目的

機械学習の始まりである単純パーセプトロンにおける教師ありの分類を行うアルゴリズムについて理解を深める.特にIrisデータセットを使った学習の手順から、前処理～評価までの一連の流れについて復習する.

目次

1. 単純パーセプトロンとその学習アルゴリズムについて
2. 単純パーセプトロンの実装
3. Irisデータセットを使った学習の実装

1. 単純パーセプトロンとその学習アルゴリズムについて

単純パーセプトロンについてはパーセプトロンについて (http://qiita.com/nishiy-k/items/1e795f92a99422d4ba7b)を参照

大まかな流れとしては

1. 重みを0か0に非常に近い値で初期化する.
2. 訓練サンプルごとに各入力と重みの総和を計算し、ステップ関数などの活性化関数で出力値を計算する.
3. 出力値との差と学習率ηの積をとり、重みを更新する.

上記の流れを適当な回数繰り返し、訓練サンプルの重みを決定する.

2. パーセプトロンの学習規則の実装

- 詳細は参考文献 (http://book.impress.co.jp/books/1115101122),pp24-27
- パーセプトロンのように個々のトレーニングサンプルを評価した後に重みを更新するのではなく、トレーニングセット全体を用いて勾配を計算する.
- 学習率ηはエポック数n_iterと同じく、学習アルゴリズムの**ハイパーパラメータ**である.分類モデルの性能を最適化するハイパーパラメータの値は自動的に見つけることが可能である.

In [2]:

```
import numpy as np
```

In [23]:

```
class Perceptron(object):
    """
    パーセプトロンの分類器

    パラメータ
    eta : float 学習率
    n_iter : int 訓練データの訓練回数

    w_ : 1次元配列 適合後の重み
    errors_ : list() 各エポック(トレーニング回数)での誤分類数
    """


    def __init__(self, eta=0.01, n_iter=10):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self, X, y):
        """訓練データに適合させる

        パラメータ
        X: {配列のようなデータ構造}, shape = {n_samples, n_features} 訓練データ
        y : 配列のようなデータ構造, shape = {n_samples} 目的変数(正解データ)

        返り値
        self.obj
        """
        self.w_ = np.zeros(1 + X.shape[1]) # 訓練データのサンプル数 + 1（結果）の0ベクトルを作る 0番
        self.cost_ = []

        for i in range(self.n_iter): # トレーニング回数回繰り返す
            # 活性化関数の出力を計算 w^Tx
            output = self.net_input(X)

            # 誤差 yi - φ(Zi) の計算 Zi = w^Tx
            errors = (y - output) # 初期化

            # W1 ~ Wm の重みの更新
            # ΔW = ηSigma( 誤差) * Xji
            self.w_[1:] += self.eta * X.T.dot(errors)

            # W0 の更新(errorsの総和と学習率との積をとる)
            self.w_[0] += self.eta * errors.sum()

            # コスト関数の計算
            cost = (errors**2).sum() / 2.0

            # コストの格納
            self.cost_.append(cost)

        return self

    def net_input(self, X):
        """
        総入力を計算
        """
        return np.dot(X, self.w_[1:] + self.w_[0])

    def activation(self, X):
        """
```

```
        線形活性化関数の出力計算

        """
        return self.net_input(X)

    def predict(self, X):
        """
        1ステップ後のクラスラベルを返す
        """
        # where で指定した条件に当てはまるインデックスを返す
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

3. Irisデータセットを使った学習

> Irisデータセットについて
>
> - 150枚のアヤメの花の計測データを収録したデータセット
> - Setosa, Versicolor, Virginca の3種類の品種に分類.
> - 各サンプルには花びらのcm単位での測定データが記録されている

In [16]:

```
# Irisデータセットを取得する
import pandas as pd
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',engine='
df.tail()
```

Out[16]:

	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

1.前処理

1. 品種ラベルが Setosa, Versicolorの2つのサンプルをそれぞれ50個抽出する.
2. 品種ラベルを 1(Setosa), -1(Versicolor)に置き換える.

2.データの確認

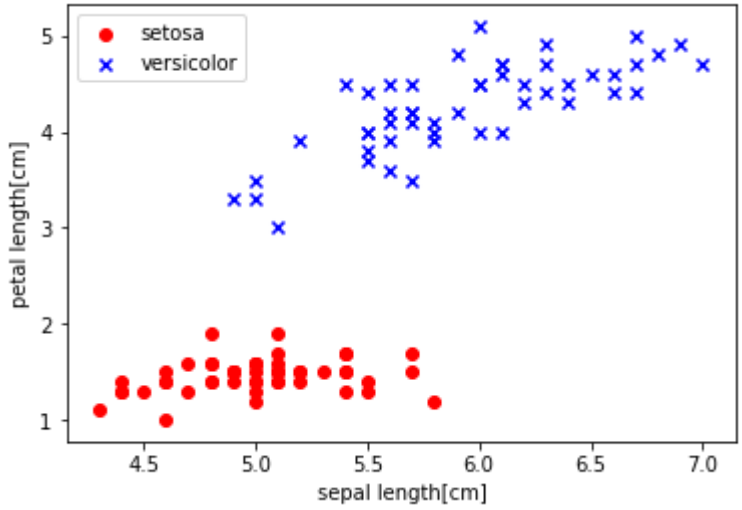
- 前処理を行った段階でsepal lengh(がく片の長さ), petal length(花びらの長さ)との関係を散布図で示すと今回の手法でクラス分類出来ることがわかる.

In [18]:

```
import matplotlib.pyplot as plt
import numpy as np
```

In [19]:

```
# データの整形

# 1-100行目の目的変数(正解ラベル)の抽出
y = df.iloc[0:100, 4].values
# Iris-setosa を -1, Iris-virginica を 1 に変換
y = np.where(y == 'Iris-setosa', -1, 1)

# 1-100行目の1,3行目(素性)の抽出
X = df.iloc[0:100, [0,2]].values

plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100,0], X[50:100,1], color='blue', marker='x', label='versicolor')

plt.xlabel('sepal length[cm]')
plt.ylabel('petal length[cm]')

plt.legend(loc = 'upper left')
plt.show()
```

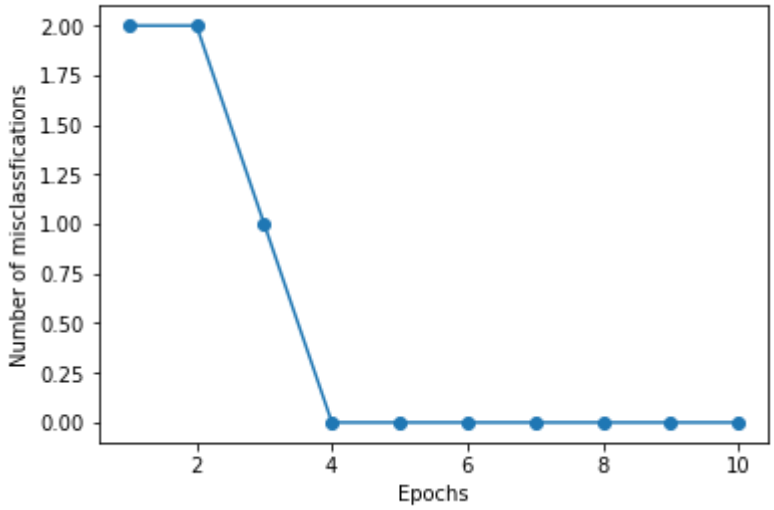


3.学習の収束度合いを確認

- 前述したパーセプトロンの学習アルゴリズムを使って、学習を行う
- 正解データとの誤差とエポック数との関係をグラフで表すと4回目以降は収束していることがわかった.

In [21]:

```
# パーセプトロンのオブジェクトの生成(インスタンス化)
ppn = Perceptron(eta=0.1, n_iter=10)

# トレーニングデータのモデル適合
ppn.fit(X, y)

# エポック(トレーニング回数)とご分類誤差の関係折れ線グラフをプロット
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')

# 軸ラベルの設定
plt.xlabel('Epochs')
plt.ylabel('Number of misclassfications')
plt.show()
```



4.決定境界の出力

- 決定境界を出力させることで、学習ができていることを確認.ほかのサンプルに対して分類が出来るようになった.

In [22]:

```
# プログラムは省略
# p.30を参照
```

まとめ

パーセプトロンは各サンプル**Xi**に対して重みWiを掛けた総和を活性化関数(単位ステップ関数等)に通すことで二値出力を得る。得られた出力と正解との差を重みに加える事によって重みを修正する。

非常に簡単な学習ではあるがIrisデータセットの2つの品種クラスを分類することができた。しかし、パーセプトロンの最大の問題点は**収束性**にある。パーセプトロンは2つのクラスを線形超平面(特徴量の数n の次元をもつ平面)で分割できない場合はエポックの最大数を設定しない限りは計算が終わらない.

次回

重みの更新方法を変更し、コスト関数の最小化に関する手法を取り入れたADALINEについて学ぶ