機械学習入門

第6回～10回　教師あり学習/ロジスティック回帰

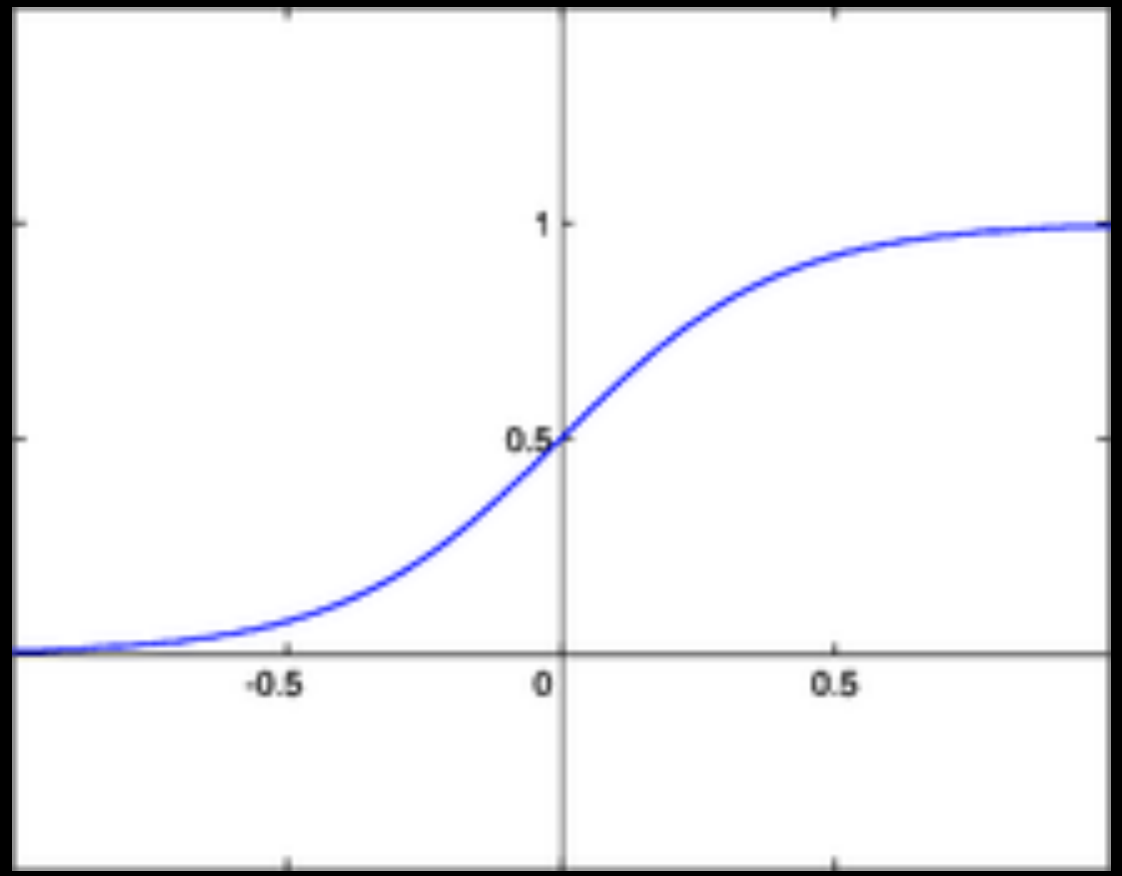
ロジスティック回帰

- 回帰とは言っているが分類に使われる。
- ある確率(閾値)以上を1、ある確率以下を0として2値分類に適用
- Ex)テストのデータから合格か不合格かを分類する、腫瘍データから悪性か良性かを分類する

