自然言語処理
―機械学習―

https://yo-sato.com/

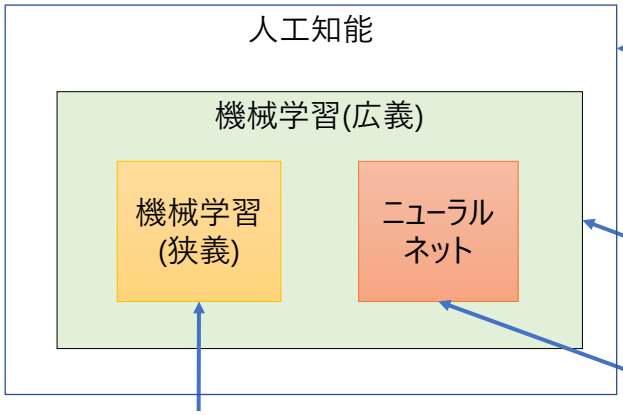
機械学習（100本ノック第5章）の位置づけ

- 狭義の機械学習よりもニューラルネットが、より広範囲の問題に適用でき、性能もうわまっているようなので、手法のいろいろに潜るよりも、ニューラルネットと共通（広義の機械学習）の考え方の部分を学習します。

- 100本ノックの課題集の第5章では、具体的な手法例としてロジスチック回帰を取り上げ、それを使って、トレーニングと評価データを分けること、評価指標としての混同行列、正則化を取り上げています。これらの概念は、ニューラルネットでも共通です。なお、第6章の単語ベクトルの課題集の中で、機械学習ネタとして、クラスタリング手法 2 つと次元圧縮手法一つが取り上げられています。

- 機械学習は、個々のテクは、パッケージですでにサポートされていることが多いので、Pythonレベルでは、Logicを組むというよりも、メソッドを選び引数を塩梅するだけです。そこで、どういう手法（メソッド）が、どういうケースに利用できるのか、を具体的な適用例を通して理解することが重要です。

人工知能、機械学習、ニューラルネット



- 人の知能を模倣する。
- 評価手法

- データでトレーニングして予測する
- データから特徴を抽出して予測に使う
- ロス関数を使って最適化する。そのときに過学習を抑える正則化を行う。

- 推論根拠となる入力特徴を人が定義する
- 推論根拠がわかりやすい

- 推論根拠となる入力特徴を機械が学習する（複雑な対応関係でもOK）
- 推論根拠がわかにくい

Package使い

- Pythonの基本的なところをやっているときは、ロジックを読む・書くという作業でした。
- ところで、Pythonは、Packageが豊富にそろっているという特徴があります。やりたいことがあったとき、クラスやメソッドとしてアルゴリズムは、すでに、たいてい、用意されているのです。
- そういう場合、コーディング上は、メソッド呼び出し一行です。
- そこで、その一行がだいたい何をやっているかのイメージを持っていないと、サンプルコードが読めません。

概念的な習得が大事

- そのため、このあたりから、ロジックというより、知識ないし概念的な理解が重要になります。
- 以下のようなことを把握していることが必要になります。
  - どのクラスのどのメソッドを、どういうときに使うか
  - メソッドに渡し受け取る引数のデータはどういう意味、構造をもつか
- 概念的な理解は、最初は難しいです。忍耐強く取り組んでください。
  - 1回目は、チンプンカンプン。
  - 2回目は、わかることがでてくる。
  - 3回目は、わかることが増えて、何がわからないかがわかる。-> ここまでくれば後は早いです。

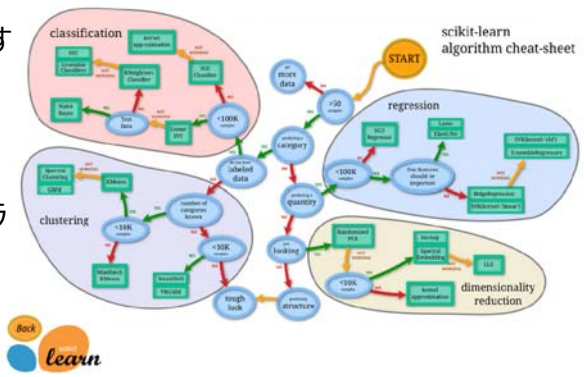
自分で書くとき

- アルゴリズムが1行で済む場合、プログラムを自分で書き起こすときは、前後の引数のデータ加工の部分で、ロジックを書く時間が必要となります。
- そこで、リスト処理、Numpy、Pandasなどが活躍します。

