assign6_2_svclassification

May 25, 2019

```
In [6]: import numpy as np
        import matplotlib

        import matplotlib.pyplot as plt
        %matplotlib inline

        np.random.seed(1)
```

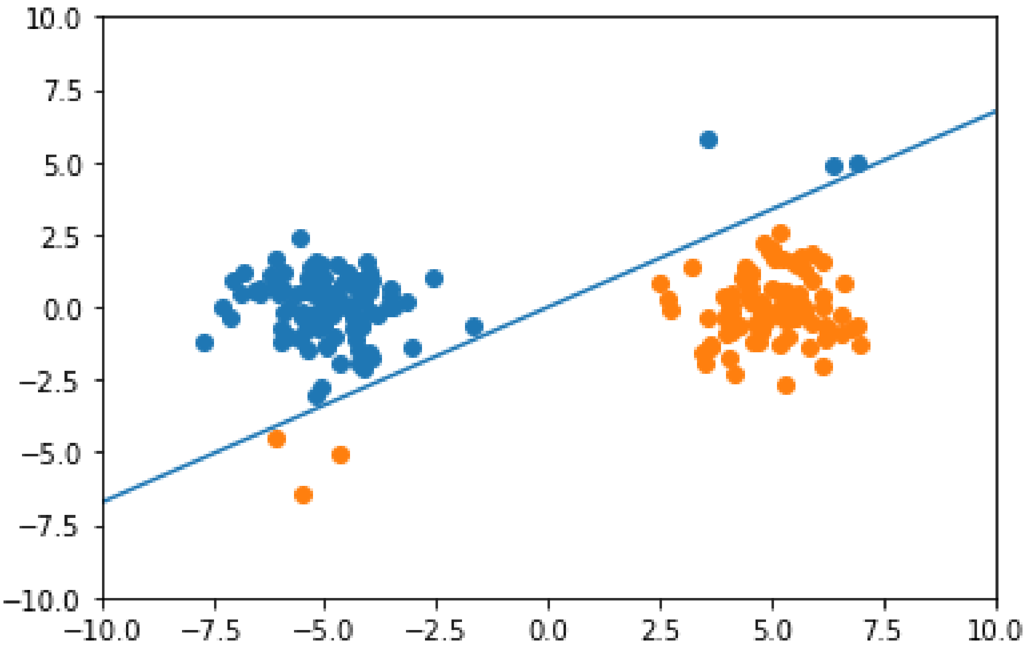
```
In [7]: def generate_data(sample_size):
            """Generate training data.

            Since
            f(x) = w^{T}x + b
            can be written as
            f(x) = (w^{T}, b)(x^{T}, 1)^{T},
            for the sake of simpler implementation of SVM,
            we return (x^{T}, 1)^{T} instead of x

            :param sample_size: number of data points in the sample
            :return: a tuple of data point and label
            """

            x = np.random.normal(size=(sample_size, 3))
            x[:, 2] = 1.
            x[:sample_size // 2, 0] -= 5.
            x[sample_size // 2:, 0] += 5.
            y = np.concatenate([np.ones(sample_size // 2, dtype=np.int64),
                                -np.ones(sample_size // 2, dtype=np.int64)])
            x[:3, 1] -= 5.
            y[:3] = -1
            x[-3:, 1] += 5.
            y[-3:] = 1
            return x, y
```

```
In [11]: x, y = generate_data(200)
```

```
In [12]: def svm(x, y, l, lr):
             """Linear SVM implementation using gradient descent algorithm.

             f_w(x) = w^{T} (x^{T}, 1)^{T}

             :param x: data points
             :param y: label
             :param l: regularization parameter
```

```
    :param lr: learning rate
    :return: three-dimensional vector w
    ”””

    w = np.zeros(3)
    prev_w = w.copy()
    for i in range(10 ** 4):
        indices = (x.dot(w)*y <= 1) # マージンが 1 以下の場合 1 を返す
        dw = -(indices * y).dot(x) + l * w
        w -= lr * dw

        if np.linalg.norm(w - prev_w) < 1e-3:
            break
        prev_w = w.copy()

    return w, indices

def visualize(x, y, w):
    plt.clf()
    plt.xlim(-10, 10)
    plt.ylim(-10, 10)
    plt.scatter(x[y == 1, 0], x[y == 1, 1])
    plt.scatter(x[y == -1, 0], x[y == -1, 1])
    plt.plot([-10, 10], -(w[2] + np.array([-10, 10]) * w[0]) / w[1])
    plt.savefig(’lecture6-h2.png’)
```

```
In [13]: w, indices = svm(x, y, l=.1, lr=1.)
         visualize(x, y, w)
```


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0