ロジスティック回帰

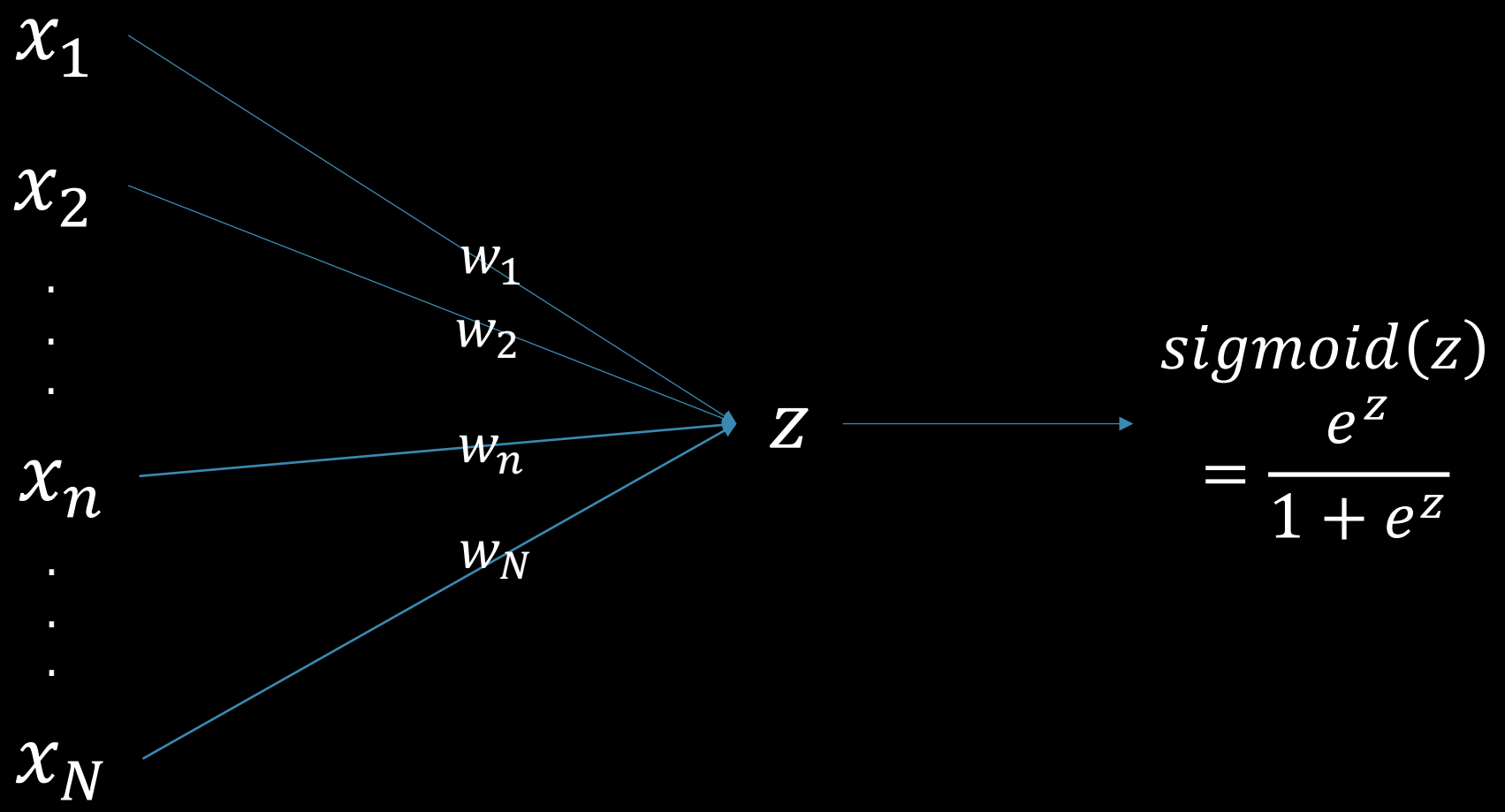
y = Xw+bをシグモイド変換する
->確率として表せる
（右図：シグモイド関数）