Python Package
scikit-learn（サイキット・ラーン）

分類（クラス分け）：データを複数のクラス（グループ）に分類すること

自動グルーピング（クラスタリング）：自動的に分類すること



回帰分析：連続尺度の従属変数（目的変数）と独立変数（説明変数）の間で予測モデルを作ること

次元圧縮（Dimensionality Reduction：次元を下げて圧縮すること。

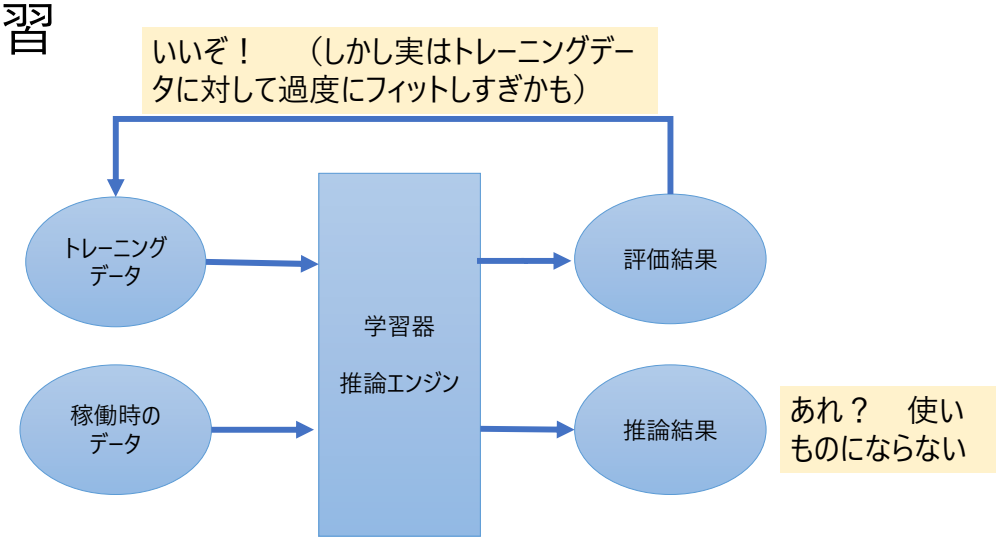
https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html#ml-map

以下を眺めてください。Google Chromeの動的な翻訳で、なかなかいい日本語で読めます。

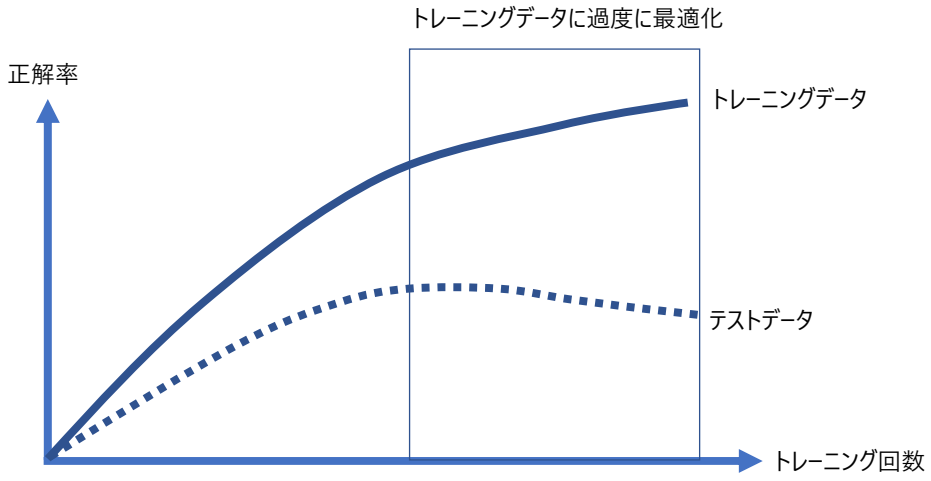
https://scikit-learn.org/stable/
https://scikit-learn.org/stable/modules/classes.html#

トレーニングするときのデータの使い分け：
train, valid, test の役割

過学習

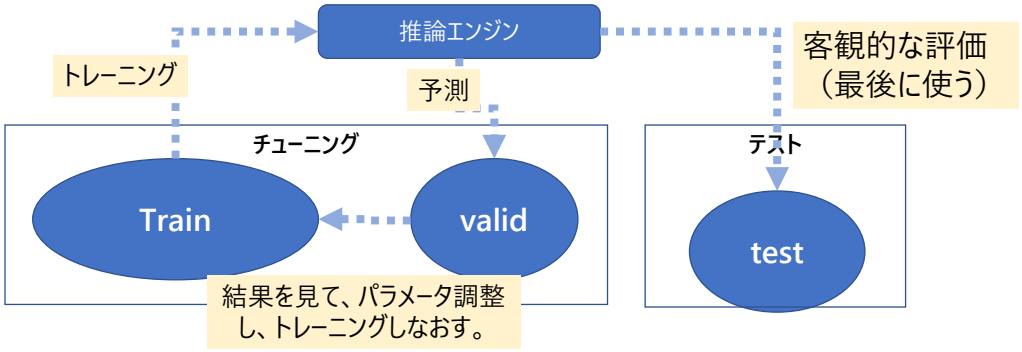

いいぞ！　（しかし実はトレーニングデータに対して過度にフィットしすぎかも）
トレーニング
データ
稼働時の
データ
学習器
推論エンジン
評価結果
推論結果
あれ？　使いものにならない

過学習が起きている



train, valid, test の役割

- Trainデータの結果だけを見ていると、**過学習**しているかどうかわからないので、trainとは別のデータvalidの結果を使ってチューニングする。
- チューニングに使ったデータ(train/valid)で評価しても客観的にならないため、最終的にはチューニングと無関係のtestデータで客観的な評価をする。



文、文書の素朴な表現

頻度ベクトル

I like a dog, and she likes a dog.

