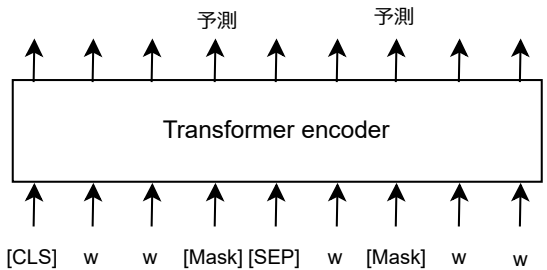
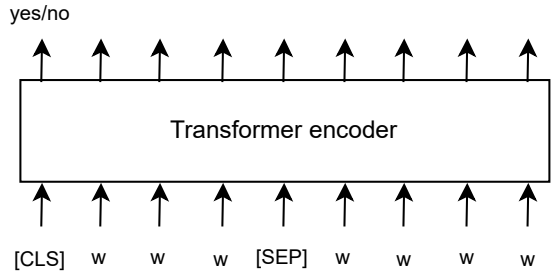
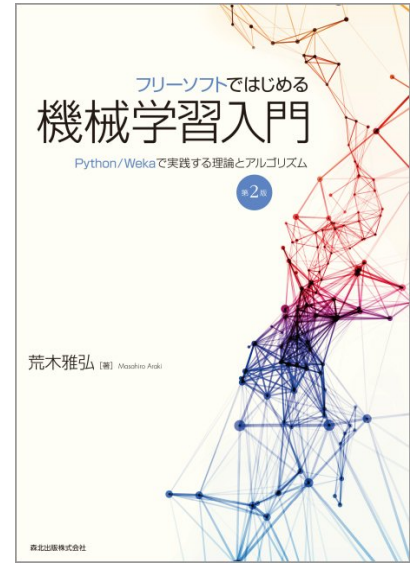
フリーソフトではじめる機械学習入門(第2版)

14. 半教師あり学習





- 14.1 半教師あり学習とは
- 14.2 ～ 14.5 各種の半教師あり学習手法
- データ拡張
- 自己教師あり学習



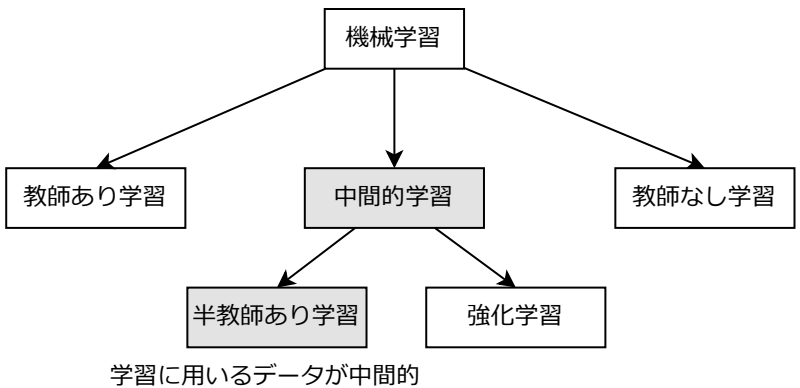
- 荒木雅弘:『フリーソフトではじめる機械学習入門(第2版)』(森北出版, 2018年)
- スライドとJupyter notebook
- サポートページ

正解付きデータが少ない状況に対するアプローチ

- 半教師あり学習
  - 少量の正解付きデータと大量の正解なしデータがある状況での学習
- データ拡張
  - 入力の多様性が規則として再現可能な場合の方法
  - 学習データおよび入力データに対する変換として実装
- 自己教師あり学習による事前学習
  - 入力の一部を正解とする自己教師あり学習による表現学習の後、少量のタスク内データで事後調整
  - プロンプト調整のみの Few-shot / Zero-shot 学習 も可能

