**Федеральное государственное образовательное**

**бюджетное учреждение**

**высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ**

**ФЕДЕРАЦИИ»**

**(Финансовый университет)**

**Факультет**

**информационных технологий и анализа больших данных**

**Направление «Прикладная математика и информатика»**

**Домашнее задание № 8**

«Методы стохастической оптимизации»

Студенты группы ПМ19-3:

Захаров Д. В.

Исмоилова М. В.

Константинов К. Л.

Мосолова К. Д.

Самофалова Т. А.

Руководитель:

Аксенов Дмитрий Андреевич

**Москва 2022**

**Оглавление**

[1. Физическая модель 3](#_Toc106396214)

[2. Математическая модель 3](#_Toc106396215)

[3. Алгоритмы 10](#_Toc106396216)

[4. Варианты использования 13](#_Toc106396217)

[5. Архитектура решения 15](#_Toc106396218)

[6. Тестирование 16](#_Toc106396219)

[7. Заключение 17](#_Toc106396220)

# **Физическая модель**

Стартапу «Рога и копыта» необходимо классифицировать акции на основе волатильности. Будут ли они сильно или слабо изменяться в течение определенного периода. Для этого стартап решил находить минимумы и максимумы определенных аналитиками функций, по которым меняется цена и передавать это в алгоритм классификации. Для решения задачи моделирования функции было предложено два варианта:

1. Использование метода имитации отжига
2. Использование генетического алгоритма.

Для решения задачи классификации на два класса было предложено использовать метод опорных векторов SVM с применением алгоритма градиентного спуска для минимизации функции ошибок (PEGASOS algorithm).  
Функции были написаны и протестированы. Было замерено время их выполнения. Алгоритмы поиска экстремума сравнены между собой. Также предложено дальнейшее развитие проекта.

# **Математическая модель**

*Градиентный спуск* — это метод оптимизации, который может найти минимум целевой функции. Это жадный метод, который находит оптимальное решение, делая шаг в направлении максимальной скорости уменьшения функции.

Название методу подарил тот факт из математического анализа, что вектор частных производных функции задает направление наискорейшего возрастания этой функции. Значит, двигаясь в сторону антиградиента функции, можно уменьшать значения этой функции быстрее всего.

*Стохастический градиентный спуск* отличается от обычного (пакетного) тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного (или нескольких), случайно выбранного элемента. Он широко используется для минимизации функции ошибки в задаче классификации методом опорных векторов (SVM).

Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

Рассмотрим задачу классификации на два класса. Имеется обучающая выборка , где – вектор признаков для объекта n, а – его метка класса. Задача заключается в предсказании метки класса для объекта, представленного своим вектором признаков . (1)

Гиперплоскость, определяемая вектором весов и смещением b может быть описана с помощью формулы: . Произведенная классификация считается правильной, когда веса, умноженные вектор меток класса, больше 1. Это исходит из того, что метка класса может быть «1» или «-1», а значит:

* в случае выборки с истинным классом «1» скалярное произведение весов и выборки должно быть положительным, а положительное значение, умноженное на 1, является положительным.
* В случае выборки с истинным классом «-1» скалярное произведение весов и выборки должно быть отрицательным, а отрицательное значение, умноженное на -1, является положительным.

Таким образом, если истинный класс, умноженный на веса, отмеченные точкой выборки, положителен, SVM правильно классифицировал выборку.

Оптимизация весов происходит по формуле:

Здесь – это параметры, причем (3), а обычно берут равной 0.001.

Для достижения наилучшей сходимости рекомендуется также нормировать веса: . (4)

Градиентный спуск также можно использовать для нахождения минимума функции.

Пусть имеется задача:

(5)

Из-за ограничений использовать градиентный спуск напрямую невозможно: даже есть начальная точка будет принадлежать допустимой области X, нельзя гарантировать, что найденная точка на очередном шаге спуска будет также принадлежать допустимому множеству. Поэтому для решения задачи (5) применяют *метод проекции градиента*, суть которого заключается в проектировании новой точки на область Х.

Рассмотрим задачу оптимизации при единственном линейном ограничении в виде равенства:

(6)

*.*

В заданной точке , в которой ▽f()≠0, делается попытка найти направление поиска, которое бы лежало на поверхности ограничения и являлось направлением спуска. Такое направление можно получить, геометрически ортогонально проектируя вектор, противоположный ▽f() на поверхность ограничения (см. рис.1).