世の中に6個の単語しかないとする。
各単語をベクトルのある位置に対応させ、
単語の出現回数をその位置の値とする。

i	like(s)	a	dog	and	she
1	2	2	2	1	1

TF-IDF (Term Frequency, Inverse Document Frequency)

- 頻度だと重要でない高頻度語（「である」など）がノイズになるので、ある単語がある文書に登場したことの重要度を測る

TF・IDF
＝ 文書内出現度・文書に登場する珍しさ

$$= \frac{\text{文書内単語出現頻度}}{\text{文書内全単語出現数}} \cdot \log\left(\frac{\text{総文書数}}{\text{単語が出現する文書数}}\right)$$

全文書に登場するとlog(1)=0

https://mieruca-ai.com/ai/tf-idf_okapi-bm25/

TF-IDFベクトル

I like a dog, and she likes a dog.

i	like(s)	a	dog	And	she
0.4	0. 6	0.1	0.9	0.2	0.4

TF-IDFベクトルは、頻度ベクトルの頻度の代わりにTF-IDF（重要度）をいれたもの

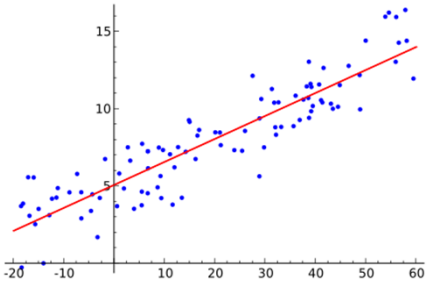
ロジスティック回帰

ロジスティック解説動画

機械学習の1例として、ロジスティック回帰を理解します。

目的変数と説明変数

目的変数 y ：…ために、結果がこうなった
従属変数 y ：結果は原因に従属して決まる



モデル y =ax+b

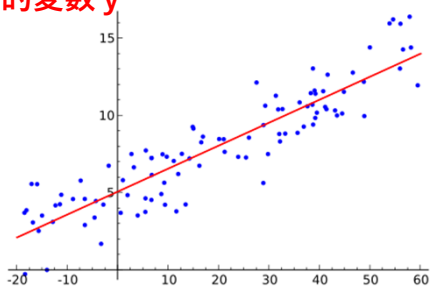
説明変数 x ： x がこれであるために…
独立変数 x ：原因 x は任意の値をとりうる

ロジスティック回帰を理解するために、まず、説明変数ないし独立変数と、目的変数ないし従属変数の違いを押さえます。
y=ax+bを考えます。 x を与えると、 y が得られます。 y ＝a x ＋ b は、 x と y の関係を示すモデルといいます。
ここで、 x がこれであるためにその結果 y がこうなった、という関係なので、 x は説明変数で、 y は目的変数と言われます。
これを別に表現すると、原因 x は任意の値をとりうるため独立変数、結果 y は原因 x に従属して決まるので従属変数、といいます。

線形回帰モデル

モデル y =ax+b　(線形式)

目的変数 y



説明変数 x

モデルパラメータa,bを適切に選ぶことで、目的変数 y が説明変数 x から、よく予測できる。

https://ja.wikipedia.org/wiki/%E5%9B%9E%E5%B8%B0%E5%88%86%E6%9E%90

説明変数 x と目的変数 y の間の関係を、y=ax+bという式でモデル化する場合、このax+bを線形式といいます。線形というのは直線で表現できる関係です。
また、予測されるべき値（目的変数 y が予測するもの）は、実数で、実数を予測するので、
回帰モデルと言われます。
併せて、y=ax+bは、線形回帰モデルと呼ばれるものの一種です。