14.1 半教師あり学習とは

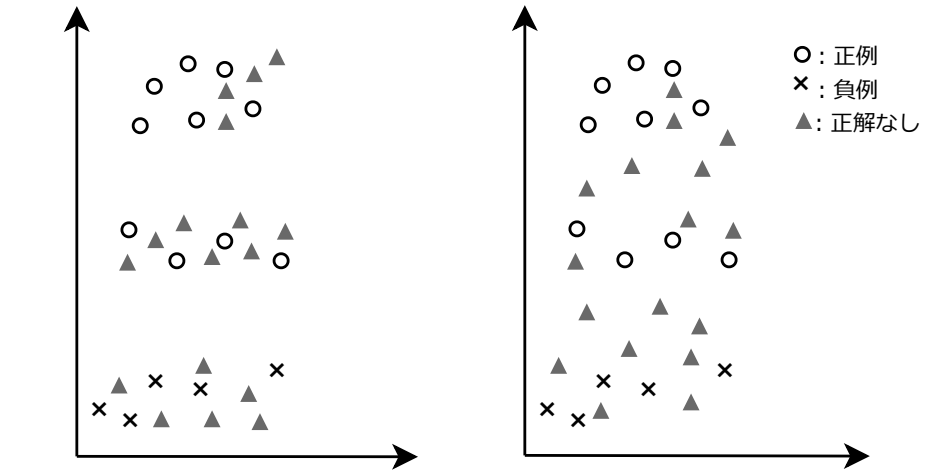
- 半教師あり学習の状況設定
  - 正解情報が一部の学習データにのみ与えられている
    - 例：tweet のP/N判定
  - データ自体はクローラ等で容易に収集できる
  - タグ付け作業にコストが掛かり、大量の正解付きデータが得にくい



学習に用いるデータが中間的

14.1.1 数値特徴の場合（1/2）

- 半教師あり学習に適した数値特徴データの条件
  - 正解なしデータから得られる $p(\boldsymbol{x})$ に関する情報が $p(y|\boldsymbol{x})$ の推定に役立つこと



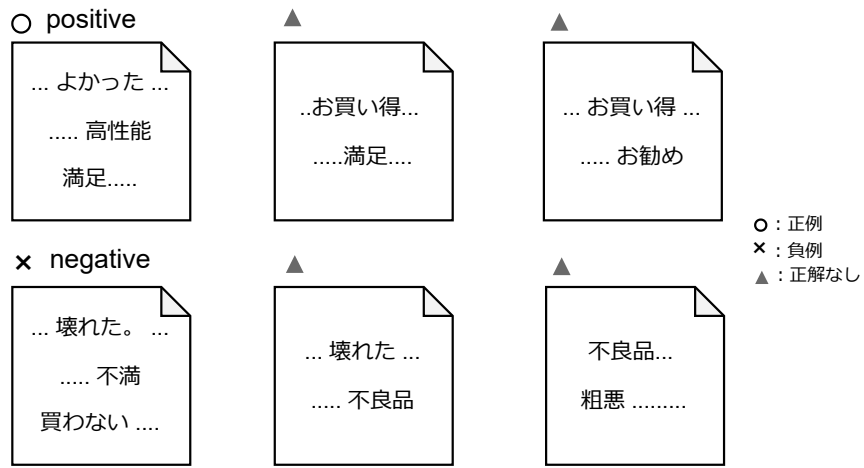
(a) 半教師あり学習に適するデータ　(b) 半教師あり学習に適さないデータ

14.1.1 数値特徴の場合（2/2）

- 半教師あり学習に適した数値特徴データが満たすべき仮定
  - 半教師あり平滑性仮定
    - 二つの入力が高密度領域で近ければ、出力も関連している
  - クラスタ仮定
    - 入力が同じクラスタに属するなら、それらは同じクラスになりやすい
    - 低密度分離（識別境界は低密度領域にある）
  - 多様体仮定
    - 高次元のデータは、低次元の多様体上に写像できる
    - 多様体：局所的に線形空間と見なせる空間

