Лекция 4. Линейная классификация. Логистическая регрессия. Метод опорных векторов.

Глинский А.В.

Цель занятия: В этом модуле мы познакомимся с такими темами, как линейная классификация, логистическая регрессия, а также с метод опорных векторов. После прохождения модуля ученики смогут понимать основные принципы работы классификаторов в задачах машинного обучения и применять различные классификаторы для решения задач классификации.

Содержание

1 Линейная классификация **3**
- 1.1 Сравнение задач регрессии и классификации . . . 3
- 1.2 Трансформация линейной регрессии в линейную классификацию . . . 3
- 1.3 Бинарная классификация . . . 4
  - 1.3.1 Геометрическая интерпретация бинарной классификации . . . 4
  - 1.3.2 Отступ модели на объекте . . . 5
  - 1.3.3 Функции потерь в бинарной классификации . . . 6
- 1.4 Многоклассовая классификация . . . 8
- 1.5 Вопросы для самопроверки . . . 12
- 1.6 Резюме по разделу . . . 13

2 Логистическая регрессия **14**
- 2.1 Визуальная демонстрация алгоритма . . . 14
- 2.2 Описание алгоритма . . . 14
- 2.3 Подготовка данных для алгоритма . . . 16
- 2.4 Процесс обучения . . . 16
- 2.5 Оценка качества алгоритма . . . 17
- 2.6 Интерпретация признаков с помощью алгоритма . . . 18
- 2.7 Модификации алгоритма . . . 18
- 2.8 Область применения алгоритма . . . 18
- 2.9 Плюсы и минусы алгоритма . . . 19
- 2.10 Реализация алгоритма в Python . . . 19
- 2.11 Вопросы для самопроверки . . . 20
- 2.12 Резюме по разделу . . . 21

3 Метод опорных векторов **22**
- 3.1 Визуальная демонстрация алгоритма . . . 22
- 3.2 Описание алгоритма . . . 22
- 3.3 Подготовка данных для алгоритма . . . 25

3.4 Процесс обучения 25
3.5 Оценка качества алгоритма 26
3.6 Интерпретация признаков с помощью алгоритма 26
3.7 Модификации алгоритма 27
3.8 Область применения алгоритма 27
3.9 Плюсы и минусы алгоритма 28
3.10 Реализация алгоритма в Python 29
3.11 Вопросы для самопроверки 29
3.12 Резюме по разделу 30

4 Резюме по модулю **30**

1 Линейная классификация

Ранее мы рассматривали задачу регрессии. Теперь давайте поговорим о задаче классификации. Для начала проведем сравнение этих подходов.

1.1 Сравнение задач регрессии и классификации

Задачи регрессии и классификации отличаются по своей природе, целям и используемым методам.

Задача регрессии заключается в предсказании непрерывного значения целевой переменной на основе входных признаков. То есть, мы строим модель, которая может предсказывать, например, цену на недвижимость, уровень дохода или количество продаж в зависимости от различных факторов. Методы, используемые для решения задач регрессии, включают метод k ближайших соседей, линейную регрессию, решающие деревья, случайный лес, градиентный бустинг и нейронные сети.

Задача классификации, с другой стороны, заключается в предсказании дискретных значений целевой переменной или вероятностей классов на основе входных признаков. То есть, мы строим модель, которая может предсказывать, например, является ли электронное письмо спамом или не спамом, какой класс имеет изображение, какое слово было произнесено на записи и т.д. Методы, используемые для решения задач классификации, включают метод k ближайших соседей, логистическую регрессию, метод опорных векторов, решающие деревья, случайный лес, градиентный бустинг и нейронные сети.

Главным отличием между задачами регрессии и классификации является тип целевой переменной: непрерывный для регрессии и дискретный для классификации. Это приводит к различным метрикам оценки модели и методам её построения. Например, для задач регрессии обычно используются метрики, такие как средняя абсолютная ошибка (MAE), среднеквадратическая ошибка (MSE), а для задач классификации — точность, полнота, F1-мера и др.

1.2 Трансформация линейной регрессии в линейную классификацию

Главной проблемой перенастройки модели линейной регрессии на задачу классификации является то, что линейная регрессия выдает непрерывное вещественное число, в то время как классификация осуществляется для дискретного набора классов.

Для преобразования модели линейной регрессии в модель линейной классификации можно использовать функцию активации, которая преобразует непрерывный выход модели в дискретное значение класса или вероятность класса. Среди таких функций — сигмоидальная функция и функция softmax. Мы изучим их далее в этом модуле. В простейшем случае модель классификации возвращает только два значения, которые обычно называют положительным и отрицательным классом. Такую классификацию называют бинарной. Давайте разберемся, как она устроена.