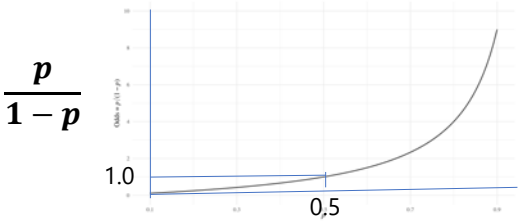
ロジット関数

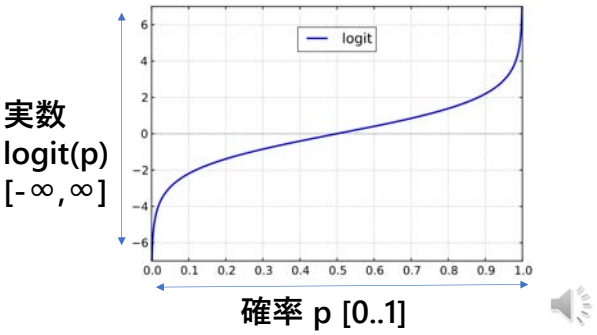
- ロジット：オッズ p/(1-p) の対数。

$$\mathrm{logit}(p) = ln\left(\frac{\mathrm{p}}{1-\mathrm{p}}\right)$$

lnはeを低とする対数

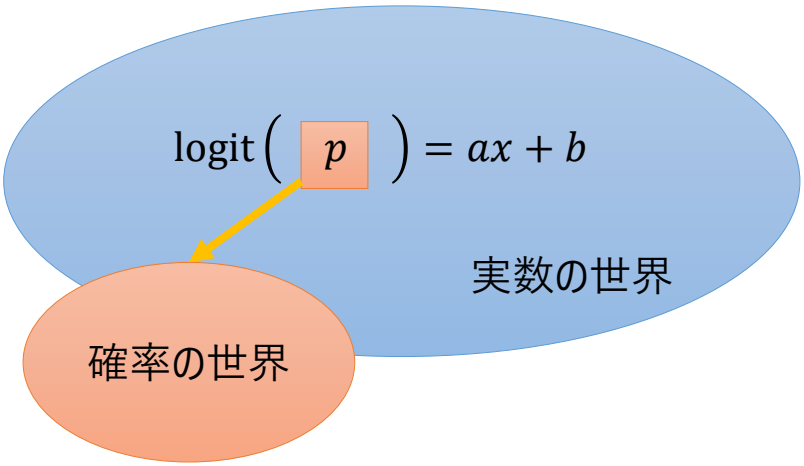
- 実数logit(p)と確率pを対応付ける。





ここで、確率の世界で、賭け事の良さを表すオッズというものがあります。
かつ確立を p とします。p/(1-p)は、負ける確率に対する勝つ確率の比です。
１より大きければ勝つ確率が勝り、１より小さければ負ける確率が勝ります。
このオッズは、右上のグラフのように、確率が0..1の範囲で、オッズが0..無限大の範囲を動き、
確率が0.5の時に1となります。
いま、オッズの対数を取ります。それを確率pのlogitと言います。
logitは、-無限大～無限大の範囲、つまり、実数の範囲の値をとります。
ここで、0..1の確率と、-無限大..無限大の実数が、関連付けられていることに留意してください。

連結関数：異なる世界をつなぐ

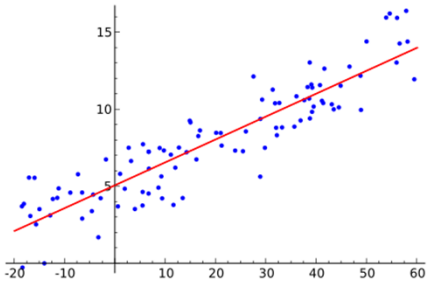


https://stats.biopapyrus.jp/glm/def.html

ロジスティック回帰を理解するために、連結関数という考え方を見ます。
logit関数は、確率と実数を関連付けるものでした。
そのように、確率と実数という異なる世界を関連付ける関数を、連結関数といいます。

予測における確率の効用

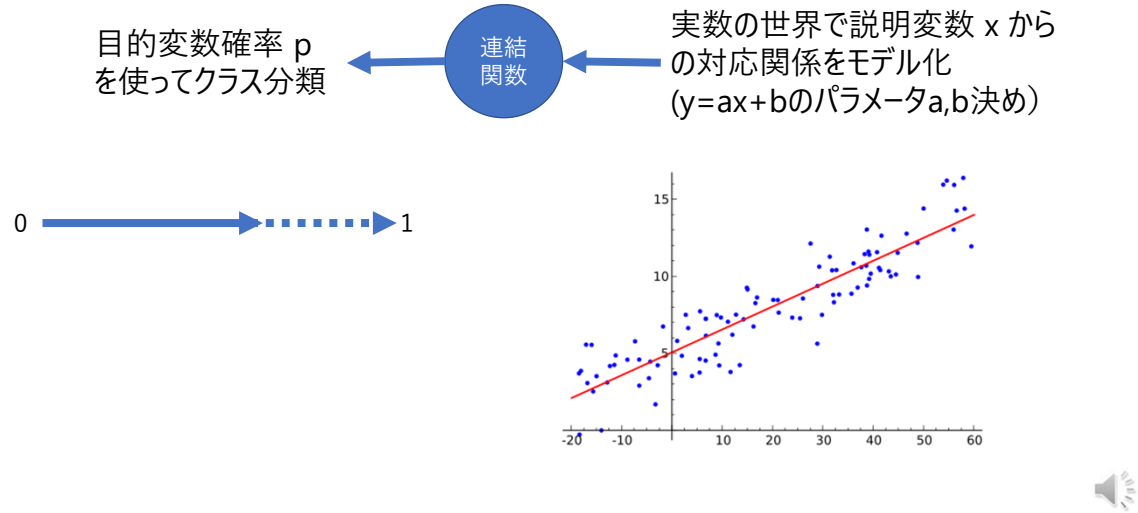
確率の世界でできることは、
YESかNOかでYESである確率が0.6、
写真のオブジェクトが犬である確率が0.6、
などクラス分類問題。



