Анализ Алгоритмов

Рубежный контроль №1

«Метод классификации Support Vector Machine»

Юмаев Артур Русланович

ИУ7-55

Преподаватель: Волкова Л.Л.

Оглавление

[Введение 3](#_Toc26375420)

[Постановка задачи 4](#_Toc26375421)

[1. Аналитическая часть 5](#_Toc26375422)

[Квадратичная оптимизация и поиск оптимальной гиперплоскости 6](#_Toc26375423)

[Проблема линейной разделимости классов 7](#_Toc26375424)

[Вывод 7](#_Toc26375425)

[3. Технологическая часть 8](#_Toc26375426)

[Вывод 10](#_Toc26375427)

[4. Исследовательская часть 11](#_Toc26375428)

[Вывод 12](#_Toc26375429)

[Литература 13](#_Toc26375430)

# Введение

В настоящее время сфера искусственного интеллекта проникла в нашу повседневную жизнь и все больше компаний и исследовательских центров применяют ИИ для решения разного рода задач. Классические методы ИИ были придуманы еще Тьюрингом в 40е годы. Позже появилась первая попытка воссоздать искусственный нейрон человека. Модель искусственного нейрона была предложена Розенблатом в 50х года. Тогда Розенблат еще не знал, насколько сильно это изменит наш мир. Также в 19 веке были изобретены первые методы классификации, такие как линейный дискриминантный анализ Фишера. В 1992 году Вапником [1] была предложена модель классификации данных, которая была названа методом опорных векторов (SVM – Support Vector Machine). Она могла обучаться на некотором наборе данных и классифицировать его на 2 класса. Далее этот метод был доработан для произвольного количества классов. Сейчас машина опорных векторов активно используется для классификации сложных многомерных данных.

# Постановка задачи

Цель: изучение метода классификации SVM.

Задачи:

* реализовать метод;
* протестировать метод;
* привести листинг на одном из языков программирования.

# Аналитическая часть

Машина опорных векторов – это линейная система, идея которой состоит в построении гиперплоскости, выступающей в качестве поверхности решений, максимально разделяющей положительные и отрицательные примеры. В частности, алгоритм настройки опорных векторов можно использовать для построения следующих трех типов обучаемых машин.

* Полиномиальные обучаемые машины.
* Сети на основе радиальных базисных функций.
* Двухслойные персептроны.

Рассмотрим множество обучающих примеров

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

где – входной образ для примера i; – соответствующий ему желаемый отклик (целевой выход). Предположим, что классы +1 и -1, составляющие множество d линейно разделимы. Уравнение поверхности решений в форме гиперплоскости, выполяющей это разделение, записывается следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

где **x –** входной вектор; **w** – настраиваемый вектор весов; *b* – порог. Таким образом можно записать:

|  |  |  |
| --- | --- | --- |
|  | , | (3) |

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Оптимальный вектор весовых коэффициентов обозначим как , а оптимальный порог обозначим как

## Квадратичная оптимизация и поиск оптимальной гиперплоскости

Задачу условной оптимизации можно решить с помощью *метода множителей Лагранжа* (Method of Langrange multipliers) [2].

Построим сначала функцию Лагранжа:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

где дополнительные неотрицательные элементы – множители Лагранжа.

Опуская некоторые преобразования запишем итоговую формулу для получения вектора весовых коэффициентов и порога.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