1.3 Бинарная классификация

Бинарная классификация - это задача машинного обучения, в которой требуется отнести объекты к одному из двух классов на основе набора признаков. Другими словами, бинарная классификация решает задачу разделения объектов на две группы, где каждая группа соответствует одному из двух классов.

Примерами бинарной классификации могут служить определение, является ли электронное письмо спамом или не спамом, прогнозирование, выздоровеет ли пациент после операции или нет, определение, является ли транзакция мошеннической или нет и т.д.

Одной из наиболее распространенных функций активации для бинарной классификации является функция порога, которая возвращает 1, если выход модели больше определенного порога (threshold), и 0 или -1 в противном случае. Это можно выразить следующей формулой (Формула 1):

$$y = \begin{cases} 1 & \text{если } w \cdot x + b \geq \text{threshold} \\ 0 \text{ или } -1 & \text{иначе} \end{cases} \tag{1}$$

Мы для удобства будем пользоваться метками классов «+1» и «-1». В таком случае, для линейной классификации модель по сути предсказывает знак скалярного произведения вектора весов на вектор признаков объекта.

1.3.1 Геометрическая интерпретация бинарной классификации

Давайте визуализируем разделение классов «+1» и «-1» с помощью гиперплоскости. Гиперплоскость - это n-мерное обобщение понятия прямой или плоскости в пространстве большей (n+1) размерности. Она определяется уравнением, которое состоит из суммы произведений коэффициентов на переменные, равной некоторой константе. (Рисунок 1). Для полу-

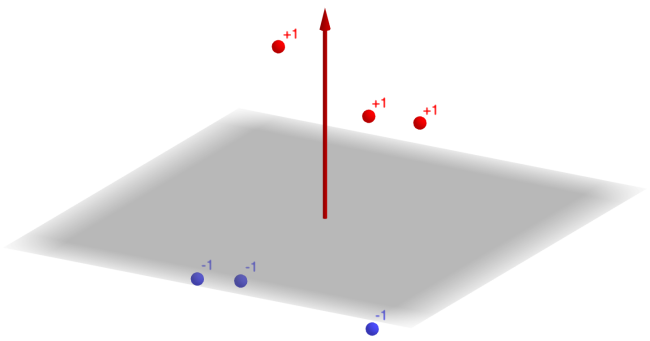


Рисунок 1: Классификация с помощью гиперплоскости

чения разделяющей гиперплоскости в задаче линейной классификации можно использовать вектор нормали. Этот вектор задает гиперплоскость, перпендикулярен ей и определяется коэффициентами модели линейной классификации. Вектор нормали направлен в сторону положительного класса. В общем случае вектор нормали можно задать по формуле 2.

$$\mathbf{w} = (w_1, w_2, \ldots, w_d) \tag{2}$$

где d - число признаков, $w_1, w_2, \ldots, w_d$ - коэффициенты модели.

Для получения единичного вектора нормали $\mathbf{w}$ его необходимо нормировать на его длину (Формула 3).

$$\mathbf{w_{norm}} = \frac{\mathbf{w}}{||\mathbf{w}||} \tag{3}$$

Таким образом, если мы знаем коэффициенты модели линейной классификации, то мы можем легко получить вектор нормали, который можно использовать для построения разделяющей гиперплоскости. Если скалярное произведение вектора признаков объекта на вектор весов модели $x_i \cdot w \geq 0$, модель отнесет объект к положительному классу, иначе — к отрицательному

В случае двух признаков модель будет использовать линию в качестве разделяющей классифицирующей поверхности (Рисунок 2).

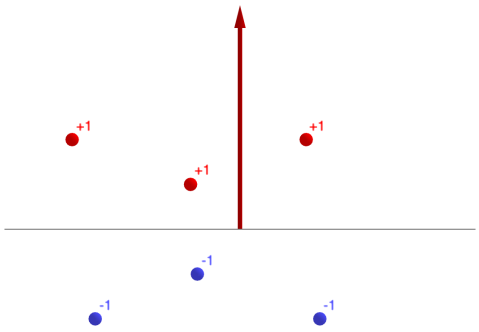


Рисунок 2: Линия в качестве разделяющей классифицирующей поверхности

Подытоживая вышесказанное, можно сказать, что простейший бинарный классификатор можно построить с помощью пороговой функции, учитывая только знак скалярного произведения вектора признаков объекта на вектор весов объекта.

1.3.2 Отступ модели на объекте

На рисунке 2 мы можем заметить, что полезно учитывать не только знак скалярного произведения, но и расстояние объекта от разделяющей гиперплоскости. Для учета этого расстояния существует мера, называемая отступом.