実数の世界でできることは、
ある値の予測や再現（回帰）

実数値を予測する回帰問題には、無数の応用があります。
人口予測、ウィルス拡散予測、販売予測、過去データからのセンサー値の予測、などなど。
ここで、なぜ確率の話が出てくるのでしょうか？
確率値を予測すると何ができるかというと、クラス分類問題に利用できるからです。
白（１）か黒（０）かの二値問題を考えてみます。
ある説明変数 x のとき、目的変数の値が１になったとします。これは x のときは白ということです。
ある説明変数 x のとき、目的変数が0.7になったとします。これは、 x は0.7くらいの確率で白ということです。
このように、確率値の予測は、クラス分類問題に利用できます。
複数のクラスへの分類問題では、クラス１へ属する確率がなんぼ、クラス2へ属する確率がなんぼ、と風に利用します。

連結関数



連結関数の役割は、例えば、実数という世界と、確率という世界を関連付けることでした。
先の確率の効用で説明したことを踏まえると、連結関数の役割がより具体的に説明できます。
実数の世界で予測モデルを作ります。実数の世界の回帰手法はいろいろあります。
ここで、連結関数を使うことで、確率の世界へ翻訳できます。
確率の世界では、クラス分類に利用できます。
つまり、実数の世界のモデル化手法がすべて、連結関数で、クラス分類に応用できるということです。

ロジスティック回帰

連結関数としてロジットを使用する一般化線形モデル (GLM) の一種

左辺：目的変数pにロジットという皮をかぶせ（右辺のモデルと左辺を関数で連結させ）ることで、実数を確率値pと関連づける。->pを利用することで2値問題や、多クラス分類に利用できる。

$$\mathrm{logit}(p) = ln\left(\frac{p}{1-p}\right) = ax + b$$

線形モデル

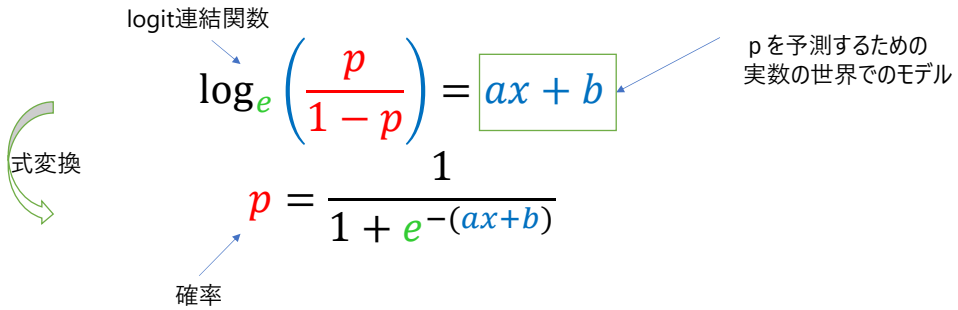
右辺：線形モデルの値をそのまま目的変数にすれば、-∞から+∞。->実数値を予測するエンジン（線形回帰）としてしか使えない。

https://ja.wikipedia.org/wiki/%E3%83%AD%E3%82%B8%E3%82%B9%E3%83%86%E3%82%A3%E3%83%83%E3%82%AF%E5%9B%9E%E5%B8%B0

ロジスティック回帰は、確率 p と線形モデルとをLogit関数で関連付けます。その結果、実数を予測する線形モデルでありながら、クラス分類に応用できる手法となっています。

対数と指数の関係を使って目的変数 p の式を得る

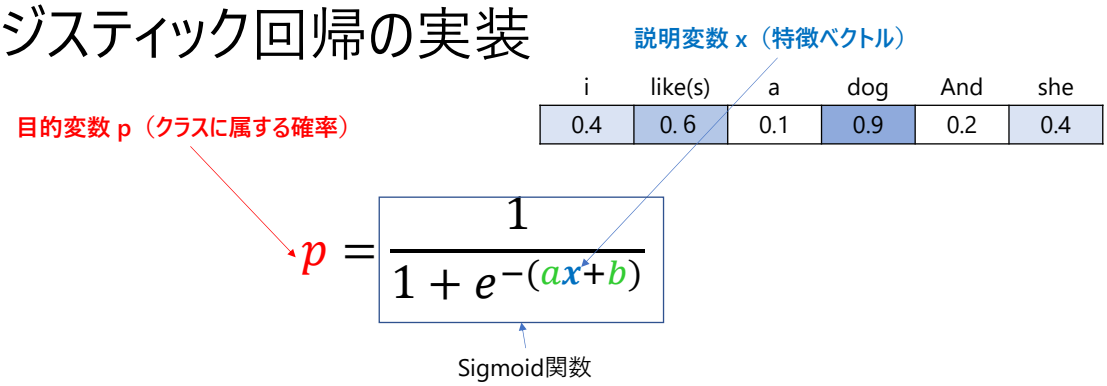
$$e^y = L \leftrightarrow \ y = \log_e L$$



説明変数xがある値をとったときに、目的変数p（あるクラスに属している確率）が得られる。

では、ロジスティック回帰を、コードで実装する際の方針を見ていきます。
前のページのロジスティック回帰の式を、指数と対数の関係で変換し、
p を左辺とするように変形します。
ここで、説明変数は x 、目的変数は p で、 x を与えたとき、あるクラスに属する確率 p を
求めます。
その p を予測するためのモデルは、ax+bです。モデルのパラメーターはaとbです。