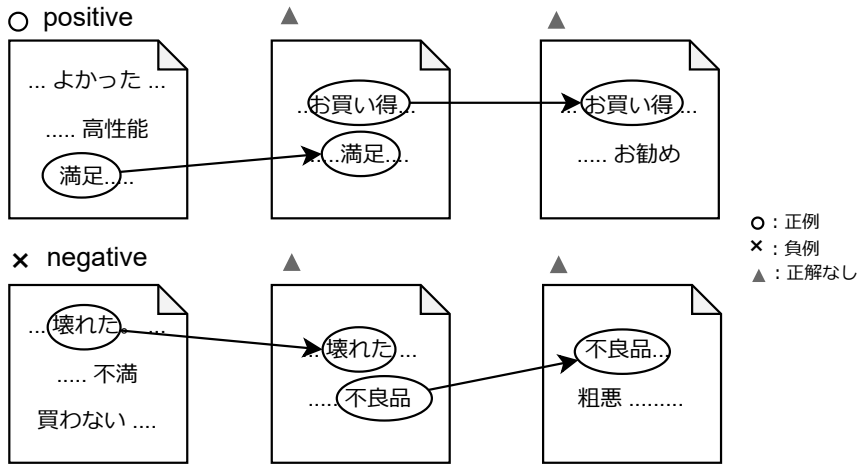
14.1.2 カテゴリ特徴の場合（1/2）

- 半教師あり学習に適したカテゴリ特徴データ
  - 識別に役立つ特徴値の伝播が生じる場合
  - 例：文書のP/N判定
    - 特徴語が抽出できていると仮定



14.1.2 カテゴリ特徴の場合（2/2）

- 文書分類タスクにおける特徴の伝播
  - オーバーラップした特徴語によって、判定に寄与する新たな特徴語が見つかる



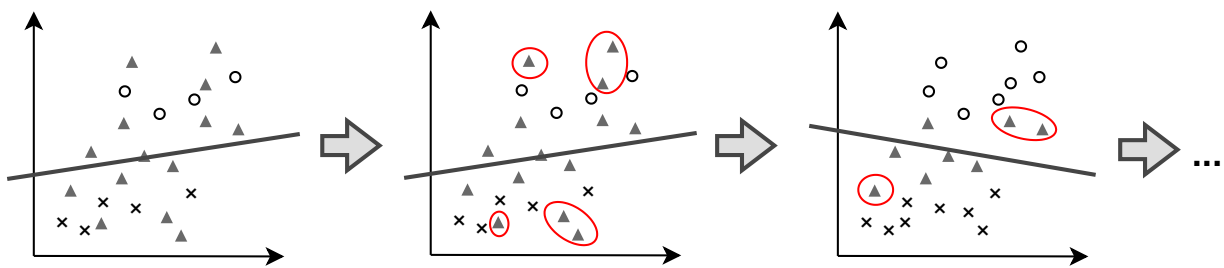
14.1.3 半教師あり学習のアルゴリズム

- 半教師あり学習の手順
  - 繰り返しに基づくアルゴリズム
    - 正解付きデータで作成した初期識別器を、正解なしデータを用いて逐次的に更新する
  - データの散らばりに基づくアルゴリズム
    - 正解付きデータの近くにある正解なしデータを利用して、識別可能な空間を広げる

14.2 自己学習 (1/6)

- 自己学習とは
  - もっとも基本的な繰り返しに基づく半教師あり学習
- 自己学習のアルゴリズム

1. 正解付きデータで初期識別器を作成
2. 正解なしデータを現在の識別器で識別し、確信度の高いものを正解付きデータとみなす
3. 新しい正解付きデータを加えて識別器を再学習
4. 事前に決めた回数または正解なしデータがなくなるまで2, 3を繰り返す



14.2 自己学習（2/6）

- 自己学習における識別器に対する要求
  - 確信度の出力：正解なしデータの識別結果を信用するかどうかの判定に必要
- 自己学習の性質
  - クラスタ仮定や低密度分離が満たされるデータに対しては、高い性能が期待できる
  - 低密度分離が満たされていない場合、初期識別器の誤りが拡大してゆく可能性がある