Отступ (англ. margin) в линейной классификации - это мера того, насколько «уверена» модель в своем прогнозе для конкретного объекта. Он определяется как произведение предсказанной моделью метки класса на расстояние от объекта до гиперплоскости (или прямой в случае двумерной линейной классификации), разделяющей классы.

Формально, для объекта с признаками x и меткой y отступ можно выразить следующим образом (Формула 4).

$$M(x_i, y_i) = y_i * x_i \cdot w$$
$$\text{где } x_i \cdot w \text{ — скалярное произведение вектора признаков объекта на вектор весов,} \tag{4}$$
$$y_i \text{ — метка класса (-1 или +1).}$$

Таким образом, если модель правильно предсказала метку класса для объекта и расстояние от него до гиперплоскости большое (то есть объект находится далеко от гиперплоскости),

то отступ будет большим и модель будет «уверена» в своем предсказании. Если же объект находится близко к гиперплоскости, то отступ будет маленьким и модель будет менее «уверена» в своем предсказании. Если модель неправильно предсказала метку класса для объекта и отступ большой, возможно, модель работает неправильно или данный объект является выбросом (Рисунок 3).

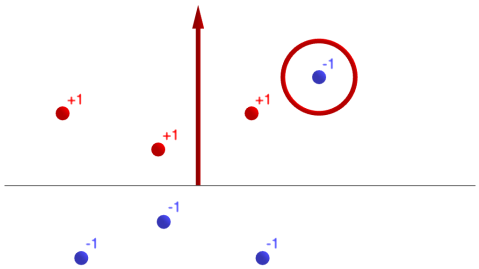


Рисунок 3: Выброс с большим отступом

1.3.3 Функции потерь в бинарной классификации

Функции потерь (или функции ошибки) в задаче бинарной классификации используются для измерения расхождения между предсказанными и реальными метками классов. Они применяются в процессе оптимизации модели, когда мы пытаемся найти такие значения параметров модели, которые минимизируют ошибку.

Рассмотрим некоторые функции потерь в бинарной классификации.

Бинарная функция потерь с индикатором определяется следующим образом (Формула 5).

$$
\begin{gathered}
L(y, \hat{y}) = \mathbb{1}(y \neq \hat{y}), \\
\text{где } y \text{ - истинный класс объекта (-1 или 1)}, \\
\hat{y} \text{ - предсказанный класс объекта (-1 или 1)}, \\
\mathbb{1} \text{ - индикаторная функция, которая равна 1, если ее аргумент истинен,} \\
\text{и 0 в противном случае.}
\end{gathered} \tag{5}
$$

Функция потерь равна 1, если предсказанный класс $\hat{y}$ не соответствует истинному классу y, и равна 0 в противном случае. Минимизация бинарной функции потерь с индикатором в задаче бинарной классификации является задачей оптимизации. Однако, такая функция потерь обычно не используется, потому что она не дифференцируема и не может быть использована в градиентных методах оптимизации, которые требуют вычисления градиента функции потерь по параметрам модели.

Для того, чтобы получить дифференцируемую функцию потерь, бинарная функция потерь с индикатором должна быть преобразована. Можно в индикаторную функцию потерь в

качестве аргумента передавать отступ на объекте, сравнивая его с нулем (Формула 6).

$$L(M(x,y)) = \mathbb{1}(M(x,y) < 0),$$
где $M(x,y)$ - отступ на объекте,
$\mathbb{1}$ - индикаторная функция, которая равна 1, если ее аргумент истинен, (6)
и 0 в противном случае.

В этом случае получается пороговая функция потерь. Такая функция по-прежнему недифференцируема. Нам необходимо сделать еще один шаг. Можно добавить некоторую другую дифференцируемую функцию потерь $\tilde{L}(M(x,y))$, которая будет верхней оценкой для $L(M(x,y))$ с расчетом на то, что оптимизация $\tilde{L}(M(x,y))$ приведет также к оптимизации $L(M(x,y))$. Для $\tilde{L}(M(x,y))$ применяются различные функции потерь:

- Логистическая функция потерь. Логистическая функция потерь используется в задачах классификации для оценки ошибки модели. Функция определяется следующим образом: (Формула 7).

  $$L(M) = log(1 + e^{-M}),$$
  где M - это отступ на объекте $(y\hat{y})$. (7)

  Функция потерь принимает на вход отрицательные значения M, и возрастает монотонно к 0 при $M \to -\infty$, и к log(2) при $M \to +\infty$. Это означает, что чем более уверенной является модель в отнесении объекта к одному из классов, тем меньше будет значение функции потерь.
- Функция потерь Хинджа. Для задач классификации, где $y \in -1{,}1$ - истинная метка класса, а $\hat{y}$ - предсказанная метка класса, функция потерь Хинджа может быть выражена следующим образом: (Формула 8).

  $$L(y,\hat{y}) = \max(0, 1 - M)$$
  где max - функция, возвращающая максимальное значение из двух аргументов, (8)
  M - это отступ на объекте $(y\hat{y})$.