ロジスティック回帰の実装



説明変数x（特徴ベクトル）がある値をとったときに、目的変数p（あるクラスに属している確率）が得られる。

ある特徴ベクトルのときの正解クラス分類を教師データとして、pが1になるように、パラメータ(バイアスb、重みa)をトレーニングする。

機械学習で、モデルを学習します。
モデルはax+bで、そのパラメータは、a、 b です。
入力データ x と、教師データ y があるとします。yは、 1 か0かとします。
入力データ x が与えられたときに、結果の y により近くなるように、a,bを動かします。
それを、何百万という訓練データを使って、うまくa,bが収束して最適な値になるように
します。その手法は、またいろいろありますが、基本的にはa,bを最適化して、x 、 y のペアが最もうまくあてはまるようにするためのアルゴリズムを使います。

ロジスティック回帰

- **連結関数としてロジットを使用する一般化線形モデル (GLM) の一種。**

説明変数の特徴ベクトル

$$\mathrm{logit}(p_i) = ln\left(\frac{p_i}{1-p_i}\right) = \alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \ldots \beta_k x_{k,1}$$

モデルのパラメータ（トレーニング対象）

式変換

$$p_i = \frac{1}{1 + e^{-(\alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \ldots \beta_k x_{k,1})}}$$

https://ja.wikipedia.org/wiki/%E3%83%AD%E3%82%B8%E3%82%B9%E3%83%86%E3%82%A3%E3%83%83%E3%82%AF%E5%9B%9E%E5%B8%B0

これまで、y=ax+bという単純な式を使いましたが、一般には、
説明変数 x は複数あり、ベクトルで表現されます。
また、目的変数 y も、多クラス分類の場合は、複数の値、ベクトルとなります。

ロジスティック回帰：連結関数、一般化線形モデル

$$\underbrace{\mathrm{logit}(\mathrm{p_i}) = ln\left(\frac{p_i}{1-p_i}\right)}_{\text{右辺は実数値。目的変数pにロジットという皮をかぶせ（右辺のモデルと左辺を関数で連結させて）ることで、確率値と関連づける。->実数値でなく、pを利用することで2値問題や、多クラス分類に利用できる。}} = \underbrace{\alpha + \beta_1 \mathrm{x_{1,i}} + \beta_2 \mathrm{x_{2,i}} + \ldots \beta_k \mathrm{x_{k,i}}}_{\text{線形モデルの値をそのまま目的変数にすれば、-∞から+∞。->実数値を予測するエンジン（これが線形回帰）としてしか使えない。}}$$

https://stats.biopapyrus.jp/glm/def.html

ロジスティック回帰：実装

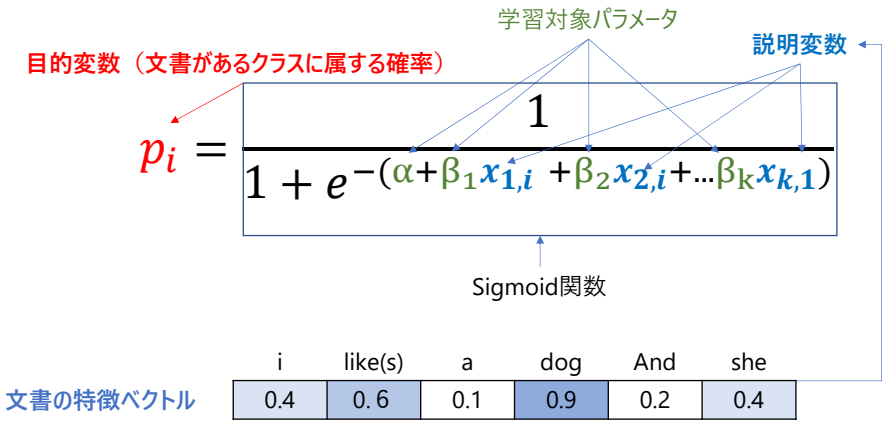
目的変数（クラスに属する確率）　　説明変数（特徴ベクトル）

$$p_i = \frac{1}{1 + e^{-(\alpha+\beta_1 x_{1,i} +\beta_2 x_{2,i}+\ldots\beta_k x_{k,1})}}$$

Sigmoid関数

説明変数Xi（特徴ベクトル）がある値をとったときに、目的変数（あるクラスに属している確率）が得られる。
ある特徴ベクトルであるときに、正解クラス分類を教師データとして
piが1になるように、パラメータ(バイアスα、重みβ)をトレーニングする。