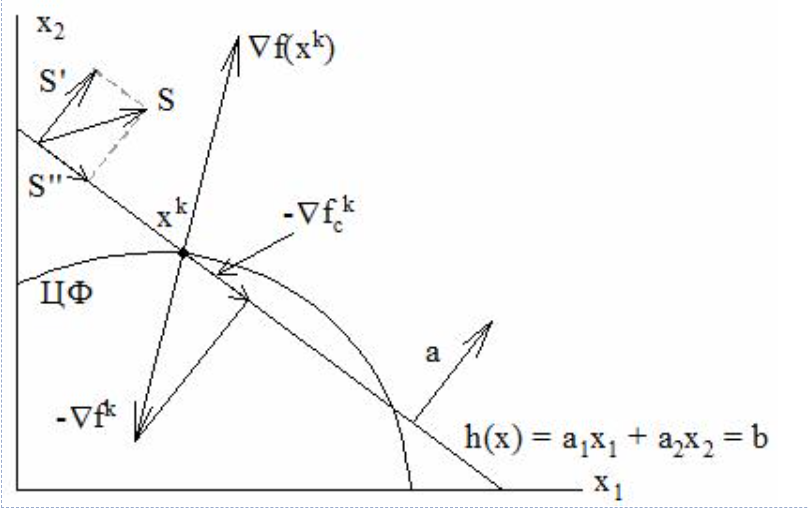


Рисунок 1

Здесь ▽ - проекция антиградиента на поверхность ограничений, которая приводит в допустимые точки. Процесс ортогонального проектирования состоит в разложении вектора на две ортогональные компоненты: параллельную поверхности, заданной ограничением, и перпендикулярную к ней. Параллельная компонента является искомой проекцией градиента. Пусть вектор a - нормаль к поверхности ограничения. При этом из выражения следует допустимость направления, задаваемого вектором S (S параллельно поверхности).

Т. о. все векторы, перпендикулярные к поверхности ограничения, должны быть параллельны к a. Следовательно, для любого вектора S его компонента S’, перпендикулярная к поверхности ограничения, равняется значению a, умноженному на константу.

Обозначим через S” компоненту S, параллельную поверхности ограничения. Тогда S” удовлетворяет соотношению (7)

Значит, любой вектор можно представить в виде векторной суммы:

S=S′+S″, (8)

где S′=λa, а S” удовлетворяет уравнению (7).

Найдем λ. Рассмотрим скалярное произведение . В силу (7) и (8) имеем:

(9)

Откуда (10)

Из (8) найдем S″=S-S′=S- λa. Подставим сюда (10), получим

(11)

где I - единичная матрица, порядок которой согласован с S.

Матрица (12) - проекционная матрица. Она проектирует вектор S на плоскость, задаваемую ограничением h(x).

Пусть - направление спуска. (

Если S” = 0, то точка удовлетворяет необходимым условиям Лагранжа. Вектор множителей Лагранжа задается выражением (14)

существует только в том случае, если значения ak линейно независимы. Если имеются линейно зависимые ограничения, их следует исключить из рассмотрения.

Длину шага h, используемую для вычисления новой точки в обычном градиентном спуске (15), можно найти, решив задачу одномерного поиска:

(16)

В свою очередь выполняет роль ограничителя, не дающего выйти за область допустимых значений, и вычисляется по формуле , j=1, …m. (17)

Здесь были рассмотрены ограничение в виде линейных равенств, однако метод применим и в случае линейных неравенств. Для этого с помощью получаемой по формуле (14) оценки множителей Лагранжа осуществляется поочередное исключение ограничений из множества активных ограничений.

В заданной точке для определения активного множества проверяются ограничения в виде неравенств: , j=1..m.

Составляется матрица ограничений из строк, соответствующих активным ограничениям. Вычисляется проекционный оператор P и проекция . Если ||≤ ε, то вычисляем множители Лагранжа (14). Если все λ≥0, то решение найдено. В противном случае ограничение с наибольшим по модулю множителем Лагранжа исключается из множества активных ограничений.