14.2 自己学習 (3/6)

- 例題:irisデータを使って自己学習

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.semi_supervised import SelfTrainingClassifier
from sklearn.metrics import classification_report

# irisデータの読み込みと学習用・評価用の分割
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

14.2 自己学習 (4/6)

- 擬似的に一部のデータを正解なしとする

```
rng = np.random.RandomState(1)
unlabeled = rng.rand(y_train.size) < 2/3
labels = np.copy(y_train)
labels[unlabeled] = -1
labels
```

```
array([-1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,  2, -1,  0, -1,  2, -1,
       -1, -1, -1,  1,  2, -1,  0,  1,  1, -1, -1, -1,  1, -1, -1,  0, -1,
        0, -1,  2,  2, -1,  0,  1,  0, -1,  1, -1, -1,  1, -1, -1, -1, -1,
        1, -1, -1, -1, -1, -1, -1, -1,  2, -1, -1,  0, -1, -1, -1, -1, -1,
        2, -1,  0, -1, -1,  1, -1, -1,  1, -1,  0,  1,  0, -1,  1, -1, -1,
        1, -1,  1, -1, -1, -1,  2, -1, -1, -1, -1,  2, -1, -1, -1])
```

14.2 自己学習 (5/6)

- `SelfTrainingClassifier` で半教師あり学習
  - 識別器には `SVC`(デフォルトのRBFカーネル)を選択
    - 識別結果に確率を付ける
    - ハイパーパラメータ `gamma` の設定から特徴ベクトルの分散を除外

```
svc = SVC(probability=True, gamma="auto")
clf = SelfTrainingClassifier(svc, verbose=True)
clf.fit(X_train, labels)
```

```
End of iteration 1, added 50 new labels.
End of iteration 2, added 7 new labels.
End of iteration 3, added 1 new labels.
SelfTrainingClassifier(base_estimator=SVC(gamma='auto', probability=True), verbose=True)
```

14.2 自己学習 (6/6)

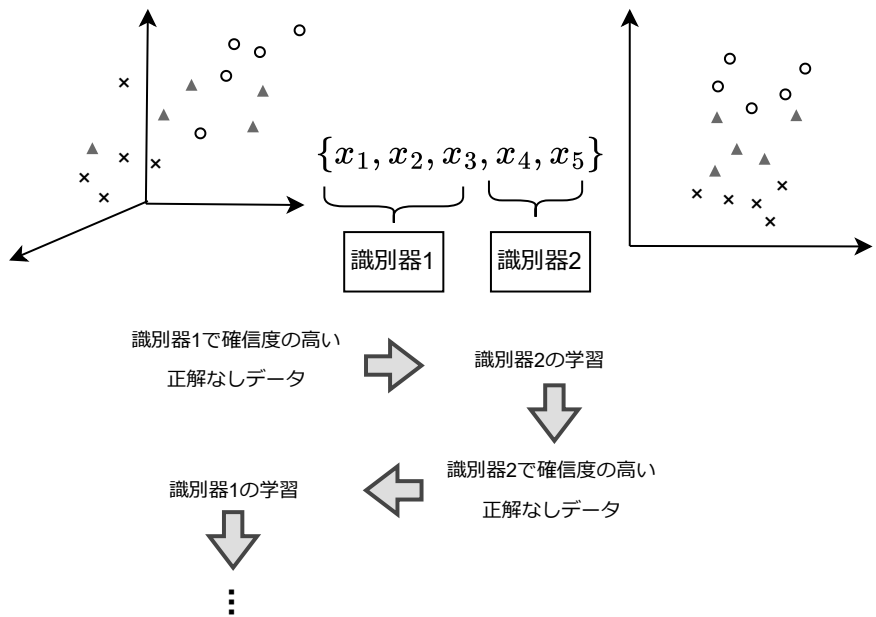
- 性能評価

```
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      0.95      0.97        19
           2       0.93      1.00      0.97        14

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50
```