$$sigmoid(z) = \frac{e^z}{1 + e^z}$$



ロジスティック回帰（アーキテクチャ）

w_n:n次元目のデータにかかるパラメータ



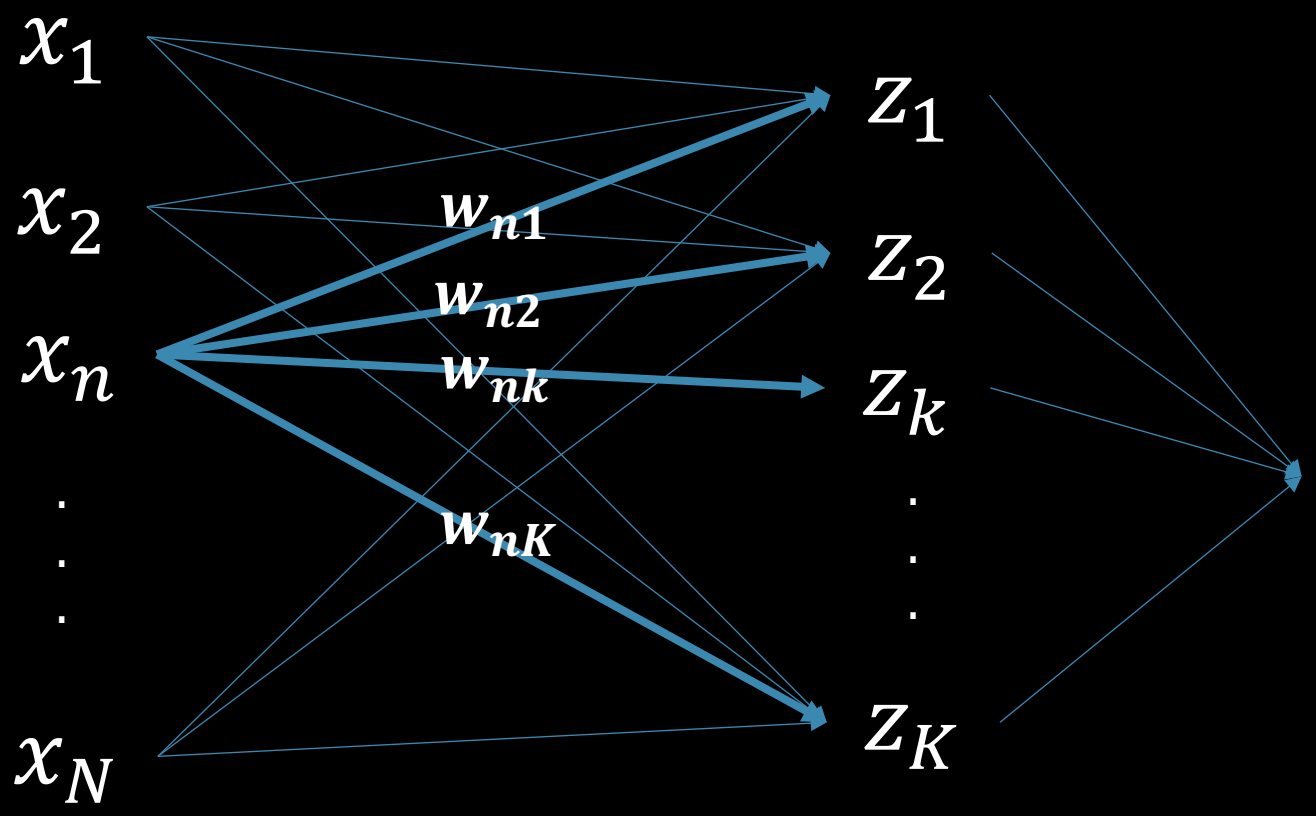
実装してみる

ソフトマックス回帰

- 多クラス分類のためのモデル
- ロジスティック回帰モデルと同様，確率を出力する