  Эта функция возвращает 0, если предсказание верно $(y\hat{y} \geq 1)$, иначе возвращает значение, пропорциональное расстоянию между предсказанием и истинной меткой класса. Чем больше расстояние между ними, тем больше значение функции потерь. Можно заметить, что $y\hat{y}$ является отступом на объекте (M).
- Логарифмическая (Бинарная кросс-энтропия). Логарифмическая функция потерь и бинарная кросс-энтропия имеют одинаковую математическую формулу, но разные интерпретации. Различие в трактовке этих понятий заключается в разных контекстах. Логарифмическая функция потерь измеряет расхождение между фактическим распределением классов и распределением, предсказанным моделью. Бинарная кросс-энтропия измеряет расхождение между фактическим и предсказанным распределениями вероятностей для двух классов. Название «кросс-энтропия» происходит от того, что эта функция потерь измеряет расхождение между двумя вероятностными распределениями. «Кросс» здесь означает «перекрестный», так как мы сравниваем два распределения. «Бинарная» здесь означает, что мы сравниваем распределения для двух классов.

- Для двух бинарных векторов y и $\hat{y}$, бинарная кросс-энтропия может быть выражена следующим образом (Формула 9):

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{d}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)] \tag{9}$$

где d - количество элементов в векторах y и $\hat{y}$,

y_i и $\hat{y}_i$ - элементы векторов y и $\hat{y}$ соответственно.

Графики пороговой и логистической функции, а также функции потерь Хинджа и логарифмической функции потерь (бинарной кросс-энтропии) приведены на рисунке (Рисунок 4).

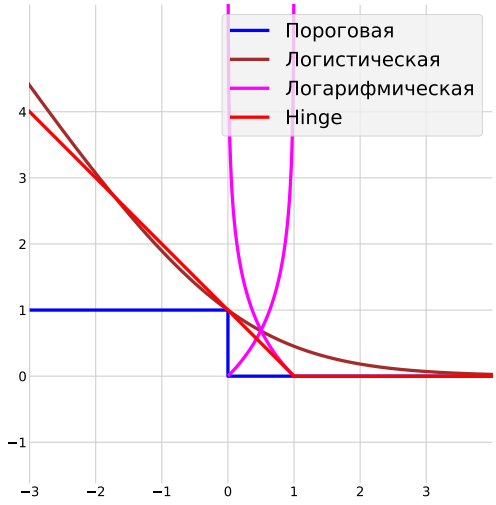


Рисунок 4: $\tilde{L}(M(x,y))$ функции потерь

1.4 Многоклассовая классификация

Также давайте рассмотрим задачи многоклассовой классификации, когда классов больше, чем два. Многоклассовая классификация - это задача классификации, где необходимо отнести объекты к одному из нескольких классов. Например, можно классифицировать изображения по видам животных (собаки, кошки, птицы и т.д.), рукописные цифры (от 0 до 9) или типы цветов.

Существует несколько подходов к решению задачи многоклассовой классификации, включая:

- Один против всех (One-vs-All) - обучение отдельной модели для каждого класса, где каждая модель разделяет этот класс от остальных. При классификации нового объекта используются все модели, и объект относится к классу, для которого модель выдала наивысшую вероятность.

- Один против другого (One-vs-One) - обучение модели для каждой пары классов, где каждая модель разделяет два класса. При классификации нового объекта каждая модель голосует за класс, к которому относится объект, и объект относится к классу, который набрал наибольшее количество голосов.
- Многоклассовая логистическая регрессия (Multinomial Logistic Regression) - обучение единственной модели, которая предсказывает вероятности принадлежности объекта к каждому классу. Для обучения модели используется перекрестная энтропия (cross-entropy) — функция потерь, которая минимизирует ошибку классификации.

Рассмотрим эти подходы подробнее.

Один против всех (One-vs-All) - это подход к решению задачи многоклассовой классификации, который заключается в обучении отдельной модели для каждого класса, где каждая модель разделяет этот класс от остальных (Рисунок 5).

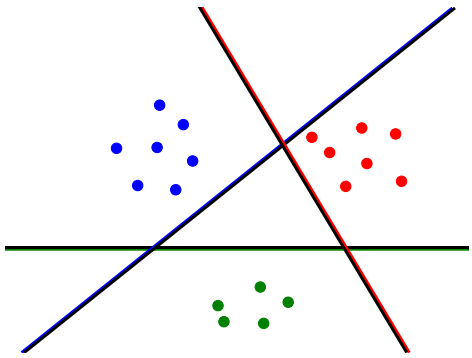


Рисунок 5: Подход Один против всех (One-vs-All)

Например, представим, что мы решаем задачу классификации изображений на 3 класса: кошки, собаки и птицы. В методе один против всех мы будем обучать 3 модели - для каждого класса отдельно. То есть, модель для кошек будет учиться разделять кошек от всех других животных, модель для собак - разделять собак от всех других животных, и модель для птиц - разделять птиц от всех других животных.

При классификации нового изображения мы будем применять все 3 модели и выбирать тот класс, для которого модель выдала наивысшую вероятность. Например, если модель для кошек выдала вероятность 0.6, а модель для собак - 0.3 и для птиц - 0.1, то мы отнесем изображение к классу кошек.

Этот подход удобен, так как мы можем использовать любой классификатор, который может работать с двумя классами. Также он может быть эффективным в случаях, когда классов много, а обучающие выборки малы. Однако он может столкнуться с проблемой, если классы имеют пересекающиеся области, так как каждая модель учится только разделять свой класс от остальных, и не учитывает отношения между остальными классами.

В библиотеке scikit-learn реализован метод One-vs-All (Один против всех) для решения задач многоклассовой классификации. Он используется в моделях, которые не поддерживают прямую многоклассовую классификацию, например, в методе логистической регрессии.

В scikit-learn реализация One-vs-All выполняется автоматически при использовании метода логистической регрессии с параметром multi_class='ovr' (один против всех) или при

использовании метода Support Vector Machine (SVM) с параметром decision_function_shape='ovr'.

Один против другого (One-vs-One) - это подход к решению задачи многоклассовой классификации, который заключается в обучении модели для каждой пары классов, где каждая модель разделяет два класса (Рисунок 6).

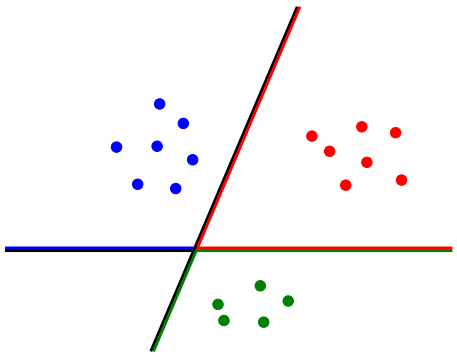


Рисунок 6: Подход Один против другого (One-vs-One)

Например, представим, что мы решаем задачу классификации изображений на 3 класса: кошки, собаки и птицы. В методе один против другого мы будем обучать 3 модели - одну для каждой пары классов. То есть, модель для кошек и собак будет учиться разделять кошек от собак, модель для кошек и птиц - разделять кошек от птиц, и модель для собак и птиц - разделять собак от птиц.

При классификации нового изображения мы будем применять все 3 модели и выбирать тот класс, к которому относится изображение, с учетом голосования моделей. Например, если изображение было отнесено к классу кошек 2 раза (модель для кошек и собак и модель для кошек и птиц), а к классу собак - 1 раз (модель для кошек и собак), то мы отнесем изображение к классу кошек.

Этот подход имеет преимущество в том, что он более точен, чем подход один против всех, потому что каждая модель учитывает отношения только между двумя классами. Однако, для классификации изображения требуется выполнение большего количества вычислений, так как нам нужно обучать больше моделей и выполнять больше предсказаний.

В библиотеке scikit-learn метод One-vs-One также реализован для решения задач многоклассовой классификации. В отличие от метода One-vs-All, в котором обучается один классификатор для каждого класса, метод One-vs-One обучает классификатор для каждой пары классов. Для использования метода One-vs-One в scikit-learn можно использовать классификаторы, которые поддерживают прямую многоклассовую классификацию, например, методы Support Vector Machine (SVM, decision_function_shape='ovo') и Random Forest (multi_class='ovo').

Многоклассовая логистическая регрессия (Multinomial Logistic Regression) - это метод многоклассовой классификации, который является обобщением бинарной логистической регрессии на случай, когда число классов больше двух.

В этом методе мы моделируем вероятность принадлежности объекта к каждому классу при помощи функции softmax. Функция softmax (софтмакс) - это функция, которая преобразует вектор произвольных действительных чисел в вектор вероятностей, где сумма всех

вероятностей равна 1. Эта функция широко используется в машинном обучении, особенно в задачах многоклассовой классификации, где требуется присвоить каждому объекту один из нескольких классов.

Формула для функции softmax (Формула 10):

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \tag{10}$$

где z_i - элемент входного вектора z, а n - количество элементов вектора z.

Таким образом, для каждого элемента z_i входного вектора z, функция softmax вычисляет экспоненту этого элемента, а затем нормирует все экспоненты путем их деления на сумму экспонент всех элементов вектора z. Результатом является вектор вероятностей, где каждый элемент соответствует вероятности принадлежности объекта к соответствующему классу.