ロジスティック回帰によるクラス分類



評価指標：混同行列（Confusion Matrix）

ConfusionMatrix解説動画

機械学習の評価に用いられる混同行列について解説します。

評価指標：Confusion Matrix

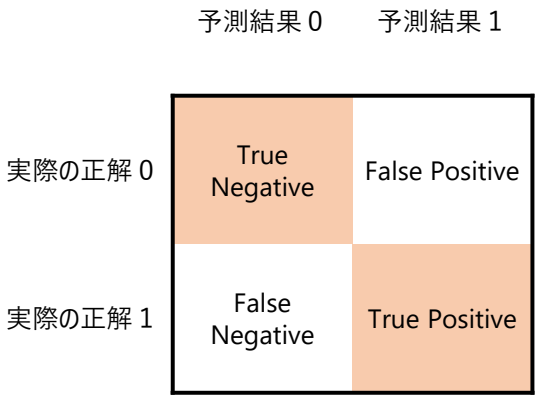
	予測結果0 (陰)	予測結果１ (陽)
実際の正解０（陰）	**True Negative** 正しく、陰とした。(真陰性)	**False Positive** 間違えて、陽とした。(偽陽性)
実際の正解１（陽）	**False Negative** 間違えて、陰とした。(偽陰性)	**True Positive** 正しく、陽とした。(真陽性)

機械学習の評価には、いくつかよく使われる尺度があります。
それらは、このConfusion Matrixに基づいています。
混同行列とは、正解と予測結果の行列です。
単純にするため、正解、予測結果とも、０，１の二値で、１を陽性、０を陰性とします。
行列の左上のマスを見てください。
実際は０、陰性のときに、それを０，陰性と予測しました。
この場合を、True Negativeといいます。True Negativeとは、正しく陰とあてたという意味です。
行列の右上のマスを見てください。
実際は０、誤りのときに、それを１，陽性と誤って予測し、間違えました。
この場合を、False　Positiveといいます。False Positiveとは、間違えて陽性とした意味です。
コロナ検査でよく聞く偽陽性というのはここに当たります。
行列の左下のマスを見てください。
実際は１、陽のときに、それを０，陰と予測しました。
この場合を、False Negativeといいます。False Negativeとは、間違えて陰としたという意味です。
行列の右下のマスを見てください。

実際は１、陽のときに、それを１，陽と当てました。
この場合を、True　Positiveといいます。True Positiveとは正しく、陽としたという意味です。
以下、このConfusion Matrixを使って、具体的な評価指標を見ていきます。

評価指標：正解率(Accuracy)

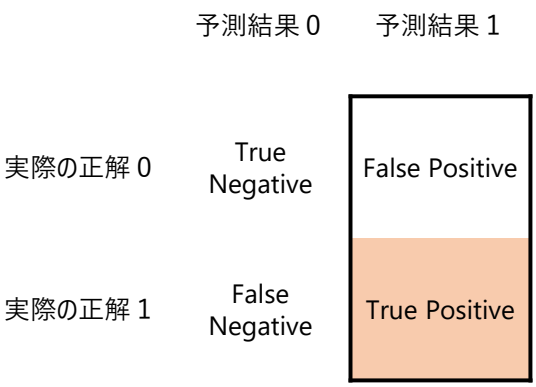
- 正解率
  - 分類したデータの総数のうち、正しく分類されたデータ数の割合
  - (TP+TN)/(TP+FN+FP+TN)
    - 右図、太枠が分母で、色付きセルが分子
    - ０，１のいずれをTrue、Falseとするかによって、値が異なることに留意。



正解率、英語でAccuracyを定義します。
Accuracyとは、右の図に、太枠を分母とし、色で染めたところを分子とする、評価です。
要するに、当たった数をテスト総数で割ったものです。
通常は、Accuracyを使うことが多いです。

評価指標：適合率(Precision)

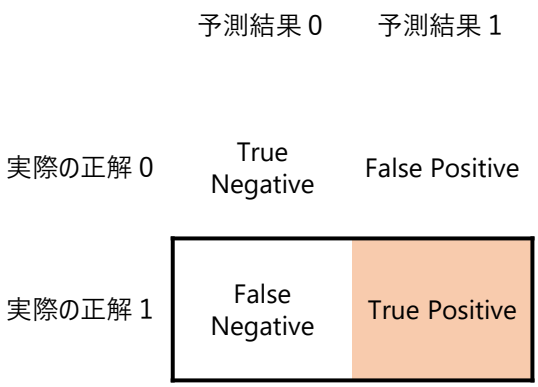
- 適合率
  - クラス1に分類されたデータのうち、実際にクラス1であるデータ数の割合
  - TP/(TP+FP)



適合率、英語でPrecisionを定義します。
Precisionは、右の図太枠を分母とし、色で染めた部分を分子とする、評価です。
テストで正解と予測したもののうち、実は確かに正解なものがどれだけあるか、なので、
要するに、正解という予測がどれだけ信用できるか、を評価します。

評価指標：再現率(Recall)

- 再現率
  - 実際にクラス1であるデータのうち、クラス1に分類されたデータ数の割合
  - TP/(TP+FN)