ソフトマックス回帰（アーキテクチャ）

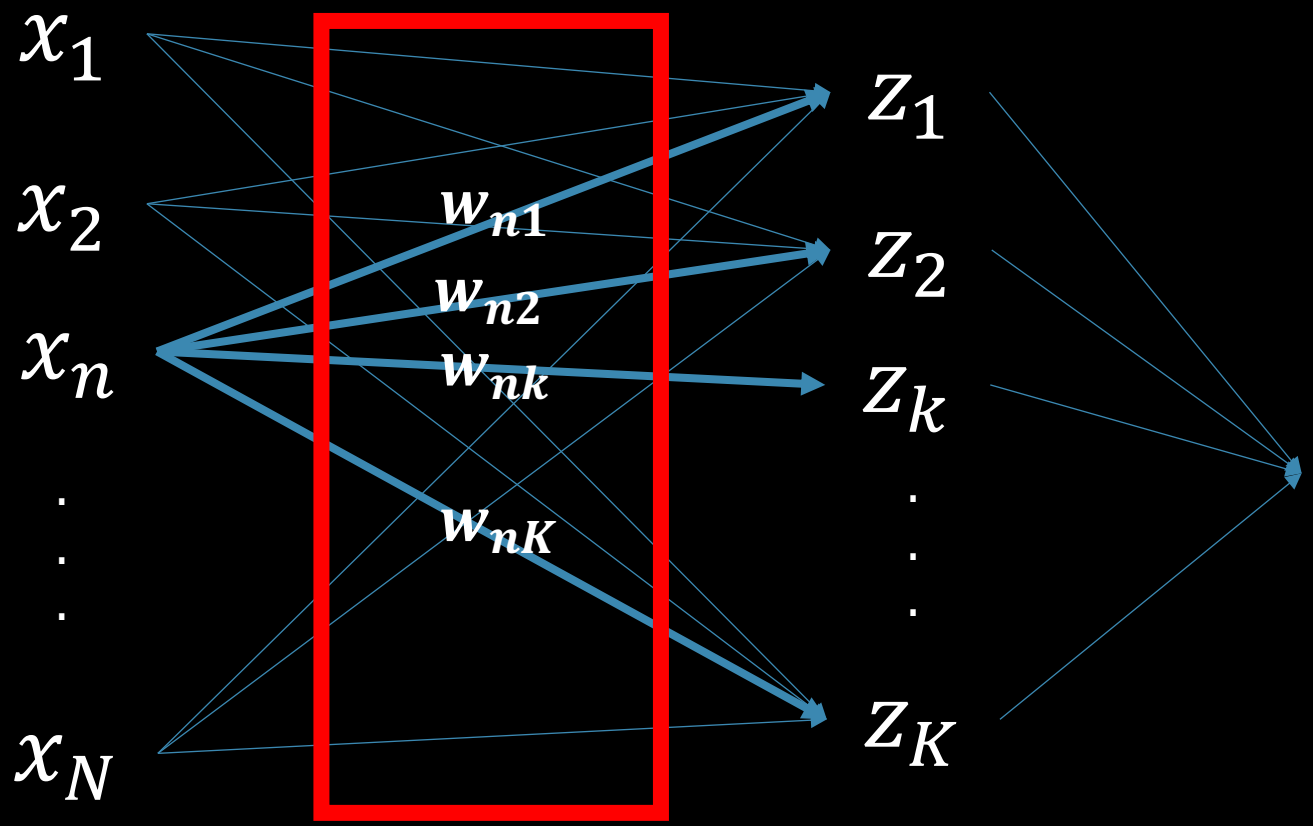
w_{nk}:n次元目のデータからk番目のクラスへと向かうパラメータ



$$softmax(z_1, z_2, z_k, \cdots z_K) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

