Лабораторная работа №10. Машина опорных векторов.

В данной работе вы получите навыки применения машины опорных векторов (SVM – support vector machine) для классификации различных наборов данных и использованием линейного и Гауссовского ядра. Теоретический материал для данной лабораторной работы можно найти в разделе 7 учебного пособия. Сам алгоритм SVM вам программировать не нужно. Вместо этого будет задействована реализация SVM в языке Python из библиотеки Scikit-learn (https://scikit-learn.org/stable/modules/svm.html).

В работе будет использовано три набора данных различной сложности для демонстрации различных аспектов SVM.

1 Загрузка и визуализация первого набора данных

На этом шаге загружается файл ex6data1.npy, который содержит матрицу $X \in \mathbb{R}^2$ и вектор y значений двух классов $\{0,1\}$. После загрузки данные отображаются на плоскости (см. рис. 1). Точки класса $y^{(i)} = 0$ показаны красным цветом, а точки $y^{(i)} = 1$ зеленые. Между обучающими примерами разных классов имеется значительный зазор, классы хорошо разделимы. Обратите внимание, что в левой части график имеется отдельно стоящий пример класса $y^{(i)} = 1$.

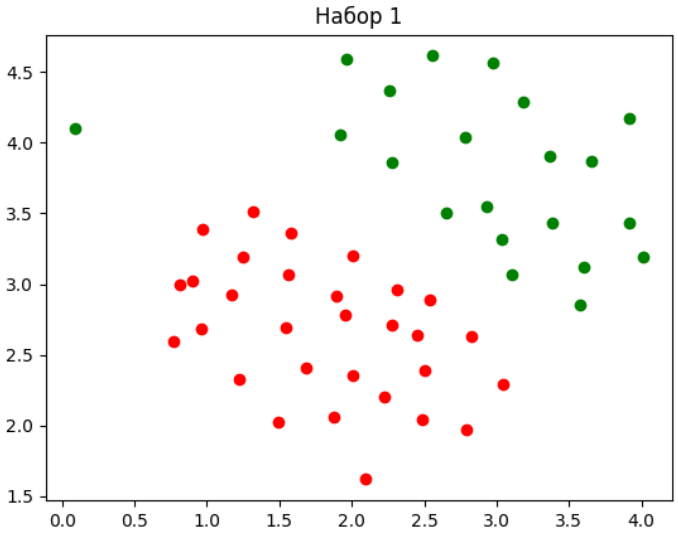

Рис. 1. Первый набор данных.

На данном шаге вам не нужно писать собственного кода. Удалите оператор return после первого шага в функции main, чтобы перейти ко второму шагу.

2 Обучение SVM с линейным ядром

На этом шаге вам также не нужно писать собственный код. Найдите функцию svm_learn и ознакомьтесь с примером обучения SVM, данным в программе.

Для обучения SVM в библиотеке scikit-learn имеется подмодуль svm. Вызов svm.SVC создает объект классификатора методом SVM. Параметры функции:

C – параметр функции стоимости SVM;

kernel – вид ядра SVM, в работе мы используем два ядра: ‘linear’ – линейное ядро (т.е. без ядра), ‘rbf’ – ядро радиальный базисный функций, к которым относится ядро Гаусса;

gamma – параметр ядра RBF;

tol – критерий остановки алгоритма (по умолчанию 0.001);

max_iter – ограничение на количество итераций (-1 – без ограничений);

Есть и другие параметры, который мы рассматривать не будем.

Параметр gamma, который задает ядро RBF может быть легко вычислен на основе параметра σ ядра Гаусса. Ядро RBF вычисляется по формуле:

$$K(x, l) = \exp(-\gamma \|x - l\|^2),$$

тогда как ядро Гаусса (частный случай ядра RBF) вычисляется как:

$$K(x, l) = \exp\left(-\frac{\|x - l\|^2}{2\sigma^2}\right).$$

Легко видно, что:

$$\gamma = \frac{1}{2\sigma^2}.$$

После того, как объект clf классификатора SVM инициализирован вызовом

clf=svm.SVC(…),

его можно обучить на выборке вызовом метода fit:

model = clf.fit(X, y),

где X, y – обучающая выборка.