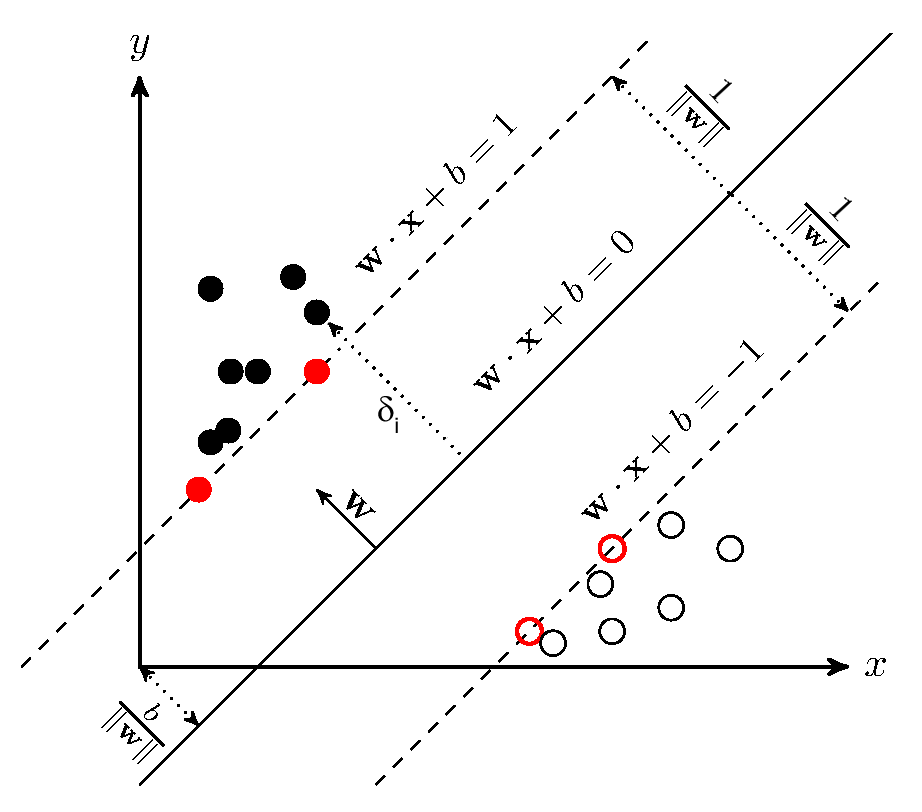


Рисунок 1. Визуализация оптимальной разделяющей гиперплоскости на плоскости

## Проблема линейной разделимости классов

В данной работе была реализована классическая машина опорных векторов. Классический подход требует от данных строгой линейной разделимости. Два множества точек в двумерном пространстве называются линейно сепарабельными (линейно разделимыми), если они могут быть полностью отделены единственной прямой. Для - мерного пространства два набора точек линейно разделимы, если они могут быть отделены - мерной гиперплоскостью.

