SVM

22 April 2023 19:35

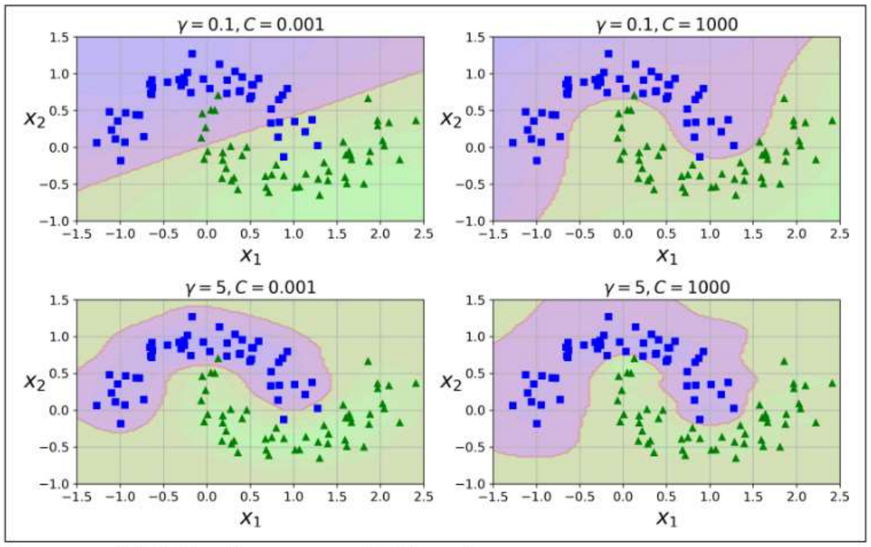
- SVM are sensitive to feature scale , so do normalization / standarization if needed.
- If your SVM model is overfitting, you can try regularizing it by reducing C .
- SVM
  - `C` parameter in sklearn svm:
    - Higher C , overfitting
    - Lower C , regulization
  - 'Gamma' parameter in sklearn introduces:
    - Higher Gamma, overfitting(similar to the C hyperparameter)
    - kernel tricks
    - to fit non linear hyperplance to create boundary
    - increase this to increase non-linearity
    - 

Figure 5-9. SVM classifiers using an RBF kernel

- Unlike Logistic Regression classifiers, SVM classifiers do not output probabilities for each class.
- Recommended , use svm.LinearSVC() in place of svm.SVC(kernel="linear", C=1)

- Alternatively, you could use the SVC class, using SVC(kernel="linear", C=1), but it is much slower, especially with large training sets, so it is not recommended
- Another option is to use the SGDClassifier class, with SGDClassifier(loss="hinge",alpha=1/(m*C))
- For LinearSVC class:
  * Use StandardScaler
  * set `loss` hyperparameter to hinge as it is not by default, loss='hinge'
  * set `dual` hyperparameter to False for better performance

- WHICH KERNEL TO CHOOSE AMONG SO MANY?
  - always try the linear kernel first (remember that LinearSVC is much faster than SVC(kernel="linear"))
  - If the training set is not too large, you should try the Gaussian RBF kernel as well; it works well in most cases.
  - Then if you have spare time and computing power, you can also experiment with a few other kernels using cross-validation and grid search, especially if there are kernels specialized for your training set's data structure.

Computational Complexity

The `LinearSVC` class is based on the *liblinear* library, which implements an optimized algorithm for linear SVMs.[1] It does not support the kernel trick, but it scales almost

1 "A Dual Coordinate Descent Method for Large-scale Linear SVM," Lin et al. (2008).

linearly with the number of training instances and the number of features: its training time complexity is roughly $O(m \times n)$.

The algorithm takes longer if you require a very high precision. This is controlled by the tolerance hyperparameter ϵ (called `tol` in Scikit-Learn). In most classification tasks, the default tolerance is fine.

The `SVC` class is based on the *libsvm* library, which implements an algorithm that supports the kernel trick.[2] The training time complexity is usually between $O(m^2 \times n)$ and $O(m^3 \times n)$. Unfortunately, this means that it gets dreadfully slow when the number of training instances gets large (e.g., hundreds of thousands of instances). This algorithm is perfect for complex but small or medium training sets. However, it scales well with the number of features, especially with *sparse features* (i.e., when each instance has few nonzero features). In this case, the algorithm scales roughly with the average number of nonzero features per instance. Table 5-1 compares Scikit-Learn's SVM classification classes.

Table 5-1. Comparison of Scikit-Learn classes for SVM classification

Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes