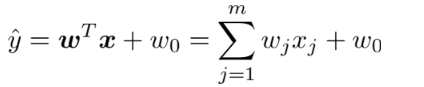
**まとめ要点１：**

　線性回帰。

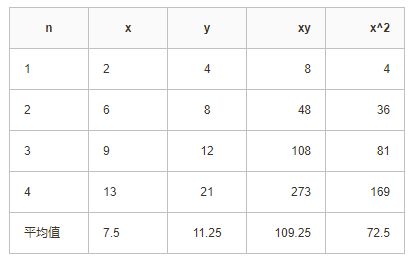
1. 回帰問題は連続の入力値から連続の出力値を予測する問題である。
2. 線性回帰モデルは入力とｍ次元パラメータの線性結合を出力モデル。

イメージ図：

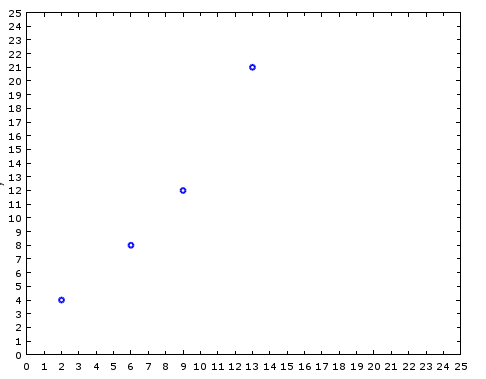
1. 出力ｙを表す式は一般的にで表示する。
2. 上記の式に、wは各パラメータ(回帰係数)、w0は誤差である。
3. パラメータを推定する方法はいくつかありますが、再急降下法は後でまとめるので、最小二乗法を使うことで、各回帰係数を求めることが可能である。
4. ここで、回帰係数と誤差は、ハイパーパラメータではない、学習で決めるパラメータである。
5. データは学習用と検証用、2種類を分けて使うことは一般的です。学習用、はパラメータを求めること時使うデータである、検証用はモデルの汎化性能を測定用である。
6. 最小二乗法で線性モデルを求める例：

例えば：線性モデルを1625920764(1)で仮定する。

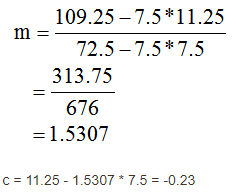
以下のデータで、最小二乗法を使うことで、線性モデルを求める。



データの分布図：



最小二乗法を使うことで、回帰係数ｍと誤差ｃを求める：



だから、線性モデルを回帰モデルは以下：

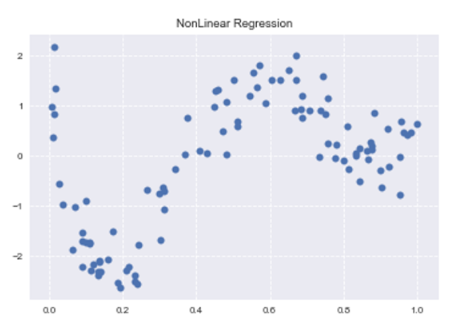
1625920949(1)

**まとめ要点２：**

非線性モデル

１、非線性のデータを予測するため、非線性モデルを使うことで、予測することが可能性である。非線性モデルは基底関数と呼ばれる既知の非線形関数とパラメータベクトルの線型結合を使用してるモデルである。

非線性回帰のイメージ図：



1. 未学習と過学習

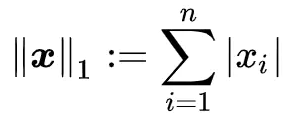
簡単に言うと、未学習はモデルの表現力は低い、正しくデータを適合することが出来ない状態である。一般的に、学習するデータが少ないどうか、モデル自体の表現力は低いどうかは原因である。

逆に、過学習は学習する時、真のデータとの誤差は小さいですが、検証用のデータを使て誤差を計算すると、誤差は大きくなる状態。いくつか対策方法があります。１、シンプルのモデルを使うことでモデルの複雑さを低くする(変数の数を減る)。