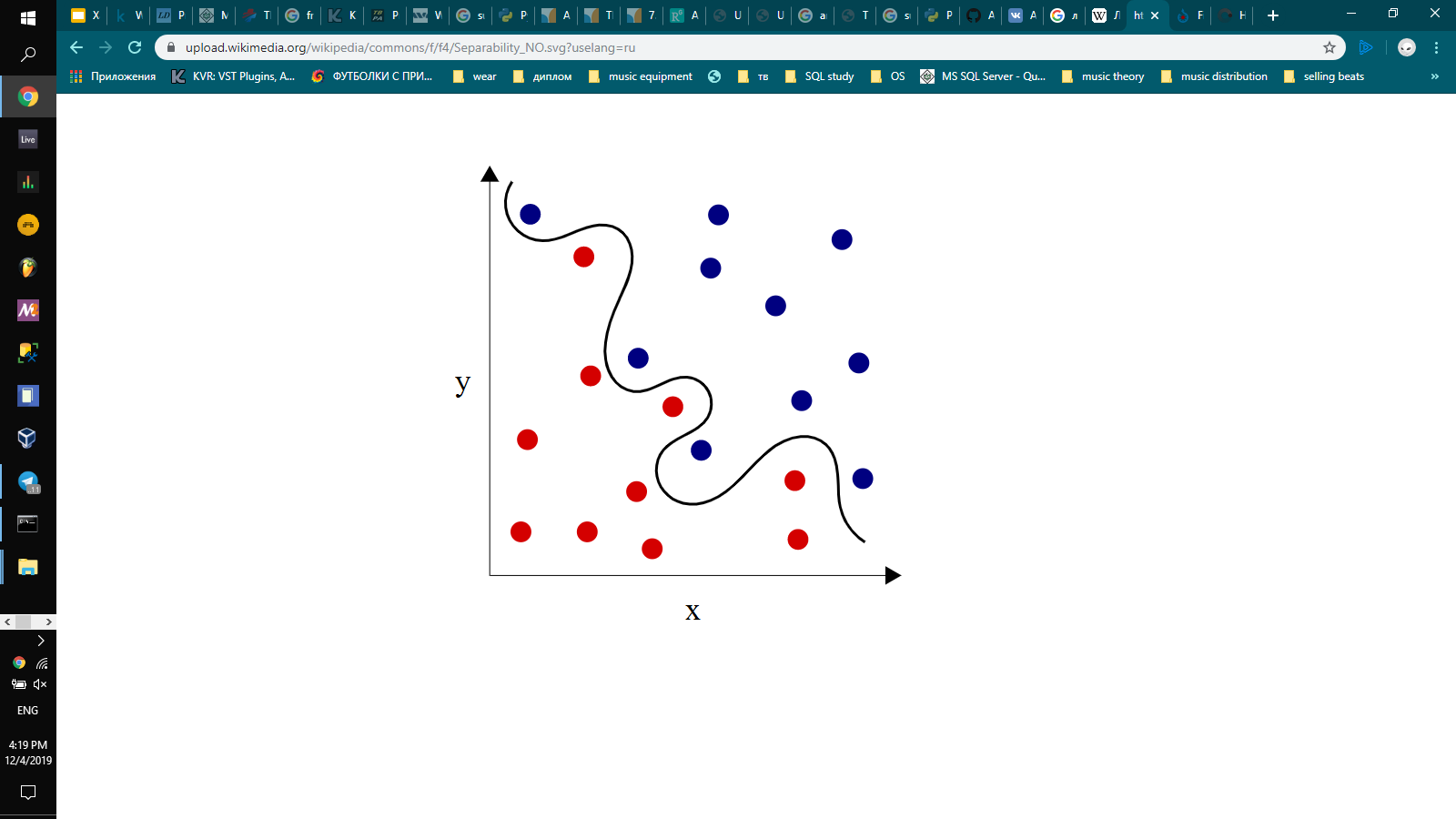


Рисунок 2. Данные линейно неразделимы

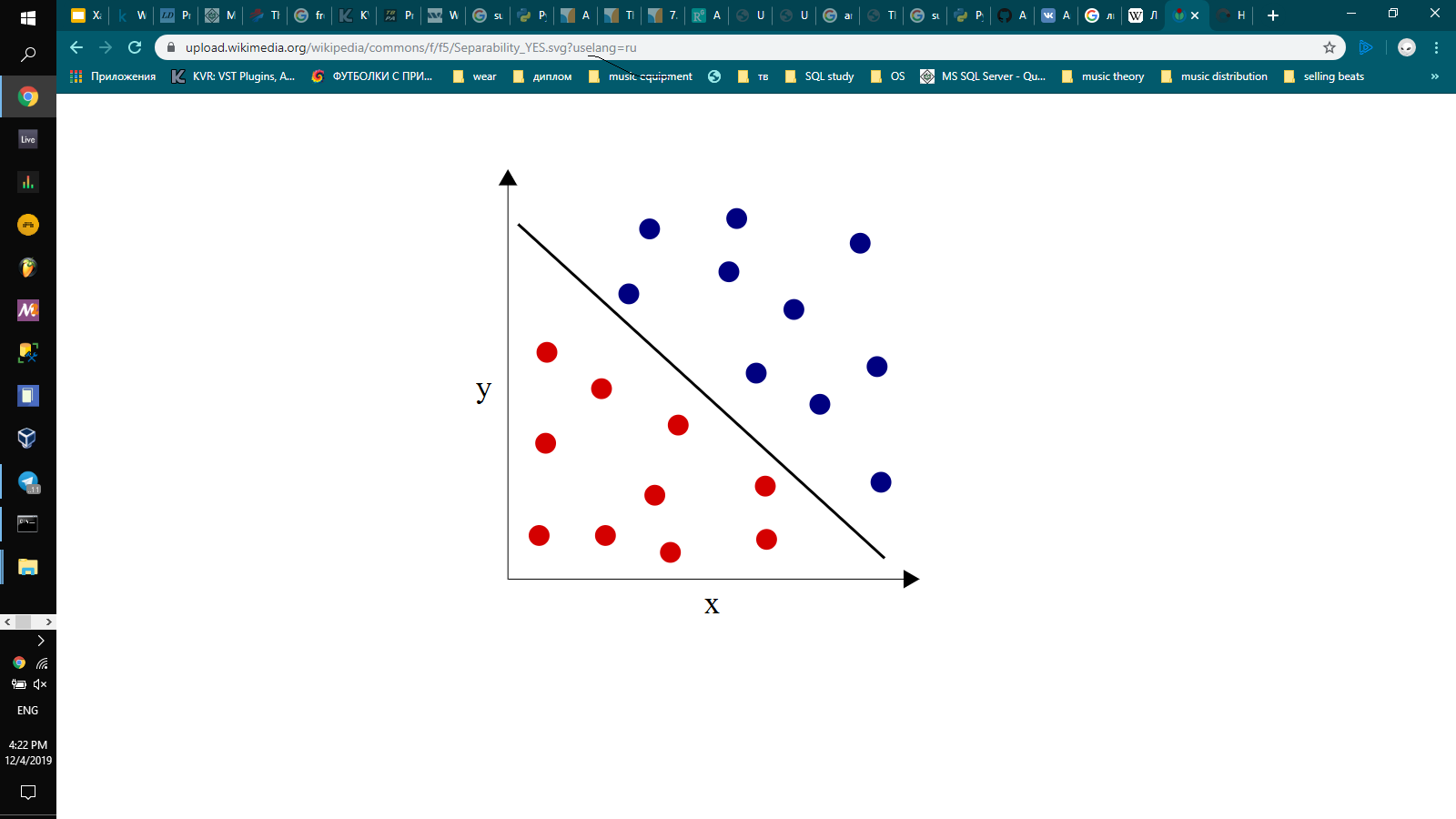


Рисунок 3. Данные линейно разделимы

## Вывод

Был проведен краткий математический обзор машины опорных векторов. Приведена целевая функция для оптимизации и поиска оптимальных значений весовых коэффициентов и порога. Была приведена проблема линейно разделимости.

# 3. Технологическая часть

В данном разделе будет приведен листинг метода опорых векторов на языке Python 3.7 [3] (листинг 1), так как имеется большой опыт работы с ним.

Листинг 1. Метод опорных векторов

|  |
| --- |
| **class** Support\_Vector\_Machine:  **def** \_\_init\_\_(self, visualization=**False**):  self.visualization = visualization  self.colors = {1:**'r'**,-1:**'b'**}  **if** self.visualization:  self.fig = plt.figure()  self.ax = self.fig.add\_subplot(1,1,1)  **def** fit(self, data):  self.data = data  self.linear\_devision\_state = **True** *# The number of features for each sample.* self.n\_features = len(data[list(data.keys())[0]][0])    *# { ||w||: [w,b] }* opt\_dict = {}   arrays = [(-1, 1) **for** \_ **in** range(self.n\_features)]  transforms = np.array(list(product(\*arrays)))   all\_data = []  **for** yi **in** self.data:  **for** featureset **in** self.data[yi]:  **for** feature **in** featureset:  all\_data.append(feature)   self.max\_feature\_value = max(all\_data)  self.min\_feature\_value = min(all\_data)  all\_data = **None**step\_sizes = [self.max\_feature\_value \* 0.1,  self.max\_feature\_value \* 0.01,self.max\_feature\_value \* 0.001,  ] b\_range\_multiple = 2b\_multiple = 