14.3 共訓練 (1/2)

- 共訓練とは
  - 判断基準が異なる識別器を交互に用いる
  - 片方の確信度が高いデータを、他方が正解付きデータとみなして学習

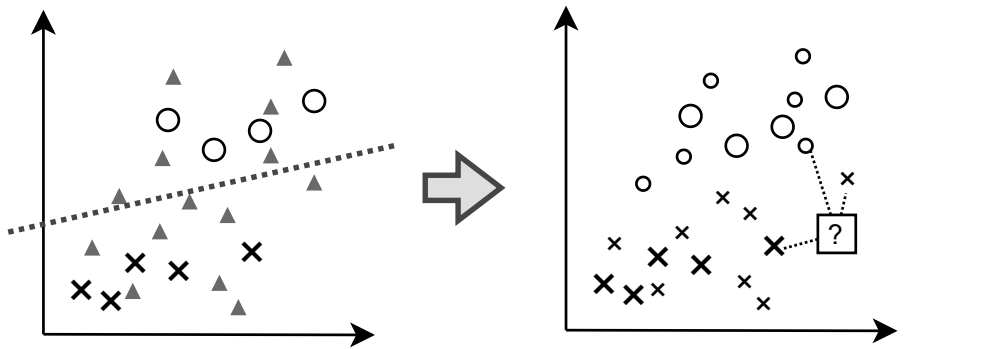


14.3 共訓練 (2/2)

- 共訓練の特長
  - 学習初期の誤りに対して頑健
- 共訓練の問題点
  - それぞれが識別空間として機能する特徴集合を、どのようにして作成するか
  - すべての特徴を用いる識別器よりも高性能な識別器が作成できるか

14.4 YATSI アルゴリズム (1/2)

- YATSI (Yet Another Two-Stage Idea) アルゴリズムとは
  - 特徴空間上のデータの散らばり方の情報を利用
  - 繰り返し学習による誤りの増幅を避ける



正解付きデータで作った
識別器で全データを識別

・正解付きデータ : 1
・識別後の正解なしデータ : $\alpha < 1$
の重みでk-NN

14.4 YATSI アルゴリズム (2/2)

- YATSIアルゴリズムのハイパーパラメータ
  - 初期識別器作成のアルゴリズム
    - 確信度出力を必要としないので、任意のアルゴリズムが選択可能
  - 正解なしデータの重み　$0 < \alpha < 1$
- YATSIアルゴリズムの欠点
  - すべての正解なしデータによる識別結果への寄与が同一

ラベル伝搬法（1/4）

- ラベル伝搬法とは
  - 特徴空間上のデータをノードとし、類似度に基づいたグラフ構造を構築
  - 近くのノードは同じクラスになりやすいという仮定で、正解なしデータの予測を行う
  - 評価関数（最大化）

$$J(f) = \frac{1}{2} \sum_{i,j \in U \cup L} w_{ij} D(f(x_i), f(x_j)) + \lambda \sum_{i \in L} D'(f(x_i), y_i)$$

    - U：正解なしデータ、L: 正解付きデータ
    - f：識別関数 $\in [-1, 1]$、D, D'：類似度関数
    - y_i：i番目のノードの正解ラベル $\in \{-1, 1\}$
    - w_{ij}：i番目のノードとj番目のノードの類似度

ラベル伝搬法（2/4）

- データ間の類似度（w_{ij}）の基準
  - RBF
    - 一定近傍内のノードに対して連続値の類似度が与えられる
    - 近傍の範囲は γ で調整（$\gamma = \frac{1}{\sigma^2}$ なので大きいほど狭い）
  - K-NN
    - 近傍の k 個のノードが結合
    - 類似度は0または1で表現

