、単語ベクトルを学習するコードのひな型を入れてあります。学習のため、ミニバッチを使わず、おまけ処理を極力省いています。予備知識で説明したことのコード版です。

オプション課題：gensimで単語ベクトルを作るコード読解

- 自然言語処理アプリを組むときは結局、単語ベクトルを自作する必要があります。自然言語アプリを組む人は、まずgensimを利用すると思います。
- gensim_word2vec.ipynbに、小規模ながら、gensimパッケージで、単語ベクトルを作成するコードを入れました。読解し、実行してください。
- 参考：データが大きく時間がかかりすぎますが、wikipedia_corpus.ipynbにもgensimを使った単語ベクトル作成コードがあります。

確認クイズ

- 確認クイズをやってください。