Support Vector Machines

An SVM is a versatile machine-learning model that can perform linear or nonlinear classification, regression, and novelty detection.

Linear SVM Classification

- An SVM classifier tries to fit the widest possible street (the space between the decision boundary) between the classes, a process called large machine classification.
- Adding more training instances “off the street” does not affect the decision boundary.
    - The decision boundary is determined by the instances on the edge of the street. These instances are called support vectors.
- SVMs are sensitive to feature scales, so scaling may be necessary (e.g., using a Standard Scaler).

Soft Margin Classification

- Hard margin classification is strictly imposing that all instances must be off the street and on the correct side.
    - This type of classification only works if the data is linearly separable.
    - It is sensitive to outliers.
- We need to balance keeping the street as wide as possible and limiting the margin violations. This is called soft margin classification.
- We can use the **LinearSVC()** class from Scikit-Learn. Note that all classes or functions mentioned forward will be from Scikit-Learn unless specified otherwise.
    - This classifier has the fit(), predict(), and decision_function() methods.
    - The hyperparameter “C” controls the width of the street. Reducing C widens the street, and increasing it narrows the street.

Nonlinear SVM Classification

- Many datasets are often not linearly separable. In such cases, we can add polynomial features.
- We can use the **PolynomialFeatures()** class to add those features.
    - We can use a pipeline to add these features, scale the data, and use the Linear SVM classifier.

Polynomial Kernel

- A low polynomial degree model will not handle very complex datasets well.
- Adding too many polynomial features will make the model too slow.
- The kernel trick can simulate a high-degree polynomial model without adding the polynomial features.

    - We simply set the kernel parameter to “poly” when creating the **SVC()** object.
    - The hyperparameter “coef0” controls how much high-degree terms versus low-degree terms influence the model.
    - We can reduce the polynomial degree if the model overfits the training data.

Similarity Features

- Another way to deal with nonlinear datasets is to add similarity features to the data.
    - We add features computed using a similarity function, which measures how much each instance resembles a particular landmark.
    - The simplest approach to accomplish this task is to create a landmark at each instance of the dataset.
- We can define a similarity function as a Gaussian RBF Kernel.
    - We can simply set the kernel parameter of the SVC() class to “rbf.”
    - The gamma parameter controls the width of the curve.
        - Increasing the gamma value narrows the curve, making the decision boundary more irregular.
        - Reducing the gamma value widens the curve, smoothening the decision boundary.
        - If the model underfits the data, we increase gamma and decrease it if the model overfits.
- As a rule of thumb, we should always try the linear kernel first.
    - The LinearSVC class is much faster than SVC(kernel="linear"), especially if the training set is very large.
    - If it is not too large, we should also try kernelized SVMs, starting with the Gaussian RBF kernel.

SVM Regression

- SVM regression tries to fit as many instances as possible on the street while limiting margin violations.
    - The hyperparameter epsilon controls the width of the street.
    - Reducing ϵ (epsilon) increases the number of support vectors, which regularizes the model.
    - Adding more training instances within the margin will not affect the model’s predictions, so the model is ϵ-insensitive.
- We can use the **LinearSVR()** class for SVM regression.
- For nonlinear regression tasks, we can use a kernelized SVM model.
    - We can use the **SVR()** class and set the kernel parameter to “poly.”