ラベル伝搬法（3/4）

- 例：半教師あり設定にしたirisデータに対して `LabelSpreading` で正解なしデータのクラス予測

```
from sklearn.semi_supervised import LabelSpreading

unlabeled = rng.rand(y.size) < 2/3
labels = np.copy(y)
labels[unlabeled] = -1

lp = LabelSpreading(max_iter=10000)
lp.fit(X, labels)
lp.score(X[unlabeled], y[unlabeled])
```

```
0.9484536082474226
```

ラベル伝搬法（4/4）

- 正解付きデータの割合の違いによる性能の変化

```
labeled_percent = [0.05, 0.1, 0.2, 0.3]
rng = np.random.RandomState(1)
num = y.size
for labeled in labeled_percent :
    score = 0
    for i in range(100):
        unlabeled = rng.rand(y.size) < 1-labeled
        labels = np.copy(y)
        labels[unlabeled] = -1
        lp.fit(X, labels)
        score += lp.score(X[unlabeled], y[unlabeled])
    print(f'labeled:{labeled*100:4.1f}%, score={score/100:6.3f}')
```

```
labeled: 5.0%, score= 0.812
labeled:10.0%, score= 0.925
labeled:20.0%, score= 0.939
labeled:30.0%, score= 0.944
```

データ拡張（1/4）

- 正解付きデータを拡張
  - 画像
    - 基本：移動、回転、拡大・縮小
      - これらの処理によって画像の意味は変わらない
    - AutoAugment
      - 対象画像に応じて様々な変換の組み合わせを強化学習で学習
  - 自然言語
    - Back translation
      - 自動翻訳後、元に戻すことでデータを水増し
    - TF-IDF word replacement
      - 意味を担っていないとみなされるTF-IDF値の低い単語を置換

データ拡張（2/4）

- 正解なしデータを拡張
  - 半教師あり学習の性能向上を目標とする
  - 損失関数を以下の2つの和とする
    - 正解付きデータに対するクロスエントロピー損失
    - 正解なしデータとそれを拡張したデータそれぞれの識別結果に対する一貫性損失
- 評価用データを拡張
  - Test Time Augmentationとよばれる
  - 多数決で識別結果を出すことによって、アンサンブル学習のような効果が期待される

データ拡張（3/4）

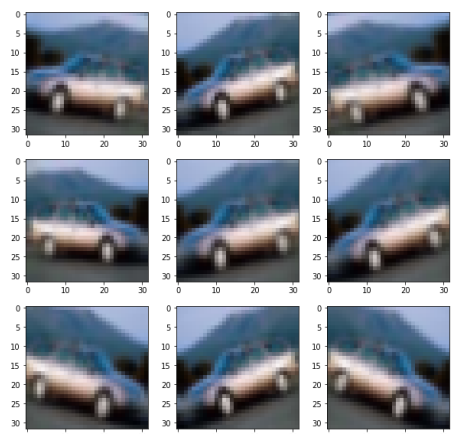
- 例:画像データ（CIFER10）を用いた正解付きデータの拡張

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

cf10 = keras.datasets.cifar10
(X_train,y_train),(X_test,y_test)=keras.datasets.cifar10.load_data()

data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.07),
        layers.RandomZoom(0.08),
    ]
)
```

データ拡張（4/4）

- 1枚の画像を拡張メソッドで変換し、結果を表示