Пример использования функции softmax: предположим, что у нас есть модель, которая должна классифицировать изображения на три класса: кошки, собаки и птицы. Модель возвращает три числа, каждое из которых соответствует вероятности принадлежности изображения к соответствующему классу. Затем мы применяем функцию softmax к этому вектору, чтобы получить вероятности для каждого класса.

Например, если модель возвращает вектор [1.5, 2.3, 0.8], то после применения функции softmax мы получим вектор вероятностей [0.217, 0.536, 0.247]. Это означает, что модель считает, что изображение наиболее вероятно принадлежит классу собак (Рисунок 7).

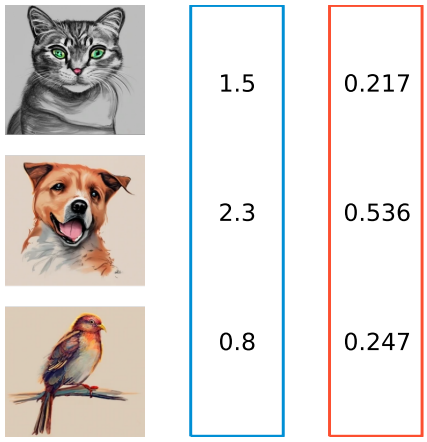


Рисунок 7: Демонстрация работы softmax

Модель многоклассовой логистической регрессии определяется набором параметров, включающих в себя веса признаков и векторы смещения для каждого класса. Обучение модели заключается в нахождении наилучших параметров, минимизируя функцию потерь, такую как кросс-энтропийная функция потерь (cross-entropy loss). Если у нас есть C классов, и y_i обозначает истинную метку класса для i-го примера в обучающем наборе, а $p_{i,j}$ обозначает предсказанную вероятность, что i-й пример принадлежит к классу j. Тогда кросс-энтропийная

функция потерь вычисляется следующим образом (Формула 11):

$$L(w) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{i,j}\log(p_{i,j}), \tag{11}$$

где w - это параметры модели, n - размер обучающего набора.

Каждое слагаемое в сумме соответствует ошибке для одного примера и одного класса, и общая функция потерь является средним значением всех таких ошибок.

В кросс-энтропийной функции потерь для многоклассовой классификации мы используем метки классов в форме one-hot encoding. То есть, если пример относится к классу k, то $y_{i,k} = 1$, а для всех остальных классов $y_{i,j} = 0$. Функция потерь штрафует модель за ошибки в предсказании вероятностей для всех классов, а не только для истинного класса. Она пытается минимизировать расстояние между предсказанными вероятностями и one-hot кодировкой истинных меток классов.

После обучения модели мы можем классифицировать новые объекты, вычислив вероятности для каждого класса при помощи функции softmax и выбрав класс с наибольшей вероятностью.

В библиотеке scikit-learn для решения задач многоклассовой классификации по умолчанию используется алгоритм One-vs-All, который может использовать различные функции активации, включая softmax. Однако, по умолчанию в scikit-learn для задач многоклассовой классификации используется логистическая регрессия (LogisticRegression), которая использует softmax в качестве функции активации для расчета вероятности принадлежности каждого класса.

1.5 Вопросы для самопроверки

Для преобразования модели линейной регрессии в модель линейной классификации используется функция активации, которая преобразует непрерывный выход модели в дискретное значение класса или вероятность класса. Какая функция активации может использоваться для этой цели?

1. MSE
2. Сигмоидальная функция
3. Функция softmax
4. Функция tanh

Правильные ответы: 2, 3

В машинном обучении отступ (margin) является расстоянием от гиперплоскости до обучающего примера. Какой из следующих вариантов ответа лучше всего описывает важность отступа в задаче классификации?

1. Отступ не является важным для задачи классификации.
2. Чем больше отступ и знак отступа правильный, тем выше вероятность правильной классификации.
3. Чем меньше отступ и знак отступа правильный, тем выше вероятность правильной классификации.

4. Отступ не имеет никакого отношения к вероятности правильной классификации.

Правильные ответы: 2

Какие подходы применяют для реализации возможности многоклассовой классификации?

1. Градиентный спуск
2. One-vs-One
3. One-vs-All
4. Multinomial Logistic Regression

Правильные ответы: 2, 3, 4

1.6 Резюме по разделу

В задаче классификации необходимо предсказать дискретное значение класса или вероятность класса.

Бинарная классификация - это задача машинного обучения, в которой требуется отнести объекты к одному из двух классов на основе набора признаков.