Теперь в переменной model содержится готовый и обученный классификатор алгоритмом SVM.

Его методу predict можно передать матрицу значений X для предсказания их классов:

pred_y = model.predict(new_X)

Теперь, когда вы ознакомились с обучением и использованием SVM в библиотеке scikit-learn, на втором шаге лабораторной работы выполняется обучение SVM для классификации первого набора данных с двумя значениями параметра $C = 1$ и $C = 100$. Для каждой из обученный моделей выводится график границы решения. Оцените разницу в графиках и определите, по какой причине такая разница возникает (см. учебное пособие разд. 7).

Удалите оператор return, чтобы перейти к следующему шагу.

3 Ядро Гаусса

На этом шаге вам необходимо запрограммировать вычисление ядра Гаусса. Найдите в программе функцию gaussian_kernel и дополните ее кодом до рабочего состояния. Функция принимает на вход параметры:

x1, x2 – две точки (вектора одинаково длины), расстояние между которыми оценивается;

sigma – параметр σ ядра Гаусса.

Функция должна возвращать значение ядра Гаусса для x1, x2.

После того, как функция готова, запустите программу. Программа вычисляет ядро Гаусса для заданных значений. Полученные значения должны быть близки к ожидаемым. Удалите оператор return, чтобы перейти к следующему шагу.

4 Загрузка и визуализация второго набора данных

Второй набор данных загружается из файла ex6data2.npy, и он более сложный, чем первый. Классы уже на являются линейно разделимыми (см. рис. 2).

После загрузки второго набора данных в работе с использованием функции svm_learn выполняется обучение SVM с линейным ядром на этих данных и выводится граница решения обученной модели. Оцените, какие недостатки у полученной модели.

Удалите оператор return, чтобы перейти к следующему шагу.

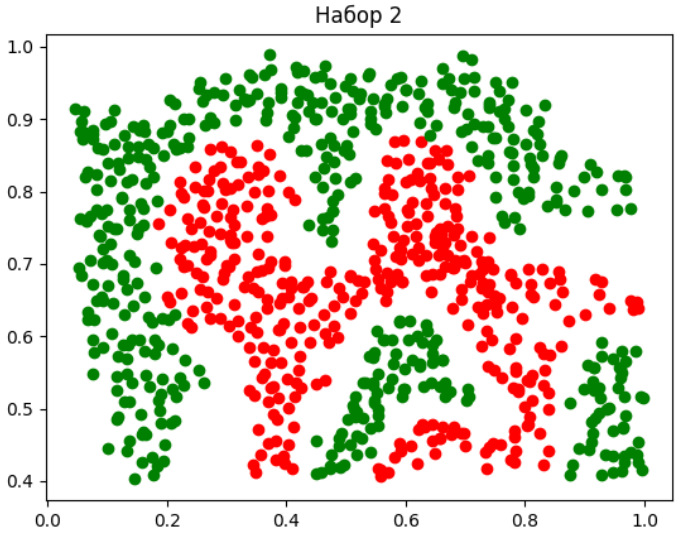

Рис. 2. Второй набор данных.

5 Обучение SVM с ядром Гаусса

Для устранения недостатков модели SVM с линейным ядром для второго набора данных, на этом шаге выполняется обучение модели с ядром Гаусса. Для этого используется та же функция svm_learn. После обучения выводится граница решения построенной модели. Оцените, насколько эта модель лучше.

Удалите оператор return, чтобы перейти к следующему шагу.

6 Загрузка и отображение третьего набора данных

На данном шаге загружается третий набор данных из файла ex6data3.npy. В отличие от первых двух, набор содержит не только два линейно неразделимых класса, но также между классами нет отступа, они перекрываются друг с другом.