```
fig_num = 4
plt.figure(figsize=(10,10))
for i in range(9):
    im = data_augmentation(np.expand_dims(X_train[fig_num], axis=0))
    im = np.array(im).astype(int)
    plt.subplot(3, 3, i+1)
    plt.imshow(im[0])
plt.show()
```

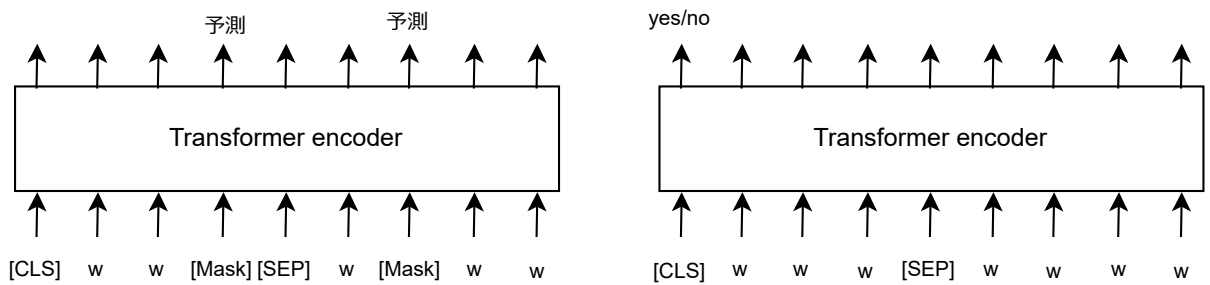


自己教師あり学習による事前学習 (1/3)

- 事前学習(pre-training) + 事後調整(fine tuning) による学習
  - 事前学習
    - 大量にあるデータを利用した教師あり学習問題を設定し、ニューラルネットワークを用いて表現学習を行う
  - 事後調整
    - 少量のタスク固有のデータでニューラルネットワークの全体または一部を調整する

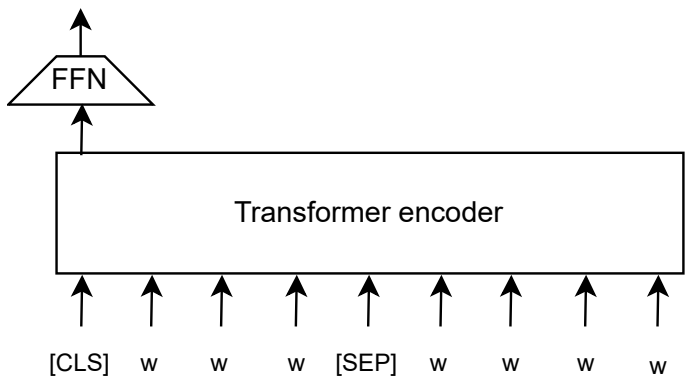
自己教師あり学習による事前学習（2/3）

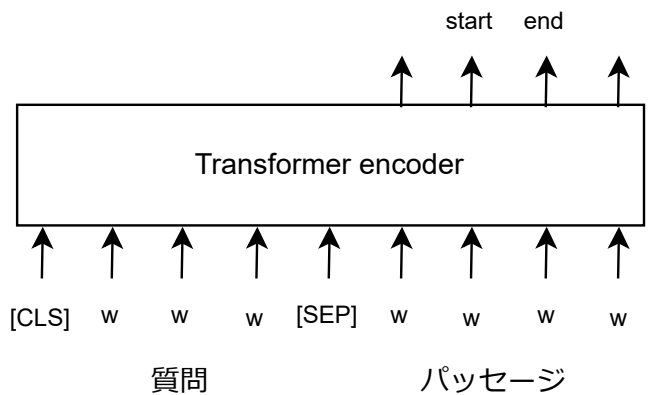
- 自己教師あり学習の事例
  - BERT
    - TransformerのencoderのみをMaskして予測
    - 2種の自己教師あり学習を事前学習として行う
      - 一部の入力単語を
      - 2つの文が連続したものかどうかを判定



自己教師あり学習による事前学習（3/3）

- 事後調整
  - 単純な識別問題など
    - [CLS]の出力を特徴ベクトルとする識別器を学習
  - 系列を扱う場合
    - 入出力をタスクに合わせてパラメータを再調整





14.6 まとめ

- 半教師あり学習
  - 繰り返しに基づくアルゴリズム
    - 初期の誤りに弱い
  - データの散らばりに基づくアルゴリズム
    - 正解の割り当てを最適化問題とみなす
- データ拡張
  - 学習データ／評価データに変換を施す
- 自己教師あり学習による事前学習
  - 画像認識・自然言語処理用の様々な事前学習済みモデルが公開されている　- https://huggingface.co/