２、正則化する。

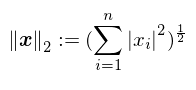
３、L2ノルムとＬ1ノルム

Ｌ1ノルムの定義は以下の式：

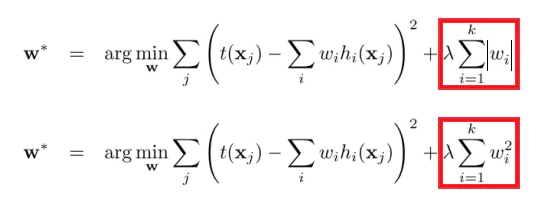


つまり、絶対値足せば求めることが出来る。

L2ノルムはよく見られるので、以下の式で求めることが出来る：



L2、L1ノルムは正則化で使用する時の式：



L2、L1ノルムの役割は似ているですが、違うところもあります。パラメータを出力する時、L2(ridge)値は0に近づけるよう推定しますが0にならないである、L1(lasso)を使う時、0になる可能性があるので、区別して覚えたほうがいい。

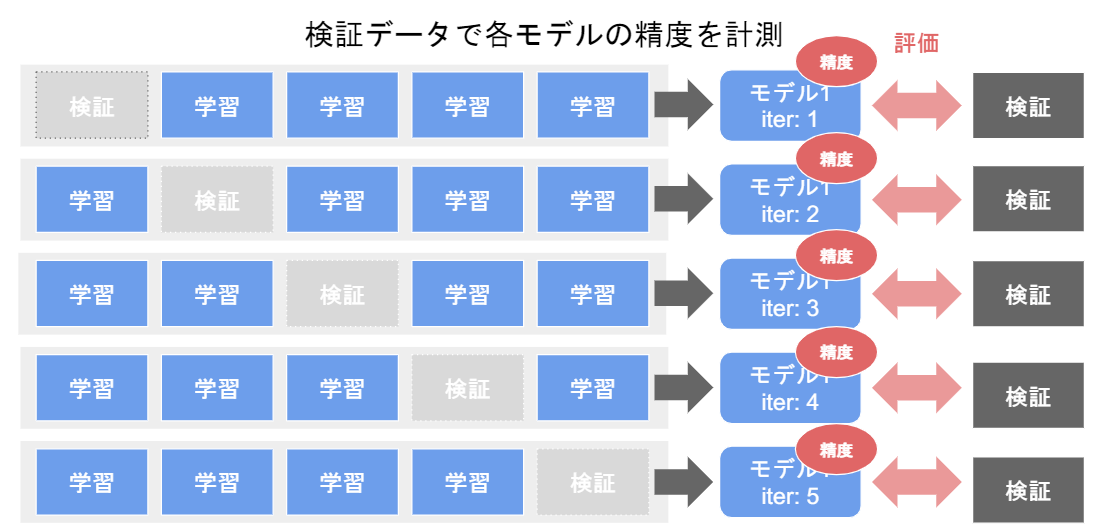
1. 誤差検証
2. ホールドアウト法

有限のデータを学習用とテスト用の2つに分割し、学習と検証用分けて使用。この方法は、大量のデータが必要というデメリットがある、大量のデータを持っていない場合、正しく性能評価を与えることは難しいである。