Еще одним стохастическим методом, позволяющим решить задачу (5), является метод имитации отжига (simulated annealing). За основу взят процесс кристаллизации вещества, оптимизацией которого интересовались металлурги для повышения однородности металла. У каждого металла есть кристаллическая решетка, которая описывает геометрическое положение атомов вещества. Совокупность позиций всех атомов будем называть состоянием системы, каждому состоянию соответствует определенный уровень энергии. Цель отжига – привести систему в состояние с наименьшей энергией. Чем ниже уровень энергии, тем «лучше» кристаллическая решетка, т.е. тем меньше у нее дефектов и прочнее металл. В ходе «отжига» металл сначала нагревают до некоторой температуры, что заставляет атомы кристаллической решетки покинуть свои позиции. Затем начинается медленное и контролируемое охлаждение. Атомы стремятся попасть в состояние с меньшей энергией, однако, с определенной вероятностью они могут перейти и в состояние с большей. Эта вероятность уменьшается вместе с температурой. Переход в худшее состояние, как ни странно, помогает в итоге отыскать состояние с энергией меньшей, чем начальная. Процесс завершается, когда температура падает до заранее заданного значения.

Целевая функция из задачи (5) – это функция энергии E. Функция изменения температуры T с течением времени обязательно должна убывать (поэтому ее также называют функцией «охлаждения»), и обычно (18), где i - это номер итерации. Новое состояние можно получить с помощью формулы из обычного градиентного спуска (15), только в этот раз h – это случайное число из нормального распределения. Если энергия «кандидата» меньше, он становится новым состоянием, в противном случае, переход будет вероятностным (поэтому метод относят к классу стохастических). Вероятность перехода в новое состояние можно вычислить по формуле: . (19)

Четвертый способ – генетический алгоритм – также был «подчерпнут» из законов природы, только в этот раз не физических, а биологических. Суть метода заключается в том, что мы модулируем эволюционный процесс: у нас есть какая-то популяция (набор векторов), которая размножается, на которую воздействуют мутации и производится естественный отбор на основании минимизации целевой функции.

Основной принцип размножения — потомок похож на своих родителей. Т.е. мы должны задать какой-то механизм наследования, который будет включать элемент случайности. Но скорость развития таких систем очень низкая — генетическое разнообразие падает, популяция вырождается (в нашем случае - значение функции перестает минимизироваться). Для решения этой проблемы был введен механизм мутации, который заключается в случайном изменении каких-то особей. Следующий важный механизм — селекция. Селекция — отбор особей (не обязательно из только что «родившихся»), которые лучше минимизируют функцию.

При этом стоит отметить, что метод имитации отжига и генетический алгоритм дают наилучшие результаты, когда имеется точка-ориентир, к которой мы стремимся. В противном случае оба метода дают неточные результаты, однако их можно использовать для нахождения начальных приближений в других методах.

# **Алгоритмы**

Алгоритм 1. Поиск минимума функции с ограничениями с помощью метода проекции градиента (задача 5).

1. Ограничения проверяются на нелинейность (; проверяется, что начальная точка удовлетворяет ограничениям.
2. В точке находят активные ограничения , такие, что .
3. Рассчитываются по формулам (12) и (13) соответственно. Если активных ограничений нет, очевидно, что .
4. Если |S(k=0)|≤ɛ, точка может оказаться оптимальной. Переход к шагу 5. Если неравенство не выполняется, то переход к шагу 7.
5. Находятся множители Лагранжа (14). Если все множители неотрицательные, точка – оптимальная. Если же нет, то переходим к шагу 6.
6. Находим наибольшее по модулю λi и вычеркиваем соответствующее ограничение из рассмотрения. Переход в пункт 3.
7. Происходит пересчет точки. Для этого находится максимальный шаг (18), h (17) и сама точка (16). С новой точкой возвращаемся в пункт 2.

Алгоритм 2. Классификация на два класса методом опорных векторов SVM с применением алгоритма градиентного спуска для минимизации функции ошибок (PEGASOS algorithm) (задача 1).

1. Инициализируем начальные веса , задается T итераций, лямбда. На каждом шаге t ≤T:
2. Обучающая выборка перемешивается, чтобы избежать зацикливания. Выбирается k случайных строк.
3. Рассчитывается (4).
4. Оценивается точность предсказания, и с учетом этого происходит обновление весов (2).
5. Веса нормируются по формуле (4).
6. Когда t > T, обновление весов заканчивается. По тестовой выборке делается предсказание.

