CSDN
不 止 于 代 码

2.12 sklearn中的SVM实现

CSDN学院

2017年11月

AI 101

Scikit learn 中的SVM实现

- sklearn.svm模块提供支持向量机算法，可用于分类、回归和异常值检测。
- 分类实现有三种方式：
  - LinearSVC基于liblinear实现线性SVM ，比基于libsvm实现的线性SVC / NuSVC更快，同时可采用更多正则选择（L1/L2）和损失函数选择
    - L1正则可以得到特征系数稀疏的效果
  - SVC和NuSVC类似，都是基于libsvm实现的C-SVM，二者在参数方面有细微不同（ NuSVC 有参数*nu*控制训练误差的上限和支持向量的下限）
- 回归和分类类似
- SVM还支持非监督的异常值检测： OneClassSVM

Scikit learn提供的SVC实现

- *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：C-SVC的惩罚参数，默认值是1.0。C越大，即对误分类的惩罚增大，趋向于对训练集全分对
  - probability ：是否采用概率估计，默认为False
  - cache_size ：核函数cache缓存大小，默认为200。核缓存的大小对较大问题求解的运行时间有非常强的影响。如果有足够内存，建议将cache_size设置为更高的值
  - decision_function_shape ：多类分类任务中用到，可为 'ovo', 'ovr' or None, default=None
- 需要调节模型复杂度参数有：C、kernel、degree、gamma、coef0

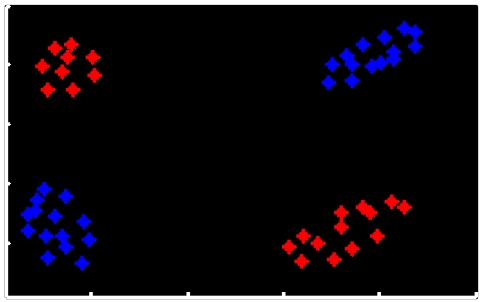
核函数kernel

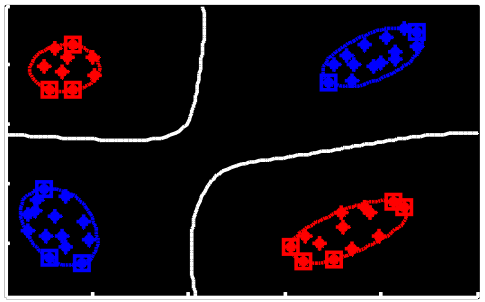
- 核函数的参数有degree(M)、gamma(γ)、coef0(θ)，核函数可以有以下几种选择：
  - 线性核：$k(\mathbf{x},\mathbf{x}')=\mathbf{x}^T\mathbf{x}'$
  - 多项式核，缺省为3阶多项式 $k(\mathbf{x},\mathbf{x}')=\left(\gamma\mathbf{x}^T\mathbf{x}'+\theta\right)^M$
  - 径向基函数（RBF）：$k(\mathbf{x},\mathbf{x}')=\exp\left(-\gamma\left(\mathbf{x}-\mathbf{x}'\right)^2\right)$
  - sigmoid函数：$k(\mathbf{x},\mathbf{x}')=\tanh\left(\gamma\mathbf{x}^T\mathbf{x}'+\theta\right)$
  - 也可以用户自定义核
- 在初始化时通过参数kernel指定用什么核

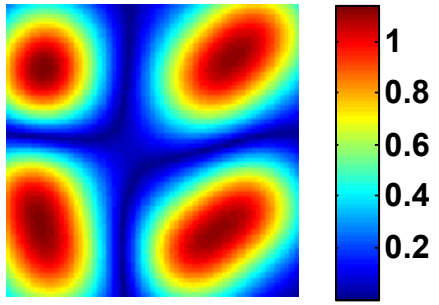
RBF核的参数

- 使用径向基函数（RBF）核训练SVM时，需要考虑正则参数*C*和核函数宽度参数gamma。
  - 参数*C*会被所有SVM核用到，用来在样本误分类和决策平面的复杂性之间做出权衡。*C*越小，决策边界越平滑； *C*越大，要求更多样本倍分正确。
  - gamma定义单个训练样本能影响多大范围。gamma越大，对应RBF的标准差σ越小，影响的范围更小； gamma越小，决策边界越平滑。
- 这两个值的选择会极大影响SVM的性能。建议使用sklearn.grid_search.GridSearchCV来在C和gamma广阔的指数空间里进行选择，得到最合适的值。