После загрузки и отображения набора на плоскости, выполняется обучение SVM с ядром Гаусса при двух вариантах параметров модели: $C = 2, \sigma = 0.01$ и $C = 2, \sigma = 5$. Для каждой из моделей отображается граница решения полученного классификатора. Оцените эти графики и сделайте выводы, какие проблемы имеют два предложенных решения (возможно, какое-то из них имеет проблему смещения или дисперсии).

Удалите оператор return, чтобы перейти к следующему шагу.

7 Подбор параметров для модели

Для построения хорошей модели SVM на данном шаге работы вам необходимо написать функцию подбора значений C и σ. В лабораторной работе по диагностики машинного обучения вам уже приходилось выполнять подбор параметра регуляризации линейной регрессии по валидационной выборке. В данной работе выполняются аналогичные действия с отличием, что подбирать придется не один, а два параметра.

Найдите в программе функцию adjust_params и дополните ее кодом до рабочего состояния. Функция принимает на вход параметры:

X, y – обучающая выборка;

Xval, yval – валидационная выборка.

Функция должна вернуть два значения: наилучшее значение C и наилучшее значение σ.

Для подбора можно использовать несколько вариантов значений C (например,

0.1, 0.5, 1., 5., 100., 1000.) и несколько значений σ (например, 0.05, 0.1, 0.3, 0.5, 1., 3.). Для каждой пары значений нужно обучить модель SVM с ядром гаусса. Для этого можно использовать уже готовую функцию svm_learn.

Обученную модель можно использовать для предсказания значений валидационной выборки, например, так:

pred = model.predict(Xval)

В результате будет получен вектор предсказанных классов для валидационной выборки, который можно сравнить с вектором yval (например, подсчитать сколько модель выдает ошибок предсказания).

После того, как функция готова и получены наиболее удачные значения параметров C и σ, удалите оператор return, чтобы перейти к следующему шагу.

8 Оценка модели

На данном шаге вам не нужно писать собственного кода. Программа использует найденные вами на предыдущем шаге значения параметров C и σ для обучения модели SVM с ядром Гаусса, затем строит границу решения обученной модели и оценивает показатель ошибки модели на валидационной выборке.

В зависимости от найденных значений C и σ, вид границы решения может немного отличаться, но в целом он должен быть похож на рис. 3 (граница построена при $C = 1, \sigma = 0.1$). Вероятность ошибки для хорошей модели не должна превышать 0.05.

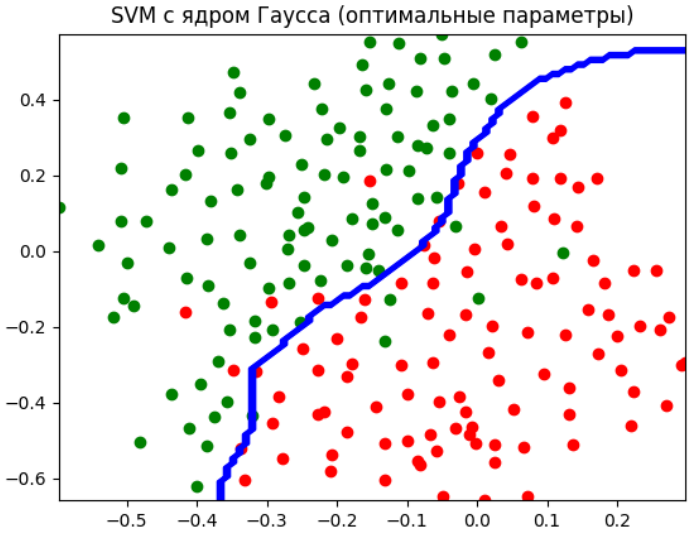

Рис. 3. Граница решения хорошей модели на третьем наборе данных.

На этом выполнение лабораторной работы №10 завершается.