Алгоритм 3. Метод имитации отжига. (задача 5)

1. Задается начальная точка , удовлетворяющая ограничениям, и количество итераций T (обычно T = 1000). Пока i≤T:
2. Высчитывается «температура» на шаге I по формуле (18).
3. По формуле (15) рассчитывается новая точка , причем h – случайное число из нормального распределения. Если точка не удовлетворяет ограничениям, расчет выполняется, пока не найдётся такое h, что .
4. Если значение целевой функции в новой точке меньше, чем в предыдущей, происходит переход в новое состояние. Иначе рассчитывается вероятность перехода по формуле (19). Если P>0.9, происходит переход в новое состояние. Иначе возвращаемся в пункт 3.

По окончанию итераций получаем оптимальную точку минимума.

Алгоритм 4. Генетический алгоритм (задача 5).

Года четыре назад, в универе услышал о таком методе оптимизации, как генетический алгоритм. О нем везде сообщалось ровно два факта: он клёвый и он не работает.

[*Генетический алгоритм — наглядная реализация (Хабр)*](https://habr.com/ru/post/254759/)

1. Из равномерного распределения от -10 до 10 создается начальная популяция (набор точек/генов), удовлетворяющая ограничениям.
2. Оценивается «полезность» каждого гена – рассчитывается значение целевой функции в каждой точке из популяции.
3. Отбирается два «родителя» с наилучшими генами – точки, значение целевой функции в которых минимально. На основе этих точек создаются «дети», которые являются всеми возможными комбинации координат точек с сохранением их позиции в векторе.

Например, если «родители» - это точки [1, 2] и [3, 4], то их «дети» - точки [1, 2], [1, 4], [3, 2], [3, 4].

«Дети» добавляются в популяцию, при этом перед добавлением популяция сокращается вдвое – убираются точки, значение целевой функции в которых слишком большое.

1. Некоторые гены (и только созданные, и уже существующие) мутируют: случайно отобранные точки умножаются на параметр strength\_of\_mutation, отвечающий за силу регуляризации.
2. Точки, не удовлетворяющие ограничениям, убираются из популяции.

По достижении количества итераций выдается точка, значение целевой функции в которой минимально.

# **Варианты использования**

ВИ 1

Пользователь хочет решить задачу классификации на два класса при помощи метода опорных векторов SVM с применением алгоритма градиентного спуска для минимизации функции ошибок (PEGASOS algorithm).

1. Пользователь использует функцию own\_pegasos. В качестве обязательных параметров необходимо передать:
2. Np.array тренировочной выборки X\_train
3. Np.array классов для тренировочной выборки y\_train (обязательно значения 1 и -1)
4. Np.array тестовой выборки, по которым будет возвращена классификация, X\_test
5. Если пользователю необходимо увеличить силу регуляризации, он может варьировать параметр lam (по умолчанию 0.001)
6. Если пользователю необходимо изменить максимальное количество итераций, он должен варьировать параметр max\_iter (по умолчанию 1000)
7. Если пользователю необходимо изменить объем тренировочной выборки, по которой будет совершаться градиентный спуск, он должен варьировать параметр stoch\_size (по умолчанию 0.2)
8. Если пользователю необходимо построить график классификации, он должен изменить флаг draw на True (работает только для датасетов с двумя признаками)
9. В качестве выходного параметра пользователь получает словарь со следующими ключами:
10. Y\_pred – массив из -1 и 1 (предсказанные классы)
11. W – вектор весов модели

Также строится график, если пользователь поставил draw = false и кол-во признаков равно двум. В противном случае, если значение draw True, а кол-во признаков двум не равно, возбуждается ошибка.

ВИ2 пользователь хочет найти экстремум методом имитации отжига.

1. Пользователь использует функцию simulated\_annealing

В качестве основных параметров следует передать

1. функцию в явном виде
2. Constraints – список ограничений, также в виде строк
3. start\_points – начальная точка
4. Если пользователь хочет изменить кол-во итераций, он меняет параметр steps (по умолчанию 1000)
5. В качестве выходного параметра функция возвращает список, точку экстремума.

