# SVC、NuSVC与LinearSVC

## 1.SVM简介

　　SVM方法建立在统计学VC维和结构风险最小化原则上，既可以用于分类（二/多分类）、也可用于回归和异常值检测。SVM具有良好的鲁棒性，对未知数据拥有很强的泛化能力，特别是在数据量较少的情况下，相较其他传统机器学习算法具有更优的性能。

使用SVM作为模型时，通常采用如下流程：

（1）对样本数据进行归一化；

（2）应用核函数对样本进行映射（最常采用和核函数是RBF和Linear，在样本线性可分时，Linear效果要比RBF好）；

（3）用cross-validation和grid-search对超参数进行优选；

（4）用最优参数训练得到模型；

（5）测试。

　　sklearn中支持向量分类主要有三种方法：SVC、NuSVC、LinearSVC，扩展为三个支持向量回归方法：SVR、NuSVR、LinearSVR。

　　SVC和NuSVC方法基本一致，唯一区别就是损失函数的度量方式不同（NuSVC中的nu参数和SVC中的C参数）；LinearSVC是实现线性核函数的支持向量分类，没有kernel参数，也缺少一些方法的属性，如support\_等。

## 2. 参数

### SVC

class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=-1, decision\_function\_shape='ovr', random\_state=None)

　　C: 惩罚系数，用来控制损失函数的惩罚系数，类似于LR中的正则化系数。C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样会出现训练集测试时准确率很高，但泛化能力弱，容易导致过拟合。 C值小，对误分类的惩罚减小，容错能力增强，泛化能力较强，但也可能欠拟合。

　　kernel: 算法中采用的和函数类型，核函数是用来将非线性问题转化为线性问题的一种方法。参数选择有RBF, Linear, Poly, Sigmoid，precomputed或者自定义一个核函数, 默认的是"RBF"，即径向基核，也就是高斯核函数；而Linear指的是线性核函数，Poly指的是多项式核，Sigmoid指的是双曲正切函数tanh核；。

　　degree: 当指定kernel为'poly'时，表示选择的多项式的最高次数，默认为三次多项式；若指定kernel不是'poly'，则忽略，即该参数只对'poly'有用。（多项式核函数是将低维的输入空间映射到高维的特征空间）

　　gamma: 核函数系数，该参数是rbf，poly和sigmoid的内核系数；默认是'auto'，那么将会使用特征位数的倒数，即1 / n\_features。（即核函数的带宽，超圆的半径）。gamma越大，σ越小，使得高斯分布又高又瘦，造成模型只能作用于支持向量附近，可能导致过拟合；反之，gamma越小，σ越大，高斯分布会过于平滑，在训练集上分类效果不佳，可能导致欠拟合。

　　coef0: 核函数常数值(y=kx+b中的b值)，只有‘poly’和‘sigmoid’核函数有，默认值是0。

　　shrinking : 是否进行启发式。如果能预知哪些变量对应着支持向量，则只要在这些样本上训练就够了，其他样本可不予考虑，这不影响训练结果，但降低了问题的规模并有助于迅速求解。进一步，如果能预知哪些变量在边界上(即a=C)，则这些变量可保持不动，只对其他变量进行优化，从而使问题的规模更小，训练时间大大降低。这就是Shrinking技术。 Shrinking技术基于这样一个事实：支持向量只占训练样本的少部分，并且大多数支持向量的拉格朗日乘子等于C。

　　probability: 是否使用概率估计，默认是False。必须在 fit( ) 方法前使用，该方法的使用会降低运算速度。

　　tol: 残差收敛条件，默认是0.0001，即容忍1000分类里出现一个错误，与LR中的一致；误差项达到指定值时则停止训练。

　　cache\_size: 缓冲大小，用来限制计算量大小，默认是200M。

　　class\_weight : {dict, ‘balanced’}，字典类型或者'balance'字符串。权重设置，正类和反类的样本数量是不一样的，这里就会出现类别不平衡问题，该参数就是指每个类所占据的权重，默认为1，即默认正类样本数量和反类一样多，也可以用一个字典dict指定每个类的权值，或者选择默认的参数balanced，指按照每个类中样本数量的比例自动分配权值。如果不设置，则默认所有类权重值相同，以字典形式传入。 将类i 的参数C设置为SVC的class\_weight[i]\*C。如果没有给出，所有类的weight 为1。'balanced'模式使用y 值自动调整权重，调整方式是与输入数据中类频率成反比。如n\_samples / (n\_classes \* np.bincount(y))。（给每个类别分别设置不同的惩罚参数C，如果没有给，则会给所有类别都给C=1，即前面参数指出的参数C。如果给定参数'balance'，则使用y的值自动调整与输入数据中的类频率成反比的权重。）

　　verbose : 是否启用详细输出。在训练数据完成之后，会把训练的详细信息全部输出打印出来，可以看到训练了多少步，训练的目标值是多少；但是在多线程环境下，由于多个线程会导致线程变量通信有困难，因此verbose选项的值就是出错，所以多线程下不要使用该参数。

　　max\_iter: 最大迭代次数，默认是-1，即没有限制。这个是硬限制，它的优先级要高于tol参数，不论训练的标准和精度达到要求没有，都要停止训练。

　　decision\_function\_shape ： 原始的SVM只适用于二分类问题，如果要将其扩展到多类分类，就要采取一定的融合策略，这里提供了三种选择。‘ovo’ 一对一，为one v one，即将类别两两之间进行划分，用二分类的方法模拟多分类的结果，决策所使用的返回的是（样本数，类别数\*(类别数-1)/2）； ‘ovr’ 一对多，为one v rest，即一个类别与其他类别进行划分，返回的是(样本数，类别数)，或者None，就是不采用任何融合策略。默认是ovr，因为此种效果要比oro略好一点。

random\_state: 在使用SVM训练数据时，要先将训练数据打乱顺序，用来提高分类精度，这里就用到了伪随机序列。如果该参数给定的是一个整数，则该整数就是伪随机序列的种子值；如果给定的就是一个随机实例，则采用给定的随机实例来进行打乱处理；如果啥都没给，则采用默认的 np.random实例来处理。