5  latest\_optimum = self.max\_feature\_value\*10    **for** step **in** step\_sizes:  w = np.array([latest\_optimum,latest\_optimum])optimized = **False  while not** optimized:  **for** b **in** np.arange(-1\*(self.max\_feature\_value\*b\_range\_multiple),  self.max\_feature\_value\*b\_range\_multiple,  step\*b\_multiple):  **for** transformation **in** transforms:  w\_t = w\*transformation  found\_option = **True****for** i **in** self.data:  **for** xi **in** self.data[i]:  yi=i  **if not** yi\*(np.dot(w\_t,xi)+b) >= 1:  found\_option = **False****if** found\_option:  opt\_dict[np.linalg.norm(w\_t)] = [w\_t,b]   **if** w[0] < 0:  optimized = **True** print(**'Optimized a step.'**)  **else**:  w = w - step   norms = sorted([n **for** n **in** opt\_dict])  **if** len(norms) == 0:  self.linear\_devision\_state = **False  break  else**:  *#||w|| : [w,b]* opt\_choice = opt\_dict[norms[0]]  self.w = opt\_choice[0]  self.b = opt\_choice[1]  latest\_optimum = opt\_choice[0][0] + step \* 2   **if not** self.linear\_devision\_state:  print(**"Невозможно построить SVM, так как классы линейно неразделимы."**)  print(**"В данных есть точки, отклоняющиеся от идеального \**  **состояния линейной разделимости."**)  **return** -1   **for** i **in** self.data:  **for** xi **in** self.data[i]:  yi = i**def** predict(self, features):  *# sign(x\*w + b)* classification = np.sign(np.dot(np.array(features), self.w) + self.b)  **if** classification != 0 **and** self.visualization:  self.ax.scatter(features[0],  features[1],  s=200,  marker=**'\*'**,  c=self.colors[classification])  **return** classification |

