**SVM理解**

**2019-1-8**

1. **由来**

支持向量机，英文名称是Support Vector Machine，故又称为SVM，是一种二分类的分类模型，基本模型定义是在特征空间上获得间隔最大的线性分类器。通过核技巧还可以变成非线性分类器。

SVM是线性分类器，那么什么是线性分类器？

给定一个数据集，数据集中的样本点分别属于两个不同的类，现在需要一个线性分类器将他们分为两类。我们用x表示样本点，y表示对应样本类别（y取+1或-1），线性分类器的学习目标是在n维数据空间中找到一个超平面，将这个数据集中根据样本类别完全分为两类，得到这个超平面的方程为：



SVM目标即是获得间隔最大的线性分类器，可以形式化一个求解凸二次规划问题，最后可以找到一组最优解 ，从而获得间隔最大的超平面：



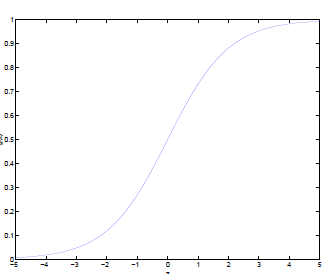
得到相应的分类决策函数：



对于类别y的取值为什么是+1或-1，则起源于Logistic回归。Logistic回归目的是通过样本的特征学习出一个0/1的分类模型，模型的输入是样本特征的线性组合，使用sigmoid函数将输入映射到（0，1）上。

Logistic函数模型为：



可以看到，只要，类别就判为1；，类别就判为0。

下面将Logistic回归做个变形，将 中的 替换为，将替换为，则，再将分类标签0/1变为-1/+1，则线性分类函数公式与Logistic回归的形式化表示的 就没有区别。由此，便通过逻辑回归函数推导到了线性分类函数。

1. **SVM推导**

**A、**

首先，我们的目标是：给定线性可分的训练数据集，通过间隔最大化或等价的求解想应的凸二次规划问题学习得到分离超平面：



及分类决策函数：



即**我们要通过学习求解出最优的参数。**

**B、**

至于这个最优的参数如何求解，我们需要首先引入**函数间隔**。一般来说，一个点距离分离超平面的远近可以表示分类预测的确信程度，这个距离就是函数间隔。

对于给定的训练数据集T及分类超平面，定义超平面对于样本点的函数间隔为：



分类超平面关于数据集T的函数间隔即为T中所有样本点的函数间隔最小值：



函数间隔会随成比例改变而改变，为解决这个问题，引入**几何间隔** ，定义超平面对于样本点的几何间隔为：



分类超平面关于数据集T的几何间隔与函数间隔同理：



两种间隔之间的关系为：

****

由于几何间隔不会随改变而改变，所以使用几何间隔作为参考标准。

**C、**

因此，SVM学习的基本想法就变成了**在正确分类训练数据集的基础上，使得数据集对于分类超平面的几何间隔最大**。

如何求得几何间隔最大的分离超平面可以表示为下面的最优化问题：





根据两种间隔的关系，上式可以进一步写成：





因为函数间隔取值不影响最优化问题的解，所以取





又因为最大化与最小化等价，上述问题进一步变为：





这是一个凸二次规划问题，将其作为原始最优化问题可以应用拉格朗日对偶性，通过求解对偶问题来得到原始问题的最优解。

**D、**

构建拉格朗日函数：



原始问题的对偶问题是极大极小问题：



1）先求极小问题

用分别对求偏导并令其等于0，得到下式：





带入拉格朗日函数得：（注意公式中的和均为标量，转制后仍为其本身）



所以，



2）求对的极大问题，即为：





进一步便得到与原问题等价的对偶问题：





然后利用线性可分的数据集对对偶问题进行求解最优 ，在用最优的带入下式求出最优的，便得到原问题的最优解。





最后便可以得到对偶形式的最优分类超平面：



及分类决策函数：



至于如何求解出最优的，需要使用后面的SMO算法来求解。

E、

对于非线性问题，使用核函数将样本点的特征映射到高维空间中去，从而在高维空间了利用线性分类学习方法从训练数据中学习分类模型。最优化问题变为：





F、

当数据存在奇异点的时候，对每个样本点引进一个松弛变量 ，这样以来约束条件变为



目标函数由原来的，变为



其中 ，被称为惩罚参数。因此，整个对偶问题变为：





1. **SMO算法**
2. **算法实现**
3. **线性分类**
4. **简化SMO算法**
5. **完整SMO算法**
6. **非线性分类**
7. **现成的库效果对比**
8. **参考资料**

[1] 《统计学习方法》李航

[2] 《机器学习实战》Peter Harrington

[3] Python3《机器学习实战》学习笔记（八）：支持向量机原理篇之手撕线性SVM： <https://blog.csdn.net/c406495762/article/details/78072313>

[4] 支持向量机通俗导论（理解SVM的三层境界）：

<https://blog.csdn.net/v_JULY_v/article/details/7624837>