机器学习实验一

数据处理

1. 将数据集随机打乱，按 9 : 1 将数据集分为训练集和测试集。
2. 对于感知机和支持向量机采用 ±1 标签；对于逻辑回归采用 0-1 标签。

关键代码：

```
# +-1 标签
if label_class == 0:
    # Iris-setosa 标签为 1， Iris-versicolor 标签为 -1
    y = 1 if one_iris_data[-1] == "Iris-setosa" else -1
# 0-1 标签
else:
    # Iris-setosa 标签为 1， Iris-versicolor 标签为 0
    y = 1 if one_iris_data[-1] == "Iris-setosa" else 0
```

感知机

- 根据误分类点跟新参数：

关键代码：

```
    for _ in range(max_iteration):
        for x, y in zip(train_X, train_Y):
            if y * (self.w.dot(x.T) + self.b) <= 0:
                self.w = self.w + lr * y * x
                self.b = self.b + lr * y
                break
        else:
            break
```

实验参数：

```
w: [-0.18761891  0.20408342 -0.26249487  0.46606935]
b: 0.7189643616730146
```

实验结果：

训练集正确率：100.00%

测试集正确率：100.00%

SVM

- 检查 α 是否符合 KKT 条件：

```
def _check_KKT(self, i):
    """
    判断 alpha_i 是否符合 KKT 条件
    :param i:
    :return:
    """
    alpha_i = self.alphas[i]
    x_i = self.train_X[i]
    y_i = self.train_Y[i]
    g_x_i = self._get_g(x_i)
    if alpha_i == 0:
        return y_i * g_x_i >= 1
    elif 0 < alpha_i < self.C:
        return y_i * g_x_i == 1
    else:
        return y_i * g_x_i <= 1
```

- 选择要更新的参数：

```python
def _choose_alpha(self):
    """
    选择 alpha_i 和 alpha_j
    alpha_i 是违背 KTT 条件的变量，外循环
    alpha_j 是与 alpha_i 对应偏差最大的变量，内循环
    :return: i 和 j
    """
    support_vector_index = []
    un_support_vector_index = []
    for index in range(self.data_size):
        if 0 <= self.alphas[index] <= self.C:
            support_vector_index.append(index)
        else:
            un_support_vector_index.append(index)
    # 先看可能是支持向量的 alpha_i
    index_i = support_vector_index + un_support_vector_index
    for i in index_i:
        if self._check_KKT(i):
            continue
        else:
            E1 = self._E[i]
            # 选择变化最大 的 alpha_j
            if E1 >= 0:
                j = min(range(self.data_size), key=lambda x: self._E[x])
            else:
                j = max(range(self.data_size), key=lambda x: self._E[x])
            return i, j
```

- SMO 是支持向量机优化的关键算法：

```python
# 1. 选取 alpha1 和 alpha2
if (alpha := self._choose_alpha()) is not None:
    i, j = alpha
else:
    # 所有点都满足 KKT 条件，优化结束
    break
# 计算相关值
alpha1, alpha2 = self.alphas[i], self.alphas[j]
x1, x2 = self.train_X[i], self.train_X[j]
y1, y2 = self.train_Y[i], self.train_Y[j]
```

```python
# 2. 更新 alpha2
E1, E2 = self._E[i], self._E[j]
eta = self._kernel_function(self.train_X[i, :], self.train_X[i, :])\
      + self._kernel_function(self.train_X[j, :], self.train_X[j, :])\
      - 2 * self._kernel_function(self.train_X[i, :], self.train_X[j, :])
alpha2_new_unc = alpha2 + y2 * (E1 - E2) / eta
```

```python
        # 3. 剪枝 alpha2
        if y1 == y2:
            L = max(0, alpha1 + alpha2 - self.C)
            H = min(self.C, alpha1 + alpha2)
        else:
            L = max(0, alpha2 - alpha1)
            H = min(self.C, self.C + alpha2 - alpha1)
        if alpha2_new_unc > H:
            alpha2_new = H
        elif alpha2_new_unc < L:
            alpha2_new = L
        else:
            alpha2_new = alpha2_new_unc
```

```python
# 4. 更新 alpha1
alpha1_new = alpha1 + y1 * y2 * (alpha2 - alpha2_new)

# 5. 更新 b 和 E 值
b1_new = - E1 - y1 * self._kernel_function(x1, x1) * (alpha1_new - alpha1) \
         - y2 * self._kernel_function(x2, x1) * (alpha2_new - alpha2) + self.b
b2_new = - E2 - y1 * self._kernel_function(x1, x2) * (alpha1_new - alpha1) \
         - y2 * self._kernel_function(x2, x2) * (alpha2_new - alpha2) + self.b
if 0 < alpha1_new < self.C:
    b_new = b1_new
elif 0 < alpha2_new < self.C:
    b_new = b2_new
else:
    b_new = (b1_new + b2_new) / 2
```

实验参数:

```
w: [  1.54009453  16.0697418  -31.79116625 -13.29147242]
b: [52.06452886]
```

实验结果:

训练集正确率：1.0
测试集正确率：1.0

逻辑回归

- 采用梯度下降法跟新参数

关键代码:

```python
for iteration in range(max_iteration):
    result = sigmoid(np.dot(self.w, X.T) + self.b)
    error = result - Y
    grad = np.dot(X.T, error)
    self.w = self.w - lr * grad
    self.b = self.b - lr * np.sum(error)
```

实验参数:

```
w: [ 0.76055262  3.13564822 -4.86397648 -1.94418025]
b: 1.0006079233735141
```

实验结果:

```
训练集正确率：1.0
测试集正确率：1.0
```

实验总结

本次实验，完成了感知机、支持向量机和逻辑回归模型的实现，对这些模型有了更加深刻的理解，并能够在实际问题中应用这些模型。