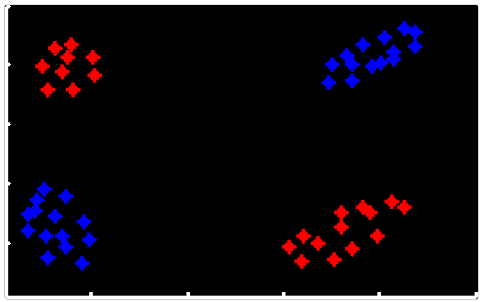
RBF核——核参数正好

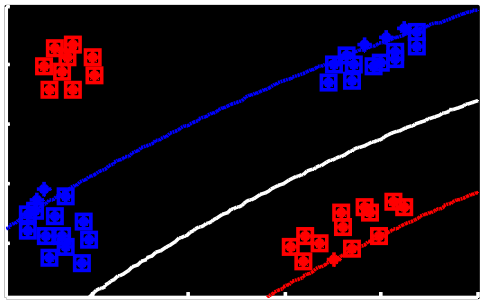


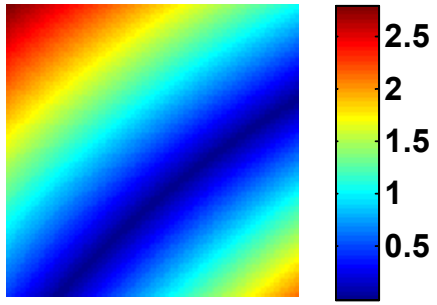


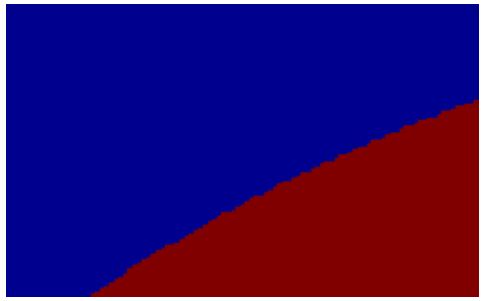


RBF核——欠拟合

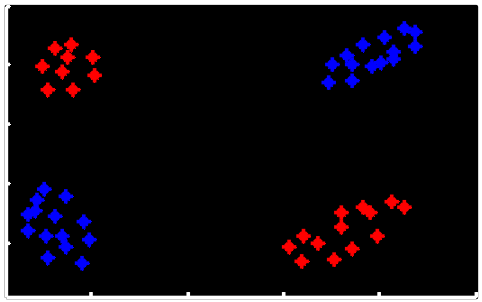


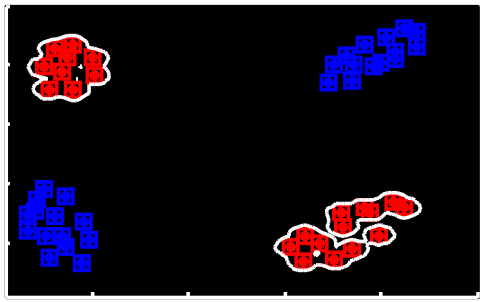


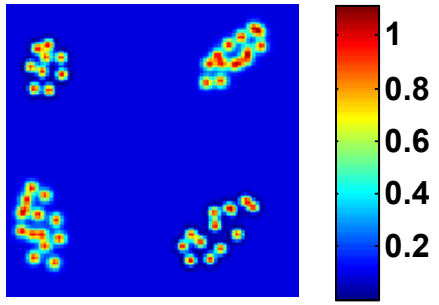


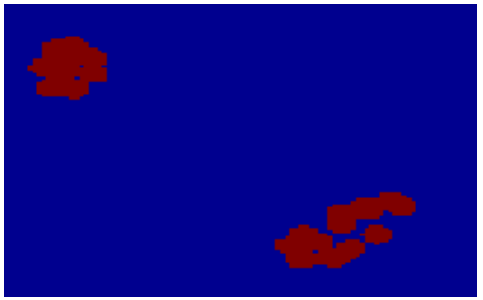


RBF核—过拟合









多类别分类

- SVC和NuSVC使用“一对多”方法（One vs. One, OvO）实现多类别分类
  - 假设进行C个类别的分类任务，构造$C * (C - 1) / 2$个分类器：每个分类器针对两个类别对数据进行训练。
  - 为了提供一个和其它分类器一致的接口，选项decision_function_shape允许调用者将所有“一对一”分类器的结果聚合进一个(n_samples, n_classes)的决策函数，其中n_classes=C
- LinearSVC实现了“一对多”分类法（One vs. Rest, OvR），因此会训练C个模型。

