1.4. Support Vector Machines

1.4. 支持向量机

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

支持向量机（SVMs）是一组用于分类、回归和离群点检测的监督学习方法。

The advantages of support vector machines are:

支持向量机的优点是：

Effective in high dimensional spaces.

在高维空间中非常有效。

Still effective in cases where number of dimensions is greater than the number of samples.

在样本数量小于维度数量的情况下仍然有效。

Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

在决策函数中使用训练点的子集（称为支持向量），因此它也具有内存效率。

Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

灵活：可以为决策函数指定不同的核函数。提供了常见的核函数，但也允许指定自定义核函数。

The disadvantages of support vector machines include:

支持向量机的不利之处包括：

If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.

如果特征数量远大于样本数量，选择核函数时要避免过拟合，正则化项至关重要。

SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

支持向量机不直接提供概率估计，这些是通过昂贵的五折交叉验证计算得出的（见下文的分数和概率）。

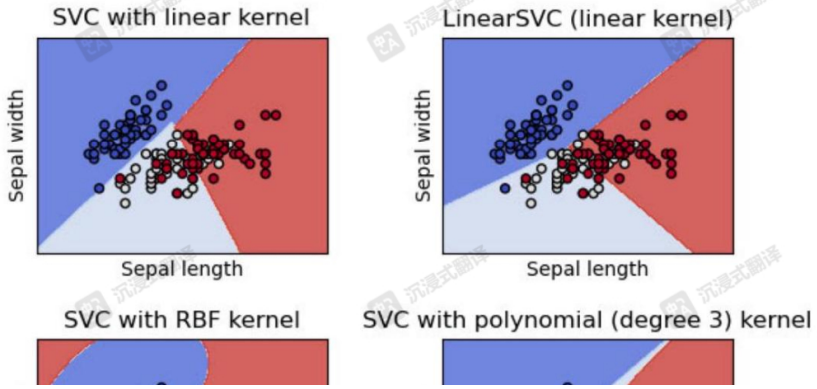
The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered numpy.ndarray (dense) or scipy.sparse.csr_matrix (sparse) with dtype=float64.

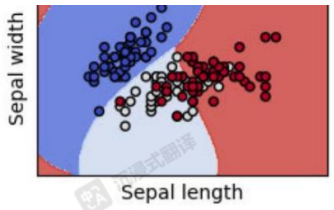
scikit-learn 中的支持向量机支持密集（numpy.ndarray 和可转换为该类型）和稀疏（任何 scipy.sparse）样本向量作为输入。但是，要使用 SVM 对稀疏数据进行预测，它必须在这样的数据上拟合。为了最佳性能，使用 C 顺序的 numpy.ndarray（密集）或 scipy.sparse.csr_matrix（稀疏）并指定 dtype=float64。

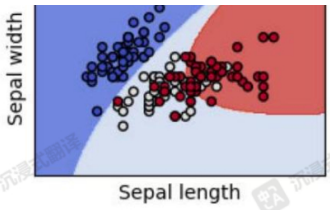
1.4.1. Classification 1.4.1. 分类

SVC, NuSVC and LinearSVC are classes capable of performing binary and multi-class classification on a dataset.

SVC、NuSVC 和 LinearSVC 是能够在数据集上执行二元和多类分类的类。







SVC and NuSVC are similar methods, but accept slightly different sets of parameters and have different mathematical formulations (see section Mathematical formulation). On the other hand, LinearSVC is another (faster) implementation of Support Vector Classification for the case of a linear kernel. Note that LinearSVC does not accept parameter kernel, as this is assumed to be linear. It also lacks some of the attributes of sVC and NuSVC, like support_.