ソフトマックス回帰（アーキテクチャ）

w_{nk}:n次元目のデータからk番目のクラスへと向かうパラメータ

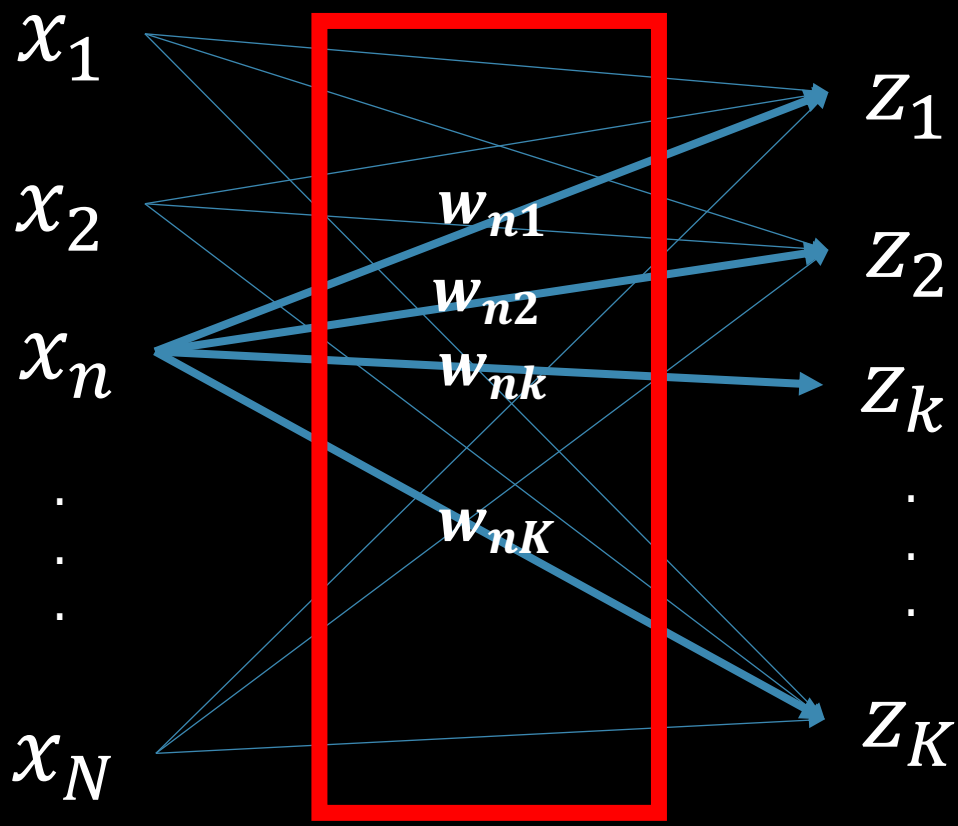


$$softmax(z_1, z_2, z_k, \cdots z_K) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