得分与概率

- SVC中的decision_function方法对每个样本都会给出在各个类别上的分数（在二元分类问题中，是对每个样本给出一个分数）。
- 如果构造函数的probability被设为True，则可以得到属于每个类别的概率估计（通过predict_proba和predict_log_proba方法）。概率使用Platt缩放进行调整，通过在训练集上做额外的交叉检验来拟合一个在SVM分数上的Logistic回归。
  - Platt缩放中的交叉检验在大数据集上是一个代价很高的操作。
  - 概率估计与实际得分可能会不一致，即使得分取得了最大值，概率并不一定也能取到最大值。
  - Platt的方法在理论上也有一些问题。
  - 如果需要拿到置信分数，而这些分数又不一定非得是概率，则建议把probability置为False，并且使用decision_function，而不是predict_proba。

计算复杂度

- 支持向量机是一种能力很强的工具，但是随着训练样本数目的增加，它们对计算和存储资源的需求也会快速增长。
- SVM的核心是二次规划问题（QP）。令特征维数为D，训练样本数目为N，取决于libsvm缓存在实践中使用的效率（依赖于数据集），基于libsvm的实现所用的QP求解器时间复杂度在（$D*N^2$）到（$D*N^3$）不等。如果数据非常稀疏，上述复杂度中的D可用一个样本中的平均非零特征数目代替。
- 还要注意的是，对线性的情况，基于liblinear实现的LinearSVC的算法比基于libsvm实现的SVC效率要高很多，而且前者在处理以百万计的样本和/或特征时仅以线性增长。

应用建议

- 避免数据拷贝：对SVC、SVR、NuSVC和NuSVR，如果传给特定方法的数据不是以C语言所使用的顺序排列，而且是double精度，那么该数据会在调用底层C实现之前被拷贝一份。
  - 通过检查flag属性，可以检查给定的numpy数组是否是以C格式的连续存储方式排列的。
  - 对LinearSVC和LogisticRegression，任何以numpy数组形式传入的输入都会被拷贝，然后转化为liblinear内部的稀疏数据表示形式（双精度浮点数，对非零元素存储32位整型的索引）。
  - 如果想训练一个大规模的线性分类器，而又不想拷贝一个稠密的numpy C-存储双精度数组，建议使用SGDClassifier。可以对它的目标函数进行配置，使其与LinearSVC模型所使用的基本相同。

► 大纲

- SVM基本原理
- 带松弛因子的SVM：C-SVM
- 核方法
- 支持向量回归
- Scikit learn中的SVM实现
- 案例分析

案例分析：Otto商品分类

总结

- SVM基本原理
  - 最大间隔原则
  - 对偶表示(Dual Representation)
  - KKT条件
- 带松弛因子的SVM：C-SVM
  - 合页损失（hinge loss）
- 核技巧
- 支持向量回归
  - ϵ不敏感损失（ ϵ insensitive loss）
- Scikit learn中的SVM实现
- 案例分析

总结

- SVM的优点：
  - 在高维空间中行之有效
  - 当维数大于样本数时仍然可用（但性能不好）
  - 在决策函数中只使用训练点的一个子集（支持向量），大大节省了内存开销
  - 用途广泛：决策函数中可使用不同的核函数
- 劣势：SVM不直接提供概率估计
  - 可通过交叉验证计算，代价比较高
- Scikit-learn中的支持向量机同时支持密集样本向量（numpy.ndarray和可通过numpy.asarray转化的数据类型）和稀疏样本向量（任何scipy.sparse对象）。但如果想用SVM对稀疏数据进行预测，则必须先在这些数据上拟合。为了优化性能，应该使用C阶（C-Ordered）numpy.ndarray（密集的）或scipy.sparse.csr_matrix（稀疏的），并指定dtype=float64。

CSDN
不 止 于 代 码

THANK YOU

AI 100