SVC 和 NuSVC 是相似的方法，但接受略有不同的参数集，并且具有不同的数学公式（见数学公式部分）。另一方面，LinearSVC 是支持向量分类在线性核情况下的另一种（更快）实现。请注意，LinearSVC 不接受参数 kernel，因为假设它是线性的。它还缺少一些 sVC 和 NuSVC 的属性，比如 support_。

As other classifiers, SVC, NuSVC and LinearSVC take as input two arrays: an array X of shape (n_samples, n_features) holding the training samples, and an array y of class labels (strings or integers), of shape (n_samples):

与其他分类器一样，SVC、NuSVC 和 LinearSVC 作为输入接受两个数组：一个形状为(n_samples, n_features)的数组 X，用于存储训练样本，以及一个形状为(n_samples)的类标签数组 y（字符串或整数）：

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>)clf fit(X, y)
Toggle Menu
```

After being fitted, the model can then be used to predict new values:

拟合后，模型可以用来预测新值：

```
>>> clf.predict([[2., 2.]])
array([1])
```

SVMs decision function (detailed in the Mathematical formulation) depends on some subset of the training data, called the support vectors.

支持向量机（detailed in the Mathematical formulation）的决策函数取决于训练数据的一个子集，称为支持向量。

Some properties of these support vectors can be found in attributes support_vectors_, support_ and n_support_:

这些支持向量的某些属性可以在 support_vectors_、support_和 n_support_属性中找到：

```
>>> # get support vectors
>>> clf.support_vectors_
array([[0., 0.],
    [1., 1.]])
>>> # get indices of support vectors
>>> clf.support_
array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_
array([1, 1]...)
```

Examples: 例子：

SVM: Maximum margin separatinghyperplane,

SVM：最大间隔分离超平面，

Non-linear SVM 非线性支持向量机

SVM-Anova: SVM with univariate feature selection,

SVM-Anova：具有单变量特征选择的支持向量机，

1.4.1.1. Multi-class classification

1.4.1.1. 多类分类

SVC and NuSVC implement the “one-versus-one” approach for multi-class classification. In total

SVC and NuSVC implement the "one-versus-one" approach for multi-class classification. In total, n_classes * (n_classes - 1) / 2 classifiers are constructed and each one trains data from two classes. To provide a consistent interface with other classifiers, the decision_function_shape option allows to monotonically transform the results of the "one-versus-one" classifiers to a "one-vs-rest" decision function of shape (n_samples, n_classes).

SVC 和 NuSVC 实现了"一对一"的多类分类方法。总共构建了 n_classes * (n_classes - 1) / 2 个分类器，每个分类器训练两个类别的数据。为了与其他分类器提供一致的接口，decision_function_shape 选项允许将"一对一"分类器的结果单调地转换为形状为(n_samples, n_classes)的"一对多"决策函数。

```
>>> X = [[0], [1], [2], [3]]
>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC(decision_function_shape='ovo')
>>> clf.fit(X, Y)
SVC(decision_function_shape='ovo')
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes: 4*3/2 = 6
6
>>> clf.decision_function_shape = "ovr"
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes
4
```

On the other hand, LinearSVC implements "one-vs-the-rest" multi-class strategy, thus training n_classes models.

另一方面，LinearSVC 实现了"一对一"的多类策略，因此训练了 n_classes 个模型。