1. クロスバリデーション法

クロスバリデーション法はホールドアウト法と違うところは、データを学習用と検証用を分ける方法が違う。

イメージ図は以下：



大量のデータが持ってない場合、クロスバリデーション法を使うのはお勧めである。

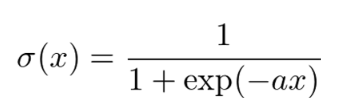
**まとめ要点３：**

　1、ロジスティクス回帰モデル。

ロジスティクス回帰モデルの出力(目的変数)は「０」または「１」なので、よく分類問題を解決する時、使うモデルである。

分類問題を解くための教師ありモデルの中で、Activation Functionのsigmoid関数よく使われる。

Sigmoid関数の出力は0~1の間、確率の性質も表現出来る、しかも単調増加関数：



sigmoid関数はよい性質を持っています。sigmoid関数を微分すると：

1625924941(1)簡単な式が計算出来るので、尤度関数を求める際に、sigmoid関数の微分は簡単になれる。

２、ロジスティクス回帰の使い方：

sigmoid関数を使う前に、線性や非線性関数のように、データの線性結合を計算する。

計算した結果はsigmoid関数のｘに導入し、データが与えられた時に、結果は１または０に、なる確率を求めることが出来る。式は以下：

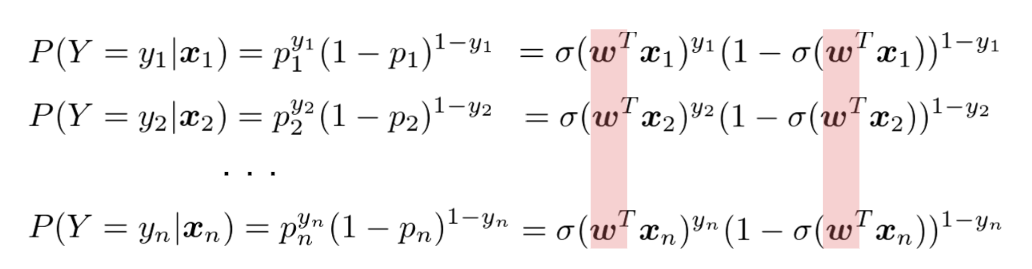
1625925355(1)

データYは確率が0.5以上ならば１、未満ならば０の予測結果を出します。

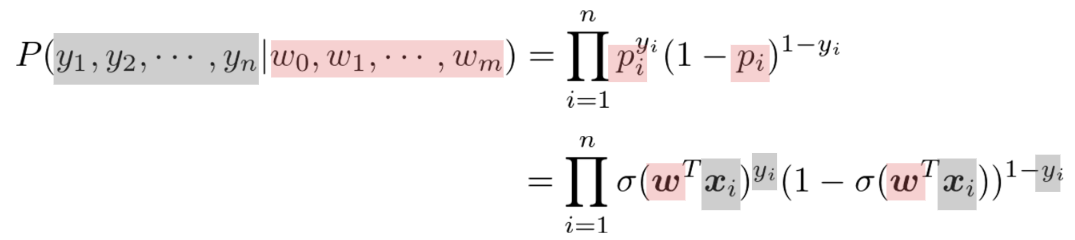
３、最尤法

尤度関数はデータを固定し、パラメータを変化させて、尤度関数を最大化するようなパラメータを選ぶ推定方法は最尤推定という。

ロジスティック回帰モデルの最尤推定を計算する時の式は微分後のsigmoid関数を合わせて見ると、分かりやすくなれる。



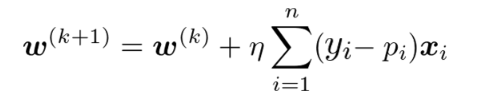
全てのデータを結合して得られた尤度関数は：



尤度関数を実際に計算する時、計算を簡単にするため、対数を取るのは一般的、対数尤度関数と呼ぶ。対数尤度関数を最大化すれば、パラメータを求めることが出来出る。

1. 勾配降下法

勾配降下法とは反復学習によりパラメータを逐次的に更新するアプローチの一つ。式は以下のように



式の意味は、ｋ＋1回データを使てパラメータを更新する時、ｋ回のデータと学習率と誤差結果が使って求めることが可能である。

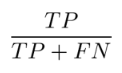
勾配降下法では、パラメータを更新するのにN個全てのデータに対する和を求める必要があるので、データ量は大きい場合、計算時間が長くなったり、メモリ容量が足りない可能性が存在する、そのため、確率勾配降下法を使って、毎回更新する時、全てデータを使うではなく、ランダムにデータを一つずつ使て更新する。