①重みを掛け算して足し合わせる.

ソフトマックス回帰（アーキテクチャ）

w_{nk}:n次元目のデータからk番目のクラスへと向かうパラメータ



①重みを掛け算して足し合わせる.

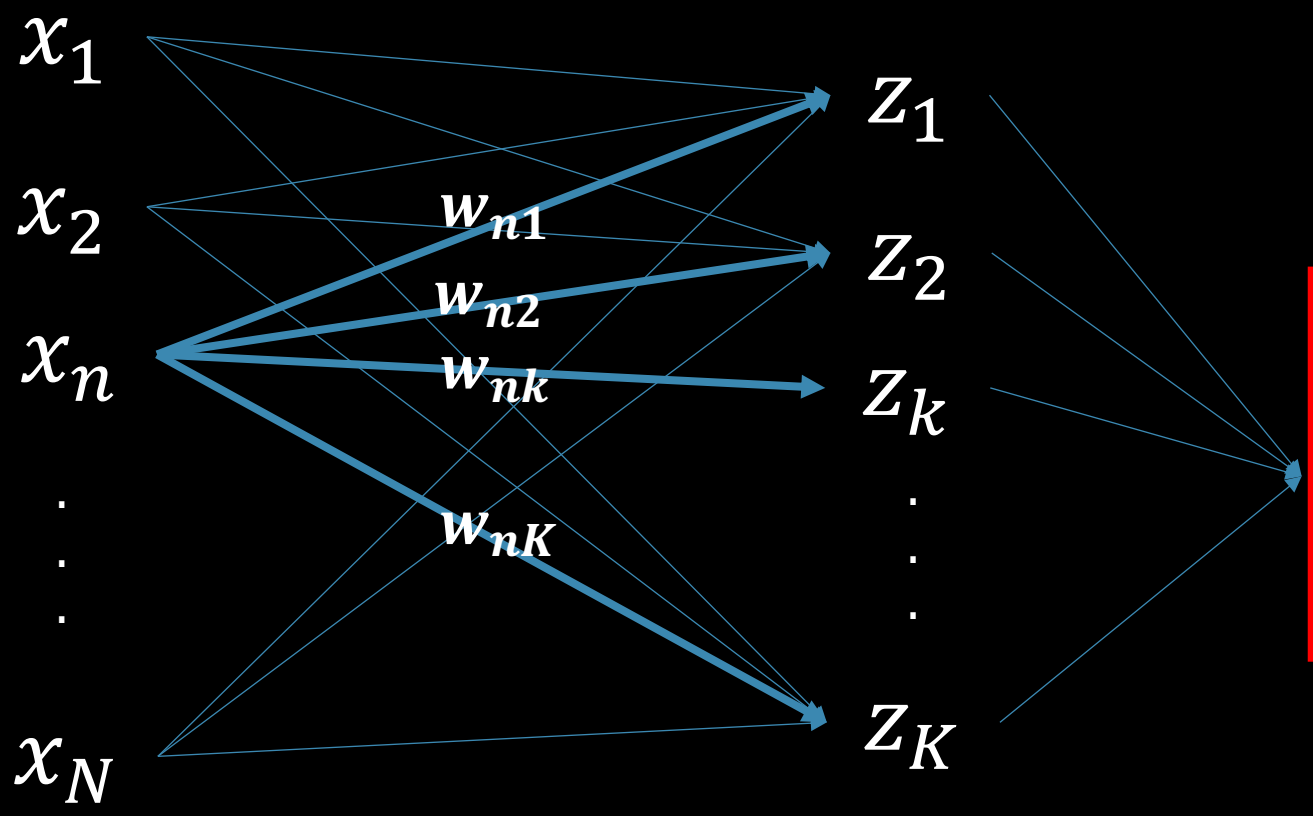
重みの掛け算、今までと何が違う？
→いろんなクラスラベルがあるので，
それぞれ計算する必要がある！