```
>>> lin_clf = svm.LinearSVC()
>>> lin_clf.fit(X, Y)
LinearSVC()
>>> dec = lin_clf.decision_function([[1]])
>>> dec.shape[1]
4
```

See Mathematical formulation for a complete description of the decision function.

请参阅数学公式，以了解决策函数的完整描述。

Note that the LinearSVC also implements an alternative multi-class strategy, the so-called multi-class SVM formulated by Crammer and Singer [16], by using the option multi_class=' crammer_singer' . In practice, one-vs-rest classification is usually preferred, since the results are mostly similar, but the runtime is significantly less.

注意，LinearSVC 还实现了一种替代的多类策略，即由 Crammer 和 Singer [16] 提出的多类 SVM，通过使用选项 multi_class=' crammer_singer' 。实际上，通常更喜欢"一对一"分类，因为结果大多相似，但运行时间显著减少。

For "one-vs-rest" LinearSVC the attributes coef_ and intercept_ have the shape (n_classes, n_features) and (n_classes,) respectively. Each row of the coefficients corresponds to one of the n_classes "one-vs-rest" classifiers and similar for the intercepts, in the order of the "one" class.

对于"一对一"的 LinearSVC，属性 coef_ 和 intercept_ 的形状分别为 (n_classes, n_features) 和 (n_classes,) 。系数的每一行对应于 n_classes 个"一对一"分类器中的一个，截距也类似，按"一对一"分类器的顺序排列。

In the case of "one-vs-one" SVC and NUSVC, the layout of the attributes is a little more involved. In the case of a linear kernel, the attributes coef_ and intercept_ have the shape ($n\_classes * (n\_classes - 1) / 2, n\_features$) *and* ($n\_classes * (n\_classes - 1) / 2$) *respectively* similar to the layout for LinearSVC described above, with each row now corresponding to a binary classifier. The order for classes 0 to n is " 0 vs 1", "0 vs 2" , ... "0 vs n", "1 vs 2", "1 vs 3", " 1 vs n", . . "n-1 vs n".

在"一对一"SVC 和 NUSVC 的情况下，属性的布局稍微复杂一些。对于线性核，属性 coef_和 intercept_的形状（ $n\_classes * (n\_classes - 1) / 2, n\_features$) *and* ($n\_classes * (n\_classes - 1) / 2$) *respectively. This is* 类似于上面描述的 LinearSVC 的布局，现在每一行对应一个二元分类器。类别 0 到 n 的顺序是"0 vs 1"，"0 vs 2"，... "0 vs n"，"1 vs 2"，"1 vs 3"，" 1 vs n"，...“n-1 vs n”。

The shape of dual_coef_ is (n _classes-1, n_sV) with a somewhat hard to grasp layout. The columns correspond to the support vectors involved in any of the n_classes * (n_classes - 1) / 2 "one-vs-one" classifiers. Each of the support vectors is used in n_classes - 1 classifiers. The n_classes - 1 entries in each row correspond to the dual coefficients for these classifiers.

dual_coef_的形状是（ n _classes-1, n_sV ），布局有些难以理解。列对应于任何 n_classes * (n_classes - 1) / 2 个"一对一"分类器中涉及的支撑向量。每个支撑向量用于 n_classes - 1 个分类器。每行的 n_classes - 1 个条目对应于这些分类器的对偶系数。

This might be clearer with an example: consider a three class problem with class 0 having three support vectors v^0_0, v^1_0, v^2_0 and class 1 and 2 having two support vectors v^0_1, v^1_1 and v^0_2, v^1_2 respectively. For each support vector v^j_i, there are two dual coefficients. Let's call the coefficient of support vector v^j_i in the classifier between classes i and $k\alpha^j_{i,k}$. Then dual_coef_ looks like this:

这可能通过一个例子更清楚：考虑一个三类问题，类别 0 有三个支撑向量 v_0^0, v_0^1, v_0^2，类别 1 和 2 分别有两个支撑向量 v_1^0, v_1^1 和 v_2^0, v_2^1。对于每个支撑向量 v_i^j，有两个对偶系数。让我们称支持向量 v_i^j 在类别 i 和 $k\alpha_{i,k}^j$ 之间的分类器中的系数。那么 dual_coef_看起来是这样的：

$\alpha_{0,1}^0$	$\alpha_{0,2}^0$	Coefficients for SVs of class 0 类别 0 的 SV 系数
$\alpha_{0,1}^1$	$\alpha_{0,2}^1$	
$\alpha_{0,1}^2$	$\alpha_{0,2}^2$	
$\alpha_{1,0}^0$	$\alpha_{1,2}^0$	Coefficients for SVs of class 1 类别 1 的 SV 系数
$\alpha_{1,0}^1$	$\alpha_{1,2}^1$	
$\alpha_{2,0}^0$	$\alpha_{2,1}^0$	Coefficients for SVs of class 2 类别 2 的 SV 系数
$\alpha_{2,0}^1$	$\alpha_{2,1}^1$	

Examples: 示例：

Plot different SVM classifiers in the iris dataset,

在鸢尾花数据集上绘制不同的 SVM 分类器，

1.4.1.2. Scores and probabilities

1.4.1.2. 分数和概率