### NuSVC

class sklearn.svm.NuSVC(nu=0.5, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=-1, decision\_function\_shape='ovr', random\_state=None)

　　nu： 训练误差部分的上限和支持向量部分的下限，取值在（0，1）之间，默认是0.5

### LinearSVC

class sklearn.svm.LinearSVC(penalty='l2', loss='squared\_hinge', dual=True, tol=0.0001, C=1.0, multi\_class='ovr', fit\_intercept=True, intercept\_scaling=1, class\_weight=None, verbose=0, random\_state=None, max\_iter=1000)

　　penalty: 正则化参数，L1和L2两种参数可选，仅LinearSVC有。

　　loss: 损失函数，有‘hinge’和‘squared\_hinge’两种可选，前者又称L1损失，后者称为L2损失，默认是是’squared\_hinge’，其中hinge是SVM的标准损失，squared\_hinge是hinge的平方

　　dual: 是否转化为对偶问题求解，默认是True。

　　tol: 残差收敛条件，默认是0.0001，与LR中的一致。

　　C: 惩罚系数，用来控制损失函数的惩罚系数，类似于LR中的正则化系数。

　　multi\_class: 负责多分类问题中分类策略制定，有‘ovr’和‘crammer\_singer’ 两种参数值可选，默认值是’ovr’，'ovr'的分类原则是将待分类中的某一类当作正类，其他全部归为负类，通过这样求取得到每个类别作为正类时的正确率，取正确率最高的那个类别为正类；‘crammer\_singer’ 是直接针对目标函数设置多个参数值，最后进行优化，得到不同类别的参数值大小

　　fit\_intercept: 是否计算截距，与LR模型中的意思一致。

　　class\_weight: 与其他模型中参数含义一样，也是用来处理不平衡样本数据的，可以直接以字典的形式指定不同类别的权重，也可以使用balanced参数值。

　　verbose: 是否冗余，默认是False。

　　random\_state: 随机种子。

　　max\_iter: 最大迭代次数，默认是1000。

## 3. 属性(Attributes)

　　support\_: 以数组的形式返回支持向量的索引，即在所有的训练样本中，哪些样本成为了支持向量。

　　support\_vectors\_: 返回支持向量，汇总了当前模型所有的支持向量。

　　n\_support\_: 比如SVC将数据集分成了4类，该属性表示了每一类的支持向量的个数。

　　dual\_coef\_: 对偶系数，即支持向量在决策函数中的系数，在多分类问题中，这个会有所不同。

　　coef\_: 每个特征系数（重要性），只有核函数是Linear的时候可用。

　　intercept\_: 决策函数中的常数项（借据值），和coef\_共同构成决策函数的参数值。

## 4. 方法(Method)

　　decision\_function(X): 获取数据集中样本X到分离超平面的距离。

　　fit(X, y): 在数据集(X,y)上拟合SVM模型。

　　get\_params([deep]): 获取模型的参数。

　　predict(X): 预测数据值X的标签。

　　score(X,y): 返回给定测试集和对应标签的平均准确率。

## 5. 核函数的使用

**RBF核**：高斯核函数就是在属性空间中找到一些点，这些点可以是也可以不是样本点，把这些点当做base，以这些base为圆心向外扩展，扩展半径即为带宽，即可划分数据。换句话说，在属性空间中找到一些超圆，用这些超圆来判定正反类。

**线性核和多项式核**：这两种核的作用也是首先在属性空间中找到一些点，把这些点当做base，核函数的作用就是找与该点距离和角度满足某种关系的样本点。当样本点与该点的夹角近乎垂直时，两个样本的欧式长度必须非常长才能保证满足线性核函数大于0；而当样本点与base点的方向相同时，长度就不必很长；而当方向相反时，核函数值就是负的，被判为反类。即，它在空间上划分出一个梭形，按照梭形来进行正反类划分。

**Sigmoid核**：同样地是定义一些base，核函数就是将线性核函数经过一个tanh函数进行处理，把值域限制在了-1到1上。

　　总之，都是在定义距离，大于该距离，判为正，小于该距离，判为负。至于选择哪一种核函数，要根据具体的样本分布情况来确定。

一般有如下指导规则：

（1）如果Feature的数量很大，甚至和样本数量差不多时，往往线性可分，这时选用LR或者线性核Linear；

（2）如果Feature的数量很小，样本数量正常，不算多也不算少，这时选用RBF核；

（3）如果Feature的数量很小，而样本的数量很大，这时手动添加一些Feature，使得线性可分，然后选用LR或者线性核Linear；

（4）多项式核一般很少使用，效率不高，结果也不优于RBF；

（5）Linear核参数少，速度快；RBF核参数多，分类结果非常依赖于参数，需要交叉验证或网格搜索最佳参数，比较耗时；

应用最广的应该就是RBF核，无论是小样本还是大样本，高维还是低维等情况，RBF核函数均适用。