$$z_1 = w_{11}x_1 + w_{21}x_2 + \cdots w_{N1}x_N$$
$$z_2 = w_{12}x_1 + w_{22}x_2 + \cdots w_{N2}x_N$$
$$\vdots$$
$$z_K = w_{1K}x_1 + w_{2K}x_2 + \cdots w_{NK}x_N$$

少し複雑だが，慣れよう！
（たくさんシグモイドがあると思えばわかりやすいかも？）

ソフトマックス回帰（アーキテクチャ）

w_{nk}:n次元目のデータからk番目のクラスへと向かうパラメータ

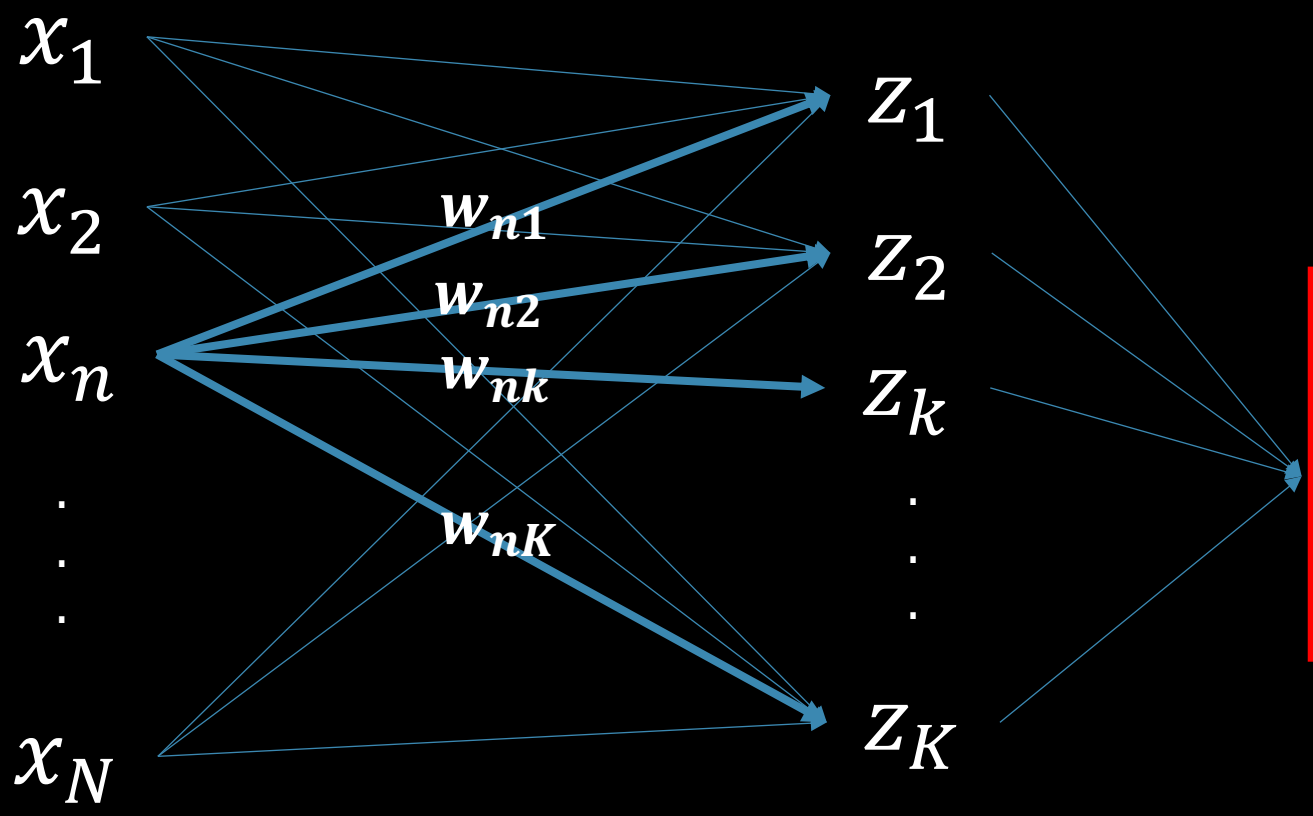


$$softmax(z_1, z_2, z_k, \cdots z_K) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

②確率を出力.

ソフトマックス回帰（アーキテクチャ）

w_{nk}:n次元目のデータからk番目のクラスへと向かうパラメータ



$$softmax(z_1, z_2, z_k, \cdots z_K) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

②確率を出力.

過学習(overfitting)と適合不足(underfitting)

- 過学習とは、学習データから得られたモデルが複雑すぎて学習データに過度な一致をしてしまう状態。
- 適合不足とは、学習データから得られたモデルが単純すぎて訓練データにもテストデータにも上手くfitしない状態。