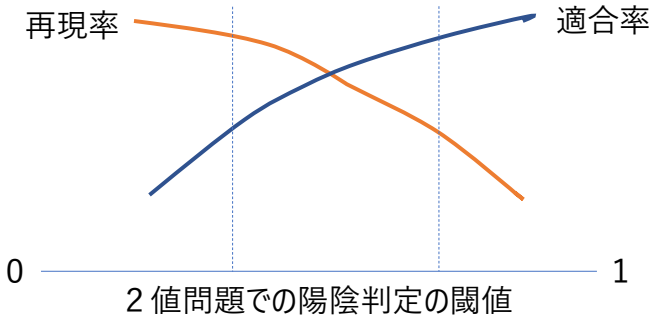


再現率、英語でRecallRateを定義します。
RecallRateは、右の図の太枠を分母とし、色で染めた部分を分子とする、評価です。
実際に正解であるテストのうち、正解と当てたものがどれだけあるか、なので、
要するに、正解をどれだけ思い出せたか（思い出す＝Recall）、を評価します。

適合率と再現率はトレードオフ

確率0.4以上も積極的に陽性と判定するとすると、適合率は下がるが、再現率は上がる。

安全をとって確率0.6以上が陽性と判定するとすると、適合率は上がるが、再現率は下がる



適合率と再現率とはトレードオフの関係にあります。
今、２値問題で、確率がある値を境界にして、陽性か陰性かを判定するとします。
安全サイドをとって、境界を大きな値に設定すれば、
陽性の信頼性を示す適合率は上がり、陽性をどれだけもれなく拾えたかの再現率は下がります。
逆に、積極的に陽性検出をしようとしたら、
陽性の信頼性を示す適合率は下がり、陽性をどれだけもれなく拾えたかの再現率は上がります。

評価指標：F1スコア

$$\frac{2*適合率 * 再現率}{適合率 + 再現率}$$

適合率と再現率のバランスをとるための指標

そこで、両者をバランスよく見たいときに、両者をミックスした指標が使われます。
それが、F1スコアです。

正則化

正則化

ロス＝ロス関数＋正則化項

学習に味付け済みのロスを使うことで、過学習しにくかったり、滑らかなモデルになったりする

モデルのパラメータが、どれだけ望ましいか（望ましくないか）の基本尺度。モデルの推論結果と正解ラベル間の距離とか。

パラメータが極端な値をとったときに罰則を与える味付け

L0ノルム：0でないパラメータの数
L1ノルム：パラメータの絶対値の和
L2ノルム：パラメータの二乗和の平方根

https://ja.wikipedia.org/wiki/%E6%AD%A3%E5%89%87%E5%8C%96

100本ノック第6章課題50～56,58

- 「100本ノック」の6章の課題を解いてみましょう。
- 「NLP、機械学習.ipynb」というノートをコピーし、冒頭の準備をやった後、各課題のセクション下のコードセルのコードを完成させ、実行ログを残してください。
- 以下に、上記の予備知識に加えて、課題を解く際に参考となることを説明します。

課題を解くための参考情報

55補足

- `confusion_matrix(y_train, lr.predict(x_train))`
  - 第1引数は正解ラベル、第2引数は予測結果ラベル

予測分類

bをeと間違えた

正解分類	b	e	m	t
b	4514	0	0	0
e	0	4225	0	0
m	0	0	723	0
t	0	0	0	1210

56補足

- classification_report(y_test, lr.predict(x_test))
  - 第1引数は正解ラベル、第2引数は予測結果ラベル

test内個数

	precision	recall	f1-score	support
b	1.00	1.00	1.00	569
e	1.00	1.00	1.00	563
m	1.00	1.00	1.00	76
t	1.00	1.00	1.00	126
accuracy			1.00	1334
macro avg	1.00	1.00	1.00	1334
weighted avg	1.00	1.00	1.00	1334

ラベルごとのPrecision等を
単純に平均したもの

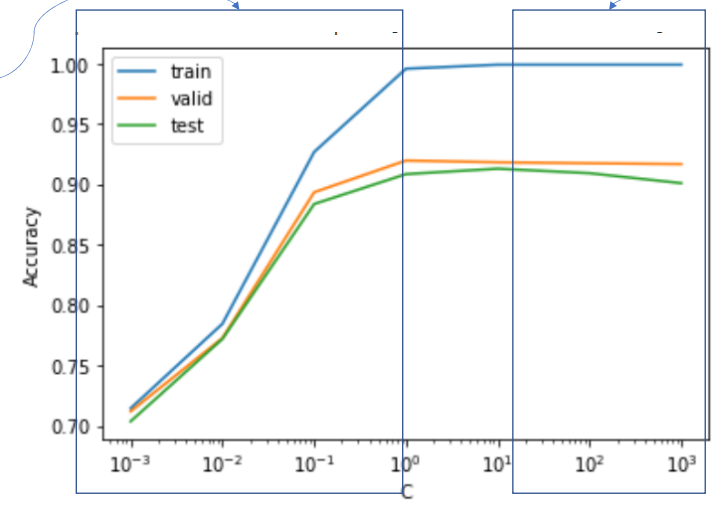
ラベルごとのPrecision等を
supportの数で重みづけして平均したもの

micro avgは、すべてのラベルに関して、
True Positive 等をカウントして、求める

58補足

正則化強すぎると、
うまく学習しない

正則化項が弱いと、
過学習で、trainにfit
過ぎて、客観的には
劣化している



正則化項強い ⟷ 正則化項弱い

確認クイズ

- スタログの確認クイズをやってください。