«НИТУ МИСиС»

Институт ИТАСУ
Кафедра инженерной кибернетики
Направление подготовки: 09.04.01 «Прикладная информатика»
Квалификация (степень): магистр
Группа: МПИ-20-4-2

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

на тему: Линейный классификатор и однослойный перцептрон

Студент: Кирсанов Г.В.

Москва 2021

Содержание

1 Введение

В рамках первой лабораторной работы были применены линейный классификатор и однослойный перцептрон для многоклассовой классификации.

2 Особенности реализации

Для проведения лабораторной работы использовались:

- язык программирования **Python3.8**;
- среда разработки **PyCharm;**
- библиотека визуализации **matplotlib**;
- библиотека машинного обучения **sklearn**;
- библиотека многомерных массивов **numpy**;
- библиотека работы с массивами данных **pandas**;
- стандартный модуль аннотации типов **typing**;
- стандартный модуль работы с входными аргументами **argparse**.

В качестве линейного классификатора был выбран класс библиотеки **sklearn** — SGDClassifier, для однослойного перцептрона — Perceptron.

Все модули, опирающиейся на случайные значения, зафиксированы seeed'ом 42. Также, для упрощения работы с кодом были написаны классы обёртки **Dataset** и **Fitter**. Первый нужен для того, чтобы в функции одним аргументом передавать признаки и ответы и чтобы иметь удобные именованные поля для доступа к ним; второй класс необходим для взаимодействия с первым и унификации кода.

3 Результаты

3.1 Двумерный датасет, количество классов 5

Датасет был сгенерирован с помощью функции *make_ blobs* библиотеки **sklearn**. Пример использования можно посмотреть в директории code/dataset_generator.py. Входные параметры:

1. --n_samples (значение по умолчанию 1000) — количество примеров;
2. --n_features (значение по умолчанию 2) — количество признаков;
3. --classes (значение по умолчанию 5) — количество классов;
4. --shuffle (значение по умолчанию *true*) — нужно ли перетасовать примеры;
5. --random_state (значение по умолчанию 42) — seed для генерации.

Результат генерации со значениями по умолчанию приведён на рис. 2.

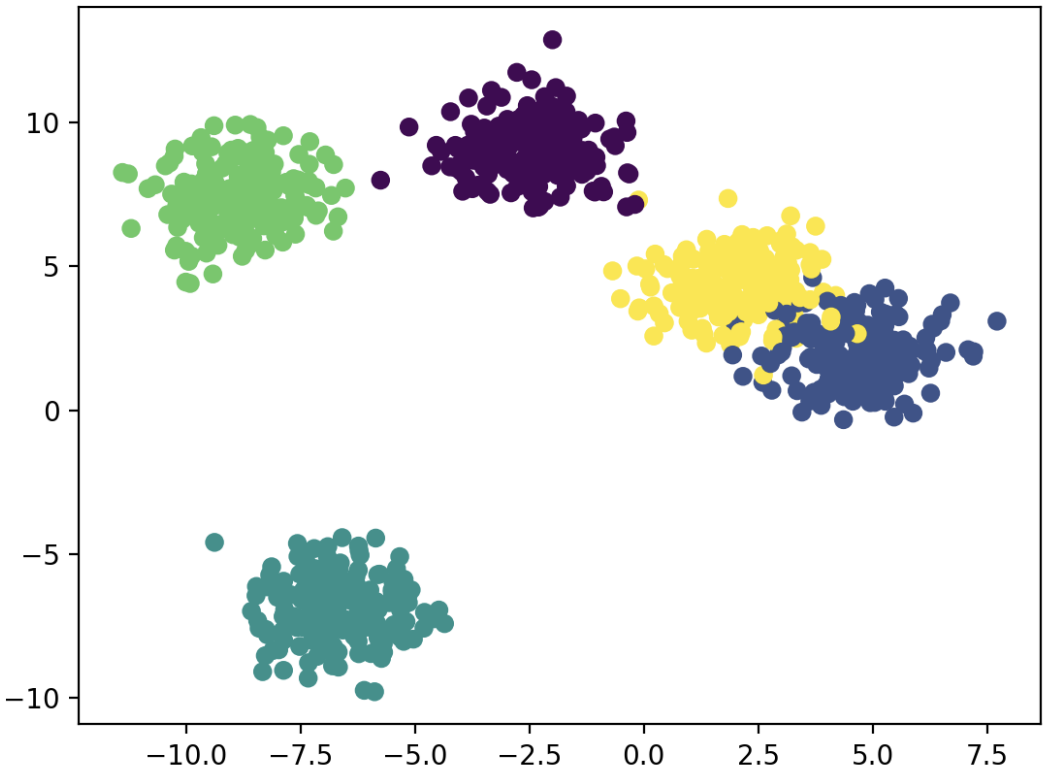


Рис. 1: Сгенерированный набор данных

Алгоритм работы кода в *code/main.py*:

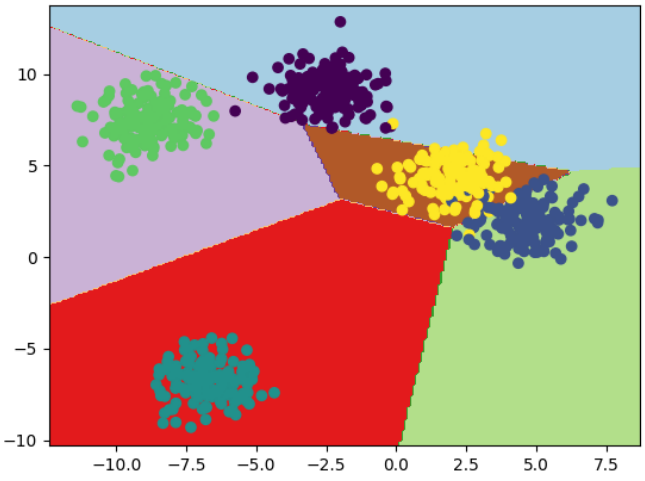
1. Создаётся по экзмпляру класса Perceptron и SGDClassifier;
2. Считывается датасет из файла *generated_ dataset.csv*;
3. Запускается обучение модели;

4. Рисуется *Decision surface*;

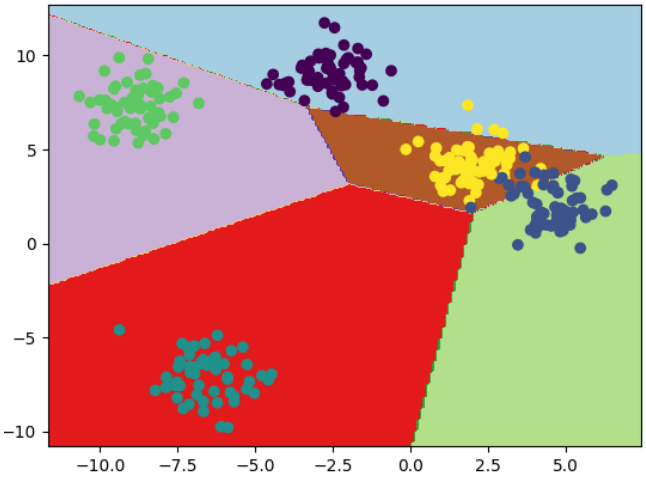
5. Рисуется набор данных;

6. В консоль выводятся метрики, достигнутые на тестовой выборке.

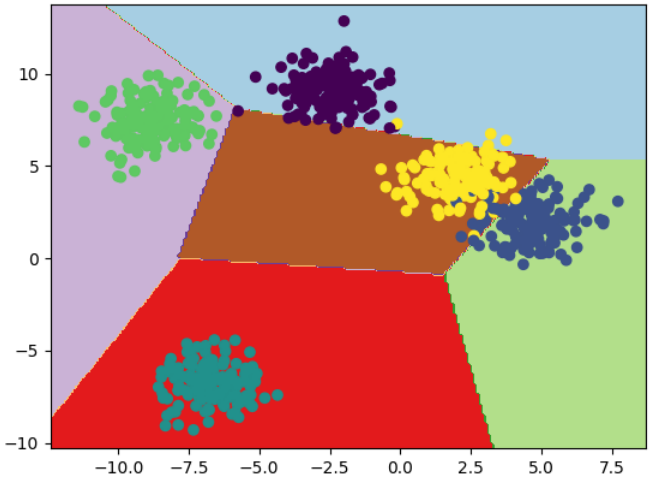
Следующие результаты были получены при инициализации класса перцептрона и линейного классификатора с дефолтными для библиотеки параметрами за исключением *random_state*=42, *n_jobs*=-1. Последний необходим для того, чтобы обучение использовало все доступные ядра CPU. Тестовая выборка составляет три десятых от всего набора данных.



