Отчет по лабораторной работе №2
по курсу «Искусственный интеллект»

Выполнил студент группы М8О-304б-16 Величутин Андрей

Тема: Алгоритмы машинного обучения

Задача:
Требуется реализовать класс на выбранном языке программирования, который реализует один из алгоритмов машинного обучения. Обязательным является наличия в классе двух методов fit, predict. Необходимо проверить работу вашего алгоритма на ваших данных (на таблице и на текстовых данных), произведя необходимую подготовку данных. Также необходимо реализовать алгоритм полиномиальной регрессии, для предсказания значений в таблице. Сравнить результаты с стандартной реализацией sklearn, определить в чем сходство и различие ваших алгоритмов. Замерить время работы алгоритмов.

Вариант: 3. Метод опорных векторов

Оборудование студента:
Intel(R) Core(TM) i5-5200 CPU @ 2.20GHz 2.19GHz
ОЗУ 6,00 ГБ

Программное обеспечение:
Windows 10, Python 3.7.4(С библиотеками Pandas, Numpy, Seaborn и Scikit-Learn), Jupyter notebook 4.4.0

Ход работы:

Модели линейных классификаторов не так сложны и по сути сводятся к оптимизации функционала ошибок. Однако в случае метода опорных векторов возникает проблема решения задачи квадратичного программирования. Классические методы решения таких задачи неэффективны, когда речь идет о реальных задачах, поэтому было придумано множество методов именно для обучения SVM. Один из таких методов — Simplified SMO. Прочитать про данный метод можно по этой ссылке: http://cs229.stanford.edu/materials/smo.pdf

Реализация метода опорных векторов: https://github.com/BeJIuK/ML/blob/master/Lab2/svm.py

Данная реализация была протестирована на датасете из лабораторной работы №0. Ссылка на код для тестировния: https://github.com/BeJIuK/ML/blob/master/Lab2/benchmark.py

Были получены следующие результаты:

```
~/L/M/Lab2 python benchmark.py
Data set for classification:
    bone_length  rotting_flesh  hair_length  has_soul  type  color_black  color_blood  color_blue  color_clear  color_green  color_white
id
0      0.354512       0.350839     0.465761  0.781142    -1            0            0           0            1            0            0
1      0.575560       0.425868     0.531401  0.439899     1            0            0           0            0            1            0
2      0.467875       0.354330     0.811616  0.791225    -1            1            0           0            0            0            0
4      0.776652       0.508723     0.636766  0.884464    -1            1            0           0            0            0            0
7      0.405680       0.253277     0.441420  0.280324     1            0            0           0            0            1            0
8      0.399331       0.568952     0.618391  0.467901     1            0            0           0            0            0            1
11     0.516224       0.536429     0.612776  0.468048    -1            0            0           0            1            0            0
22     0.431685       0.438959     0.239212  0.471820     1            0            0           0            1            0            0
23     0.584543       0.593082     0.681166  0.935721    -1            0            0           0            1            0            0
25     0.390712       0.335069     0.556109  0.784217    -1            0            0           0            0            0            1
|------------------------------------------------------------|
                    Testing my SVM model
Training model...
Model trained. Elapsed time: 2.4365274906158447
Testing model...
Accuracy score on test set: 0.6666666666666666
|------------------------------------------------------------|
                 Testing sklearn SVM model
Training model...
Model trained. Elapsed time: 0.0022101402282714844
Testing model...
Accuracy score on test set: 0.6274509803921569
```

Реализованный мною алгоритм обучения не гарантирует точной сходимости, однако для данного датасета удалось получить точность классификации на уровне стандартной реализации из sklearn. Среди особенностей стоит заметить очень медленную сходимость. 2 секунды для обучения на датасете из 300 записей — очень плохой результат. Из-за такой медленной сходимости не удалось нормально протестировать работу алгоритма на корпусе текстовых документов.

По заданию лабораторной работы необходимо было реализовать полиномиальную регрессию. Однако для моих датасетов трудно спроектировать признак для регрессии. Поэтому я решил создать искусственный датасет — сгенерировать значения нелинейной функции двух переменных со случайным шумом

Полиномиальная регрессия сводится к линейной путем добавления полиномиальных признаков. Для этого я использовал готовую реализацию (PolynomialFeatures из библиотеки sklearn). Так как я сгенерировал довольно маленький датасет, для нахождения коэффициентов линейной регрессии я использовал аналитическое решение МНК

Ссылка на реализацию: https://github.com/BeJIuK/ML/blob/master/Lab2/polynomial_regression.py

Результаты:

```
! ~/L/M/Lab2 python polynomial_regression.py
Degree: 0; MSE: 261.7217055238973
Degree: 1; MSE: 1.9980643868830454
Degree: 2; MSE: 1.1305044210142468
Degree: 3; MSE: 0.8055444644354278
Degree: 4; MSE: 0.7480395941826278
Degree: 5; MSE: 0.655996321681716
Degree: 6; MSE: 0.6103254689113793
Degree: 7; MSE: 0.5667472795213074
Degree: 8; MSE: 0.4841790828253066
Degree: 9; MSE: 1.433415411813148
Degree: 10; MSE: 801.1940453089528
Degree: 11; MSE: 41164.456280742845
Degree: 12; MSE: 22215.992080768498
Degree: 13; MSE: 2528.1465317799725
Degree: 14; MSE: 39657700.81669301
```

Выводы:
Данная работа показалась мне достаточно сложной в виду слишком замудренного метода оптимизации (Даже при условии, что в названии стоит слово «упрощенный»). Точнее реализация простая, математика сложная. Полученная программа может показывать хорошую точность (Как повезет, алгоритм сильно рандомизирован), однако по производительности уступает сильно.

Также в ходе выполнения лабораторной была реализована модель полиномиальной регрессии, однако ничего особенного она из себя не предсталяет. При увеличении степени MSE падает, однако и количество вычислений растет очень быстро, к тому же при больших степенях начиная с некоторой решение перестает аппроксимировать данные.