SVM

本文公式参考《机器学习理论导引》一书，故采用和书本中一样的编号。

SVM的最优化问题

$$\begin{gathered}\min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t.\ \ y_i(w^T x_i + b) \geq 1\ (i \in [m])\end{gathered} \tag{1.59}$$

式(1.59)的拉格朗日函数为

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^T x_i + b)) \tag{1.60}$$

主问题1.59的对偶问题

$$\begin{gathered}\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{m} \alpha_i \\ s.t.\ \ \sum_{i=1}^{m} \alpha_i y_i = 0, \\ \alpha_i \geq 0\ (i \in [m])\end{gathered} \tag{1.63}$$

由笔记“2-拉格朗日对偶性”可知，x^*, α^*是原始问题和对偶问题解的充要条件为满足KKT条件。

$$\begin{cases} \sum_{i=1}^{m} \alpha_i y_i x_i = w\ \ 常定方程式 \\ \sum_{i=1}^{m} \alpha_i y_i = 0\ \ 常定方程式 \\ \alpha_i \geq 0\ \ 对偶可行性 \\ y_i(w^T x_i + b) - 1 \geq 0\ \ 约束条件 \\ \alpha_i(y_i(w^T x_i + b) - 1) = 0\ \ 互补松弛性 \end{cases}$$

对原始空间中线性不可分的问题，可将样本从**原始空间映射到一个高维特征空间**，$\phi(x)$。则式1.59变成：

$$\begin{gathered}\min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t.\ \ y_i(w^T \phi(x_i) + b) \geq 1\ (i \in [m])\end{gathered} \tag{1.65}$$

对偶问题变为：

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) - \sum_{i=1}^{m} \alpha_i$$

$$
\begin{gathered}
s.t.\ \ \sum_{i=1}^{m}\alpha_i y_i = 0, \\
\alpha_i \geq 0\ (i \in [m])
\end{gathered} \tag{1.66}
$$

由于$\phi(x_i)^T\phi(x_j)$计算困难，所以引入核函数$k(x_i, x_j)$

引入**软间隔**，允许某些样本不满足约束 $y_i(w^T\phi(x_i)+b) \geq 1$ (1.69)

在最大化间隔时，使不满足约束的样本尽可能少。则优化目标可以写为：

$$
\min_{w,b}\frac{1}{2}\|w\|^2 + \beta\sum_{i=1}^{m}\ell_{0/1}(y_i(w^T\phi(x_i)+b)-1) \tag{1.70}
$$

其中，$\ell_{0/1}(x) = \mathbb{I}(x<0)$，由于$\ell_{0/1}(x)$非凸不连续，用hinge损失函数代替$\ell_{hinge}(x) = max(0, 1-x)$

则1.70改写为：

$$
\min_{w,b}\frac{1}{2}\|w\|^2 + \beta\sum_{i=1}^{m}max(0, 1-y_i(w^T\phi(x_i)+b)) \tag{1.73}
$$

引入松弛变量$\xi_i \geq 0$，可将式1.73重写为

$$
\begin{gathered}
\min_{w,b,\xi}\frac{1}{2}\|w\|^2 + \beta\sum_{i=1}^{m}\xi_i \\
s.t.\ y_i(w^T\phi(x_i)+b) \geq 1-\xi_i \\
\xi_i \geq 0\ (i \in [m])
\end{gathered} \tag{1.74}
$$

其对偶问题：

$$
\begin{gathered}
\min_{\alpha}\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{m}\alpha_i \\
s.t.\ \ \sum_{i=1}^{m}\alpha_i y_i = 0, \\
0 \leq \alpha_i \leq \beta\ (i \in [m])
\end{gathered} \tag{1.75}
$$

sklearn中的SVM使用demo：

```
# -*- coding: utf-8 -*-
from sklearn import svm
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

iris_df = pd.read_csv('mooc-机器学习作业/iris/iris.data', header=None)
iris_df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width',
'label']
iris_array = np.array(iris_df)

# 划分训练集和测试集
X_train, X_test, Y_train, Y_test = train_test_split(iris_array[:, 0:4],
iris_array[:, 4], test_size=0.2, random_state=2)

svm_classifier = svm.SVC(C=1.0, kernel='rbf', decision_function_shape='ovr',
gamma=0.01)
svm_classifier.fit(X_train, Y_train.reshape(Y_train.size))
print("训练集:", svm_classifier.score(X_train, Y_train))
print("测试集:", svm_classifier.score(X_test, Y_test))
```

引申:

为什么将SVM原始问题变换成对偶问题求解呢?

1. 对偶问题往往更容易计算。求解原始问题，我们得到最优w，而并不清楚a_i。当样本维度d很大时，计算w很复杂；求解对偶问题是，我们得到α_i（除了少数点-支持向量，其他的α_i均为零），当支持向量很少时，计算很高效。

2. 引入核函数（核技巧），从而推广到非线性问题。因为对偶问题是对x内积的优化，这样可以很方便的引入核函数技巧。当x_i不是支持向量时，α_i则为0，所以实际上是支持向量的内积。

支持向量机模型的使用技巧? 假定n为特征数，m为训练样本数。下面的普遍使用准则源于吴恩达机器学习个人笔记完整版 P196

1. 如果相较于m而言，n要大很多，即训练集数据量不够支持我们训练一个复杂的非线性模型，我们选用逻辑回归模型或者不带核函数的支持向量机。

2. 如果n较小，而且m大小中等，例如n在1-1000之间，而m在10-10000之间，使用高斯核函数的支持向量机。

3. 如果n较小，而m较大，例如n在1-1000之间，而m大于50000，则使用支持向量机会非常慢，解决方案是创造、增加更多的特征，然后使用逻辑回归或者不带核函数的支持向量机。

值得一提的是，神经网络在以上三种情况下都可能会有较好的表现，但是训练神经网络可能非常慢，选择支持向量机的原因主要在于它的代价函数是凸函数，不存在局部最小值。