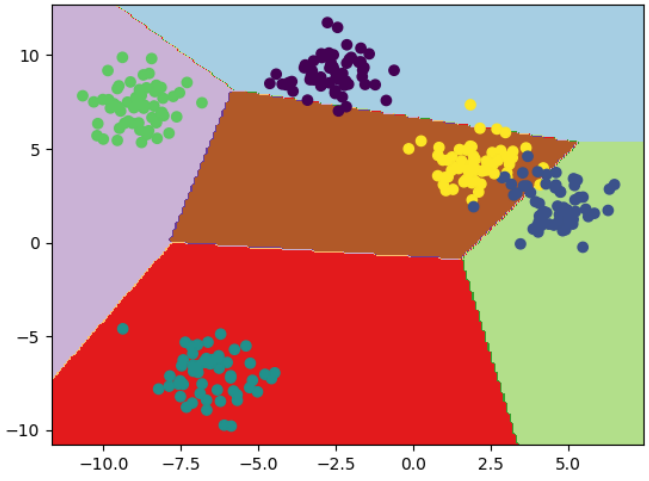
(a) Перцептрон: обучающая выборка



(b) Перцептрон: тестовая выборка



(c) Линейный классификатор: обучающий набор



(d) Линейный классификатор: тестовый набор

Рис. 2: *Decision surface* и выборки данных

Метрики на тестовом множестве показаны в таблице 1.

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.9433	0.9487	0.9458	0.9449
SGDClassifier	0.9500	0.9558	0.9520	0.9517

Таблица 1: Метрики на тестовом множестве

Статистика и картинки получены следующим скриптом:

```
python3.8 code/main.py
```

Так как однослойный перцептрон в своей основе имеет тот же принцип, что и линейный классификатор, попробуем добиться одинаковых результатов на обеих моделях.

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.8167	0.8324	0.8229	0.8009

Таблица 2: Перцептрону указана функция регуляризации l_2

Скрипт получения такого результата:

```
python3.8 code/main.py --penalty_perceptron l2
```

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.9767	0.9774	0.9776	0.9774

Таблица 3: Перцептрону добавлен флаг ранней остановки, если оценка на валидации перестаёт улучшаться

Скрипт получения такого результата:

```
python3.8 code/main.py --early_stopping_perceptron 1
```

Перцептрон с ранней остановкой перегнал по метрикам линейный классификатор из библиотеки **sklearn**. Попробуем улучшить его путём изменения входных параметров. Укажем флаг ранней остановки, который помог с перцептроном, и добавим знаки после запятой, чтобы лучше видеть картину при близости результатов:

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.9767	0.9774	0.9776	0.9774
SGDClassifier	0.9667	0.9696	0.9682	0.9678

Таблица 4: Линейному классификатору добавлен флаг ранней остановки.

Сейчас, судя по таблице 4 разница по метрикам около одного процента.

Скрипт получения такого результата:

```
python3.8 code/main.py --early_stopping_perceptron 1 --early_stopping_sgd 1
```

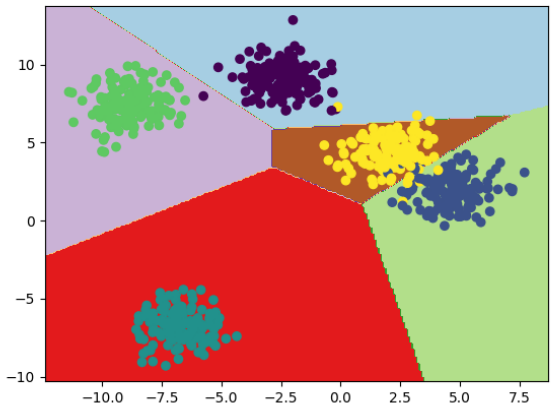
	accuracy	**precision**	**recall**	**f1**
Perceptron	0.9767	0.9767	0.9767	0.9767
SGDClassifier	0.9700	0.9714	0.9713	0.9710

Таблица 5: Линейному классификатору добавлен флаг ранней остановки и отключена функция регуляризации (по умолчанию включена и равна l_2).

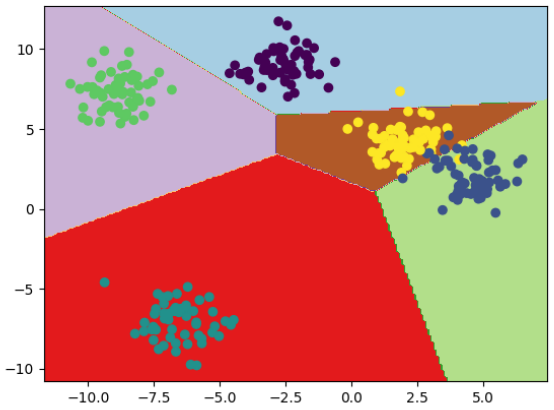
Это наименьшая разница в метриках, которую удалось добиться изменением параметров классов (без полного превращения линейного классификатора в перцептрон по параметрам). *Decision surface* при этом получились следующие (рис. 3).