→**防ぐには、学習データの過度なfittingを抑えながらデータ量を増やしたり、モデルを複雑化していく。**

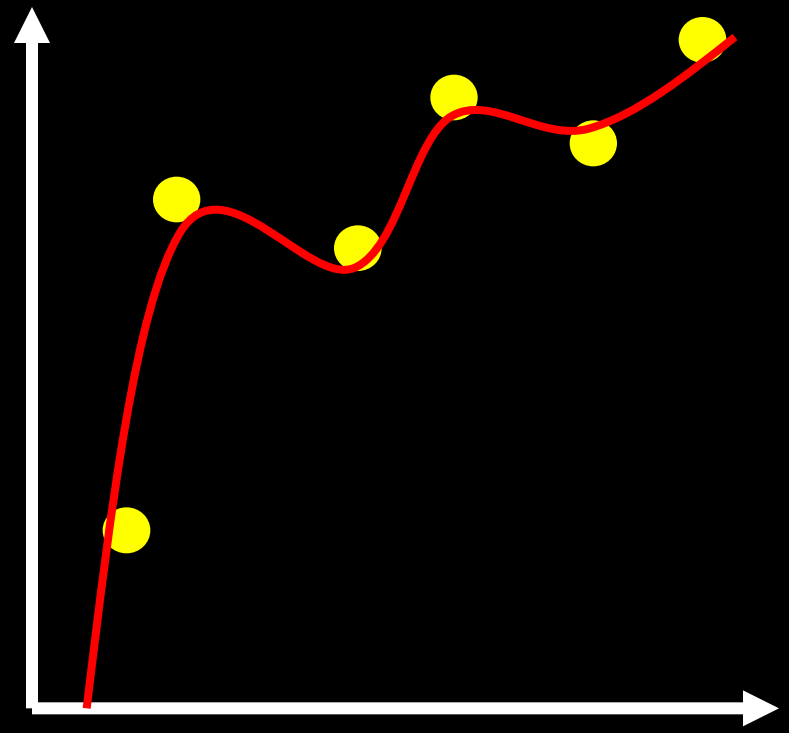
正則化

過学習を防ぐ手法の一つ。

モデルに条件(罰則項)を加えることで、モデルが複雑になりすぎるのを防ぐ。
＝モデルを単純化する。

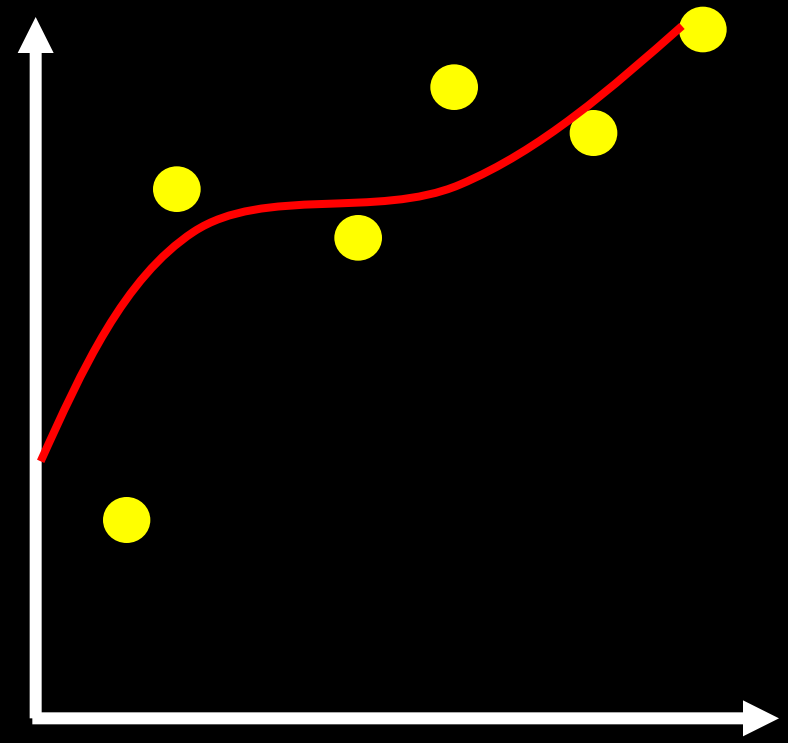
正則化

過学習！



正則化

ちょっとマシ。



正則化

- L1正則化
- L2正則化

の2種類、どちらも最適化したい関数に何らかの罰則項を追加したもの。

≒過剰に最適になりすぎない

→個々の説明変数が、過度に影響しないようにその係数が0に近くなるように調整する。

L1正則化

- いくつかの係数を 0 に出来る≒次元圧縮的な効果。
- 自動的に特徴量を選択している、とも言える。
- 特徴量が減るので、結果の解釈が容易になる。

＝どの特徴量が重要かが分かりやすくなる。

＊罰則項が絶対値であったりと、解析的(微分するなど)に最適化できない

L2正則化

- 係数が0にならない場合がある。
- L1では次元圧縮を行っているため、L2の方が精度が高い傾向がある。

正則化

- どちらが良いか、は目的による。
- 特徴量のうち重要な物がわずかしかない、解釈しやすいモデルがほしいなら、L1正則化を行う。
- 基本的にはL2で試してみるのが良い。
- L1とL2の２つの正則化を用いても良い(罰則項を選択するのにコストがかかる)

正則化LogisticRegressionを実装する

- Sklearnではデフォルトで L2正則化が行われている。
- そのパラメータ(罰則項を決める物)はCという値。
- このCをいじって結果の精度がどうなるかを見てみる。

C値

- Cを大きくすると正則化の影響が小さくなる。つまり、訓練データにより適合したモデルが得られる。
- Cを小さくすると係数を0に近づけるように働く。つまり、モデルをより単純化しようとする。

数値を眺める

- C=100

訓練：0.9812206572769953

テスト：0.965034965034965

Default(C=1):

訓練： 0.9530516431924883

テスト：0.958041958041958

数値を眺める

- C=0.01

訓練：0.9530516431924883

テスト：0.951048951048951

Default(C=1):

訓練： 0.9530516431924883

テスト：0.958041958041958

Subject2

- Titanicのデータを使ってやってみる

HINT

```
drop_df = ["sibsp","parch","fare","embarked","class","who","adult_male","deck","embark_town","alive","alone"]

titanic_data = titanic_data.drop(columns = drop_df, axis = 1)

titanic_data["age"] = titanic_data["age"].fillna(titanic_data["age"].mean())

titanic_data["sex"] = titanic_data["sex"].map({"male": 1, "female":0})
```