The decision_function method of SVC and NuSVC gives per-class scores for each sample (or a single score per sample in the binary case). When the constructor option probability is set to True, class membership probability estimates (from the methods predict_proba and predict_log_proba) are enabled. In the binary case, the probabilities are calibrated using Platt scaling [9]: logistic regression on the SVM's scores, fit by an additional cross-validation on the training data. In the multiclass case, this is extended as per [10].

SVC 和 NuSVC 的 decision_function 方法为每个样本提供每类的分数（在二分类情况下，每个样本提供一个分数）。当构造函数选项 probability 设置为 True 时，将启用类成员概率估计（来自 predict_proba 和 predict_log_proba 方法）。在二分类情况下，概率使用 Platt 标准化 [9] 进行校准：对 SVM 的分数进行逻辑回归，并通过在训练数据上进行的额外交叉验证来拟合。在多分类情况下，按照 [10] 扩展此方法。

Note: The same probability calibration procedure is available for all estimators via the CalibratedClassifiercv (see Probability calibration). In the case of SVC and NuSVC, this procedure is builtin in libsvm which is used under the hood, so it does not rely on scikit-learn's CalibratedClassifierCV.

注意：相同的概率校准程序可通过 CalibratedClassifiercv 为所有估计器提供（参见概率校准）。在 SVC 和 NuSVC 的情况下，此程序是 libsvm 底层使用的内置功能，因此它不依赖于 scikit-learn 的 CalibratedClassifierCV。

The cross-validation involved in Platt scaling is an expensive operation for large datasets. In addition, the probability estimates may be inconsistent with the scores:

Platt 标准化中涉及的交叉验证对于大型数据集来说是一个昂贵的操作。此外，概率估计可能与分数不一致：

the "argmax" of the scores may not be the argmax of the probabilities

分数的"argmax"可能不是概率的 argmax

in binary classification, a sample may be labeled by predict as belonging to the positive class even if the output of predict_proba is less than 0.5 ; and similarly, it could be labeled as negative even if the output of predict_proba is more than 0.5 .

在二分类中，即使 predict_proba 的输出小于 0.5，样本也可能被标记为属于正类；类似地，即使 predict_proba 的输出大于 0.5，它也可能被标记为负类。

Platt's method is also known to have theoretical issues. If confidence scores are required, but these do not have to be probabilities, then it is advisable to set probability=False and use decision_function instead of predict_proba.

Platt 方法也已知存在理论问题。如果需要置信度分数，但不需要是概率，则建议设置 probability=False 并使用 decision_function 而不是 predict_proba。

Please note that when decision_function_shape='ovr' and n_classes > 2, unlike decision_function, the predict method does not try to break ties by default. You can set break_ties=True for the output of predict to be the same as np.argmax(clf.decision_function(...), axis=1), otherwise the first class among the tied classes will always be returned; but have in mind that it comes with a computational cost. See SVM Tie Breaking Example for an example on tie breaking.

请注意，当 decision_function_shape='ovr'且 n_classes > 2 时，与 decision_function 不同，predict 方法默认不会尝试打破平局。您可以为 predict 的输出设置 break_ties=True，使其与 np.argmax(clf.decision_function(...), axis=1)相同，否则将始终返回平局类中的第一个类；但请记住，这会带来计算成本。有关平局打破的示例，请参阅 SVM 平局打破示例。

1.4.1.3. Unbalanced problems

1.4.1.3. 不平衡问题

In problems where it is desired to give more importance to certain classes or certain individual samples, the parameters class_weight and sample_weight can be used.

在希望对某些类别或某些单个样本给予更多重视的问题中，可以使用 class_weight 和 sample_weight 参数。

SVC (but not NuSVC) implements the parameter class_weight in the fit method. It's a dictionary of the form {class_label : value}, where value is a floating point number > 0 that sets the parameter C of class class_label to C* value. The figure below illustrates the decision boundary of an unbalanced problem, with and without weight correction.

SVC（但 NuSVC 不实现）在 fit 方法中实现了 class_weight 参数。它是一个形式为{类别标签：值}的字典，其中值是一个浮点数 > 0，该值将类别标签为 class_label 的参数 C 设置为 C* 值。下图说明了不平衡问题的决策边界，包括和不包括权重校正的情况。