Скрипт получения такого результата:

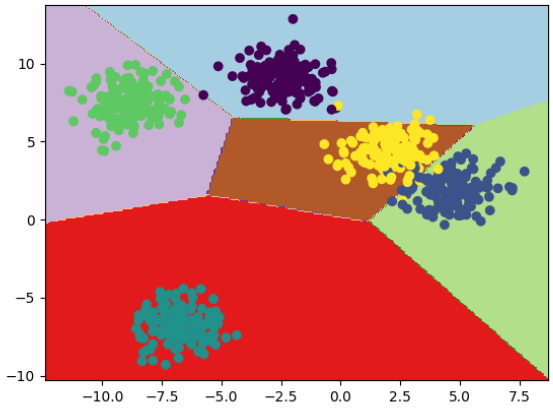
```
python3.8 code/main.py --early_stopping_perceptron 1 --early_stopping_sgd 1
--penalty_sgd None
```



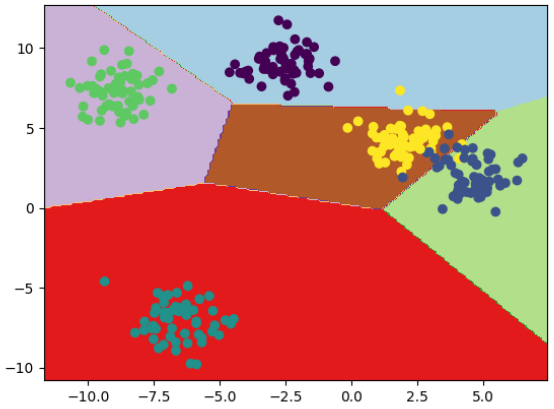
(a) Перцептрон: обучающая выборка



(b) Перцептрон: тестовая выборка



(c) Линейный классификатор: обучающий набор



(d) Линейный классификатор: тестовый набор

Рис. 3: *Decision surface* и выборки данных с новыми параметрами моделей

3.2 Авила датасет

Набор данных был получен из 800 изображений Библии Авилы, сделанная в 12 веке копия Библии на латыни. Цель набора — подготовить модель для определения букв по шаблону, чтобы помогать переписывающему. Классами в наборе данных являются 11 букв: A, B, C, D, E, F, G, H, I, W, X, Y.

Характеристика датасета:

- количество примеров — 20867;
- количество атрибутов — 10 (расстояние между колонками, верхний отступ, нижний отступ, использование, номер строки, модульное соотношение, интерлиньяж, вес, максимальное число, частное модульного соотношения и интерлиньяжа);
- проведена z-нормальзиация;
- пропущенных значений нет;
- несбалансирован;
- количество классов — 11.

Перед тем, как передавать датасет на обучение моделям, было проведено кодирование ответов: отсортированному множеству букв было присвоено число из натурального множества с нулём (A — 0, B — 1, . . . , Y — 11).

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.4496	0.2965	0.1992	0.2123
SGDClassifier	0.5462	0.3718	0.3728	0.3480

Таблица 6: «Чистый» запуск без параметров.

```
python3.8 code/main.py --enable_generated_dataset 0 --enable_archive_dataset 1
```

Воспользовавшись **GridSearchCV**, были получены параметры, при которых перцептрон прибавляет в каждой метрике:

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.4870	0.3095	0.2626	0.2678
SGDClassifier	0.5462	0.3718	0.3728	0.3480

Таблица 7: Запуск перцептрона с параметрами, подобранными **GridSearchCV**.

```
python3.8 code/main.py --enable_generated_dataset 0 --enable_archive_dataset 1 --alpha 1 --early_stopping_perceptron 1 --eta0 0.1 --tol 1 --validation_fraction 0.2
```

Попробуем зайти с другой стороны и изменим датасет. Оставим только те классы, количество примеров которых превышает тысячу: A(8572), E(2190), F(3923), H(1039), I(1663), X(1044).

Получаем следующие метрики качества обучения (таблица 8):

	accuracy	**precision**	**recall**	**f1**
Perceptron	0.5401	0.5062	0.4778	0.4859
SGDClassifier	0.6036	0.5821	0.4828	0.4986

Таблица 8: Метрики с урезанным набором данных.

Результат был достигнут следующим скриптом:

```
python3.8 code/main.py --enable_generated_dataset 0
--enable_archive_squeezed_dataset 1
```

Наблюдаем, что каждая метрика улучшилась на несколько процентов. Урезанный набор данных более сбалансирован, чем исходный. Дальнейшее увеличение значений метрик осложнено двумя факторами:

- невозможность закодировать ответы методом *one hot encoding* — реализация **sklearn** не позволяет моделям принимать эталонные ответы в виде векторов с размерностью не равной 1;
- линейность моделей — линейная неразделимость классов не позволяет добиться точных моделей только с линейными «разграничителями» классов (гиперплоскость).