## Вывод

В данном разделе был приведен код алгоритма бинарной классификации методом SVM на языке программирования Python.

# 4. Исследовательская часть

В качестве исследования и тестирования машины опорных векторов были подготовлены тестовые данные, которые являются точками в двумерном пространстве. В таблице 1 приведены данные и принадлежность каждой точки к классу -1 или 1.

Таблица 1

Тестовые точки

|  |  |  |
| --- | --- | --- |
| x | y | Класс |
| 2 | -14 | -1 |
| -14 | 3 | -1 |
| -20 | 6 | -1 |
| -11 | 7.5 | -1 |
| -4 | 22 | 1 |
| 16 | 6 | 1 |
| 6 | -5 | 1 |
| 5 | -1 | 1 |

На рисунке 4 приведена визуализация работы машины опорных векторов. Красным цветом отмечены элементы, принадлежащие к классу 1. Синим цветом отмечены элементы, принадлежащие к классу -1. Звездочками помечены элементы, на которых тестировалась модель. Как видно визуально, машина опорных векторов смогла безошибочно классифицировать данные. Именно поэтому в классическом подходе не воодится понятие ошибки модели, так как классы должны быть линейно разделимы заранее.

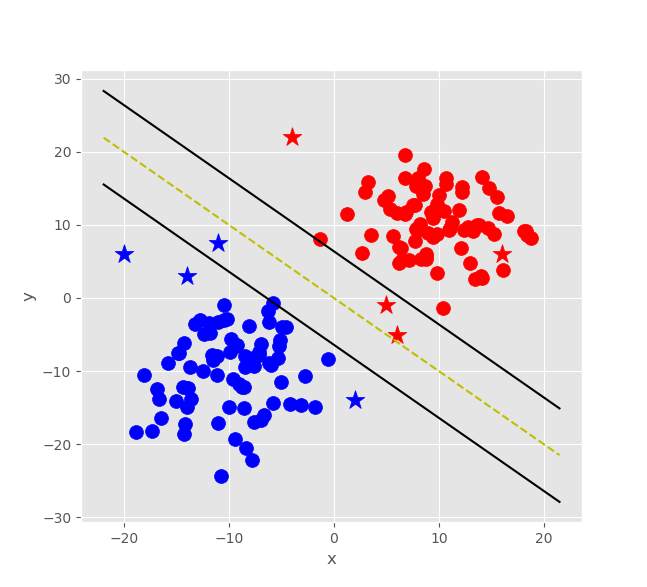


Рисунок 4. Тестирование SVM

На основе данного примера можно определить опорные вектора для обоих классов.

Таблица 2

Опорные вектора

|  |  |  |
| --- | --- | --- |
| x | y | Класс |
| -5.81 | -0.73 | -1 |
| -1.3 | 8.12 | 1 |

Предсказание класса строится на основе линейной комбинации точки для тестирования, вектора весов и смещения как:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

На основе данных, на которых обучалась модель также были вычислены вектор весов и смещеные, данном случае они равны соответственно:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |
|  |  |  |

## Вывод

В данном разделе было проведено тестированик реализованной машины опорных векторов, приведена оценка точности, найдены опорные элементы и значения, необходимые для предсказания класса на новых данных.

# Литература

[1] Boser B., I. Guyon and V.N. Vapnik. “A training algorithm for optimal margin classifiers”, Fifth Annual Workshop on Computational Learning Theory, 1992, p. 144-152. San Mateo, CA: Morgan Kauffman.

[2] Bertsekas D.P. Dynamic Programming and Optimal Control, 1995, vol. I and vol. II, Belmont, MA: Athenas Scientific.