1. 混同行列



モデルの性能評価を行う際に、以下指標はよく使われる

再現率：「本当にPositiveなもの」の中からPositiveと予測できる割合、式は以下



適合率：モデルが「Positiveと予測」したものの中で本当にPositiveである割合

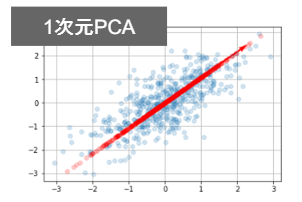
1625928083(1)

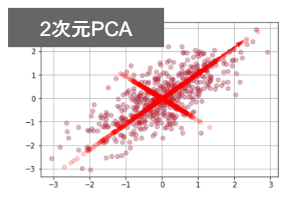
**まとめ要点４：**

主成分分析(PCA)

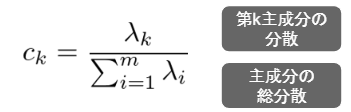
　主成分分析とは多変量データの持つ構造をより少数個の指標に圧縮したい時、使う手法である。

例えば：2次元のデータを1次元に圧縮する時、線形変換後の変数の分散が最大となる射影軸を探せば、第一主成分を求めることが可能である。





寄与率：第k主成分の分散の全分散に対する割合



**まとめ要点４：**

k近傍法:

最近傍のデータをｋ個取ってきて、それらがもっとも多く所属するクラスに識別

例えば：ある点の分類問題解く時、ｋ＝３の時、隣の点は、青い、青い、赤の場合、、その点は青いと分類される。

実装例は：データは以下の場合

|  |  |  |
| --- | --- | --- |
| 映画の種類 | 戦い回数 | KISS回数 |
| 愛情 | 0 | 4 |
| 1 | 3 |
| アクション | 4 | 0 |
| 5 | 1 |

もし、新しく入力のデータの戦い回数は３回、KISS回数は２回の時、どの映画の種類に分類するかの問題。

定義用：



出力の結果はアクションであること。

k平均法:

k平均法(k-means)のアルゴリズム

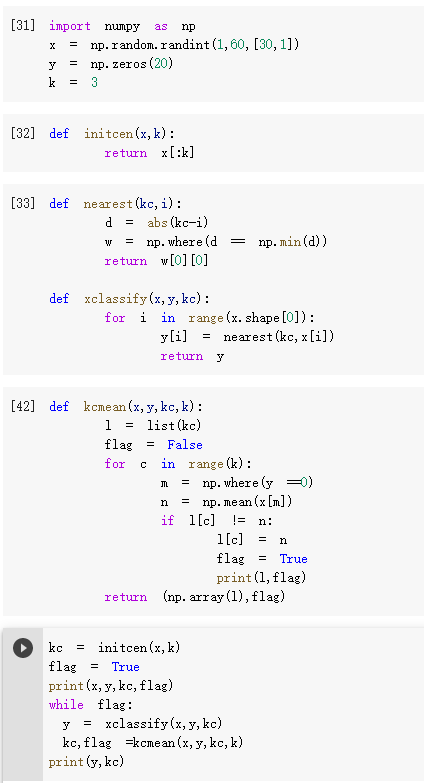
各クラスタ中心の初期値を設定する

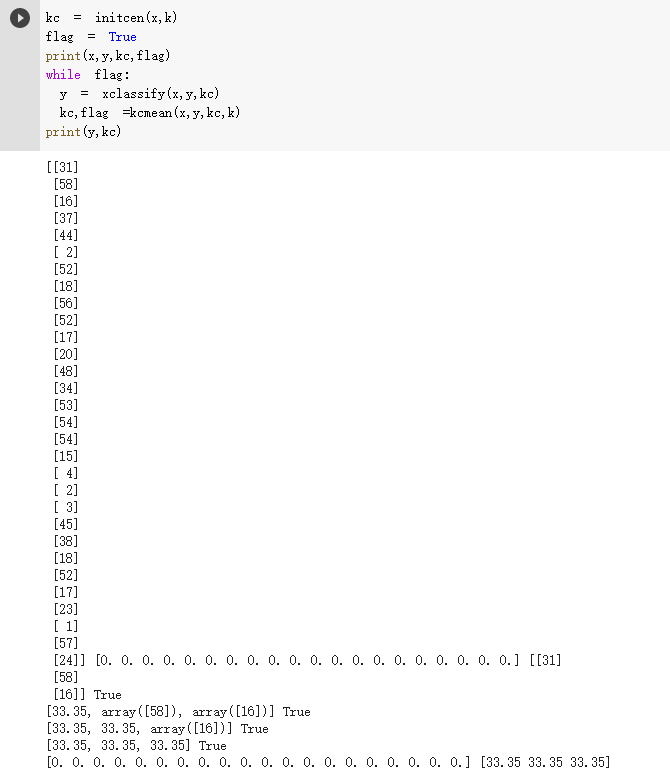
各データ点に対して、各クラスタ中心との距離を計算し最も距離が近いクラスタを割り当てる

3) 各クラスタの平均ベクトル（中心）を計算する

4) 収束するまで2, 3の処理を繰り返す次から各手順の詳細を説明するk-平均法(k-means)

実装例は、ランダムに数字を選んで、中心の初期値を表す、そして、各データと中心の距離を計算して、一番近いクラスに割り当てる、そして各クラスの平均を新たの中心として計算する、最後、クラス中心と結果の値は変わったのかを確認する。



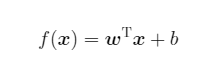
実装結果：

**まとめ要点５：**

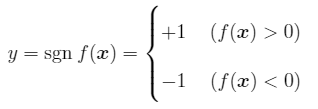
SVM

SVMとは、2クラス分類問題を解決するための手法の一つである。

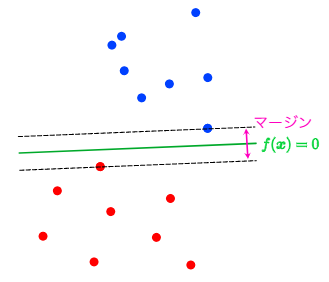
決定関数は通常のように：



sgn()は符号関数と呼ばれる関数で、引数が正の場合には+1、負の場合には1を返す関数です。



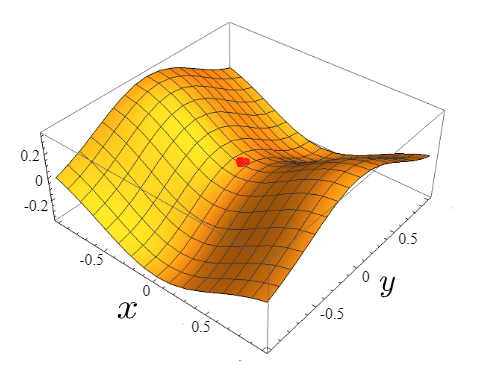
マージンとは：分類境界を挟んで2つのクラスがどのくらい離れているかをマージン(margin)と呼びます、イメージ図は以下のように、マージンを最大化することで、最適のSVMを求めることが出来る。分類境界に最も近いデータはサポートベクトル(support vector)と呼びます。



一般的SVM(ハードマージン)は線性分離可能のデータしか処理出来ないですが、線性分離不可のデータに対して、ソフトマージンを使う可能性もあります。

停留点：関数の微分係数が0となる点

鞍点：1変数には見られない鞍点(saddle point)22と呼ばれる停留点(赤い点)。イメージ図は以下：



カーネル法：SVM中、特徴空間上の目的関数を最大化時、特徴空間上での分類境界を決定し、入力空間に戻すことで元データに対する非線形分離が可能となるなずだが、次元拡張により内積部分の計算量が莫大になるため、内積部分をカーネル関数(kernel function)と呼ばれる関数で置き換える方法である。カーネル法を使うことで、計算量を軽減することが可能である。