ВИ3 пользователь хочет найти экстремум функции при помощи генетического алгоритма

1. Пользователь использует функцию genetic\_algorithm
2. В качестве обязательных параметров передаются
3. Функция в явном виде
4. Список ограничений в явном виде
5. Если пользователь хочет изменить силу мутации, ему следует изменять параметр strength\_of\_mutation (по умолчанию 1)
6. Если пользователь хочет изменить кол-во итераций, ему следует изменить параметр number\_of\_generations (по умолчанию 20)
7. В качестве выходного параметра функция возвращает список, точку экстремума.

# **Архитектура решения**

simulated\_annealing – Функция, решающая задачу поиска экстремума методом имитации отжига

Параметры:

F: str

Функция в виде строки

Constraints: list

Список строк с ограничениями

start\_points: list

количество итераций

genetic\_algorithm – функция, решающая задачу поиска экстремума функции при помощи генетического алгоритма.

Параметры:

func : string

функция для оптимизации

restr : list

список строк с ограничениями

strength\_of\_mutation : float, default = 1

Если значение отличается от 1, то некоторое количество генов

будет умножено на указанное значение.

number\_of\_generations : int, default = 20

Количество итераций.

# **Тестирование**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № теста | Метод имитации отжига | Генетический алгоритм ([x2, x1]) | Верный результат | Время ИО | Время ГО |
| 1 | [5.74652414070998, -0.986096562839933] | [-9.981331746884363, -3.360411304214468] | [6, 0] | 105 ms | 2.09 ms |
| 2 | [1.00136001295672, 1.00136001295672] | [0.4078756287087373, 0.7824401766622664] | [1, 1] | 62.7 ms | 95.1 ms |
| 3 | [1.74900892193083, 0.875495539034583] | [1.1033564408942897, 2.0545451920083746] | [2, 1] | 70.9 ms | 1.36 s |
| 4 | [0.750147574128691, 3.52501475741287] | [4.047155714463395, 0.7444795430051379] | [1, 4] | 53.8 ms | 29.8 ms |
| 5 | [-12.7773723196601, -2.25912410655338] | [-9.591602953322534, -9.987976554208345] | [0, 0] | 26.9 ms | 32.2 ms |
| 6 | [0.566797983370339, 3.13359596674068] | [2.5643836982212562, 1.5791526138196215] | [1, 3] | 65.5 ms | 23.9 ms |
| 7 | [0.667587366876424, 2.33241263312357] | [3.0678584809129834, 1.0926700107094796] | [1, 3] | 43.8 ms | 53.2 ms |
| 8 | [2.48197716773578, 0.248197716773578] | [0.935244835020514, 3.542117176462705] | [2, 0] | 93.6 ms | 331 ms |
| 9 | [3.42006455281632, 0.946709701877546] | [1.112712515168024, 3.0315631112102963] | [3, 1] | 67.8 ms | 61 ms |
| 10 | [-1.49914697418616, 7.49914697418617] | [9.732284018537623, -9.939351010823287] | [0, 7] | 48 ms | 34 ms |

Учитывая среднее время выполнения и точность ответа, наилучшим будет использование метода имитации отжига, но при использовании генетического алгоритма следует учесть, что координаты точки выдаются в обратном порядке!

**Модель классификации на два класса методом опорных векторов SVM с применением алгоритма градиентного спуска для минимизации функции ошибок.**

Первые два теста были проведены на датасете с сильной корреляцией признаков. Последние два – искуственно сгенерированные.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер датасета (кол-во наблюдений, колво пр-ков) | Время выполнения | % верно предсказанных | Коэффициенты регрессии |
| (40, 3) | 6min 43s | 100 | [-1.47322675, -0.37594501, 27.71473319, 7.12179126] |
| (40, 2) | 3min 22s | 100 | [-0.29288166, 0.76992113, 44.2531261 ] |
| (400, 3) | 2min 9s | 100 | [ 5.23764493e-02, -1.20159284e-01, -8.55493830e-02, 1.58096721e-17] |
| (900, 2) | 58.5 s | 9926 | [ 1.39792868, -0.100957 , 0.36646816] |

# **Заключение**

Таким образом, исходя из всего вышеперечисленного, в стохастической оптимизации лучше использовать метод имитации отжига, при этом, возможно есть смысл посмотреть алгоритмы нестохастической оптимизации. Для данной задачи они могут отработать быстрее и точнее.