Существует несколько подходов для реализации возможности многоклассовой классификации:

1. One-vs-One
2. One-vs-All
3. Multinomial Logistic Regression

2 Логистическая регрессия

Цель занятия: ученик может применить алгоритм логистической регрессии для решения задач классификации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Описание алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Интерпретация признаков с помощью алгоритма
- Модификации алгоритма
- Область применения алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

2.1 Визуальная демонстрация алгоритма

Логистическая регрессия - это метод машинного обучения, который используется для решения задач бинарной классификации, т.е. для разделения данных на два класса.

Классический пример задачи, который может быть решен с помощью логистической регрессии, это предсказание вероятности того, что клиент купит продукт, основываясь на истории его покупок и других данных.

Логистическая регрессия использует логистическую функцию (сигмоиду) для прогнозирования вероятности принадлежности объекта к одному из двух классов. Входные данные, обычно представленные вектором признаков, умножаются на веса, которые настраиваются во время обучения, и затем подаются на вход логистической функции. Результатом является вероятность принадлежности объекта к положительному классу. Визуальная демонстрация алгоритма представлена на иллюстрации 8.

Давайте разберемся с тем, что изображено на иллюстрации 8. На рисунке 8.1 приведена зависимость целевой переменной от признака как пример построения линейной регрессии с одним признаком. Такая зависимость аппроксимируется (приближается) линией. Далее мы вводим сигмоидальную активацию ($\sigma(x) = \frac{1}{1+e^{-x \cdot w}}$), чтобы сжать получившиеся значения в диапазон (0, 1) (Рисунок 8.2). Соответственно, применение сигмоиды к линейной комбинации весов и признаков позволяет спроецировать данные на сигмоиду (Рисунок 8.3) и сделать вывод о том, к какому классу принадлежит объект (Рисунок 8.4). В данном случае порог классификации для разделения классов установлен на уровне 0.5.

2.2 Описание алгоритма

Перейдем от визуального описания алгоритма логистической регрессии к формальному определению.

Логистическая регрессия - это метод машинного обучения, который используется для классификации данных. Он применяется для прогнозирования вероятности отнесения наблюдаемого объекта к определенному классу. Этот метод широко применяется в задачах бинарной классификации, когда требуется определить, принадлежит ли объект к одному из двух классов.

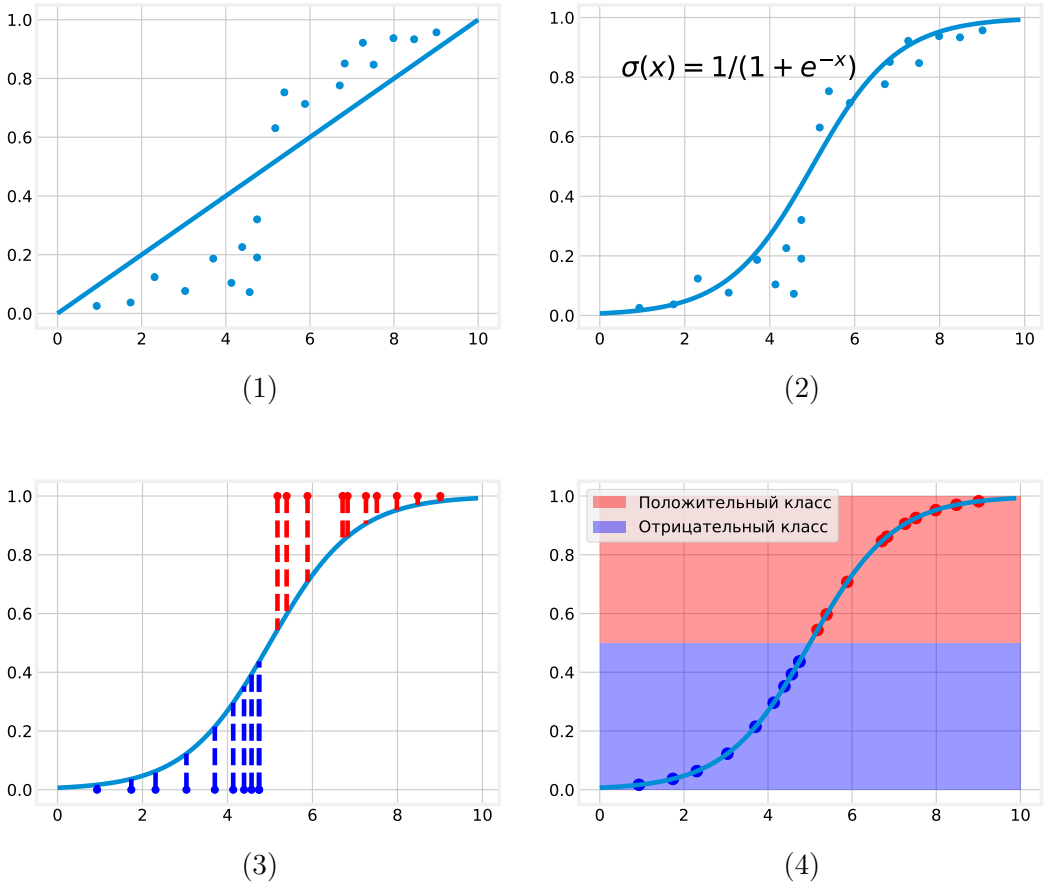


Рисунок 8: Визуальная демонстрация алгоритма логистической регрессии

Формальное определение логистической регрессии:

Пусть даны обучающие данные $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, где x_i - это вектор признаков, а y_i - это метка класса для i-го объекта. Логистическая регрессия строит модель, которая связывает вектор признаков x с вероятностью $p(y = 1|x)$ принадлежности объекта x к классу $y = 1$. Функция активации, используемая в логистической регрессии, называется сигмоидной функцией и определяется следующим образом (Формула 12).

$$p(y = 1) = \sigma(w \cdot x_i) = \frac{1}{1 + e^{-w \cdot x_i}} \tag{12}$$

где w - вектор весов, а σ - сигмоидная функция.

В качестве функции потерь в логистической регрессии обычно используется логистическая функция потерь (Формула 7).

Цель обучения логистической регрессии - найти вектор весов w, которые минимизируют функцию потерь на обучающих данных. Эту задачу можно решать различными численными методами оптимизации, такими как градиентный спуск.

2.3 Подготовка данных для алгоритма

Перед применением алгоритма логистической регрессии необходимо выполнить следующие шаги подготовки данных:

- **Очистка данных:** удаление неполных или некорректных записей, заполнение пропущенных значений и преобразование данных в формат, подходящий для анализа.
- **Нормализация данных:** приведение данных к общему масштабу, чтобы каждый признак имел одинаковый вклад в анализ. Нормирование признаков является важным шагом в предобработке данных перед использованием логистической регрессии. Это связано с тем, что логистическая регрессия оценивает веса (коэффициенты) каждого признака в модели, чтобы предсказать целевую переменную. Если масштабы разных признаков значительно отличаются друг от друга, то оценка весов может быть смещена в сторону признаков с большими значениями. Это может привести к неверным выводам и плохому качеству предсказания. Обычно используются два основных способа нормирования:
  - Стандартизация: признаки масштабируются так, чтобы их среднее значение было равно 0, а стандартное отклонение было равно 1.
  - Нормализация: признаки масштабируются так, чтобы их значения находились в диапазоне от 0 до 1.

  После нормирования признаков, каждый из них будет иметь примерно одинаковый вклад в предсказание целевой переменной, что позволит логистической регрессии более точно оценивать их веса и делать более точные прогнозы.
- **Разбиение данных на обучающую и тестовую выборки:** обучающая выборка используется для обучения модели, а тестовая выборка - для оценки ее точности.

2.4 Процесс обучения

Процесс поиска оптимума весов в логистической регрессии градиентным спуском аналогичен тому, как это происходит в линейной регрессии, за исключением функции потерь. Алгоритм логистической регрессии получил свое название от функции потерь, с помощью которой он обучается — логистической регрессии (Формула 7). Давайте поймем, почему выбрана именно такая функция потерь.

Логистическая регрессия нацелена на то, чтобы предсказывать вероятность положительного класса в случае бинарной классификации. Что значит предсказывать вероятность в контексте классификации? Если модель предсказывает вероятность определенного класса, равную, допустим 60%, для определенной подвыборки, например, из 1000 объектов, то количество объектов этого класса в подвыборке должно составлять примерно 600 объектов. Тогда мы можем говорить, что модель корректно предсказывает вероятности.

Для того, чтобы получать вероятности, в логистической регрессии используется сигмоида ($\sigma(x) = \frac{1}{1+e^{-x \cdot w}}$), которая переводит скалярное произведение весов модели на признаки объекта в отрезок (0, 1). Если параметры модели будут подобраны правильно, мы сможем получать:

- вероятность, близкую к 1, для объектов положительного класса
- вероятность, близкую к 0, для объектов отрицательного класса.

В таком случае:

- для объектов положительного класса должно выполняться: $\sigma(x \cdot w) \to 1, x \cdot w \to +\infty$ (сигмоида скалярного произведения стремится к 1 при стремлении скалярного произведения к $+\infty$)
- для объектов отрицательного класса должно выполняться: $\sigma(x \cdot w) \to 0, x \cdot w \to -\infty$ (сигмоида скалярного произведения стремится к 0 при стремлении скалярного произведения к $-\infty$)

Такие условия аналогичны максимизации правильных отступов на объектах. Их можно записать с помощью такого функционала ошибки:

$$\frac{1}{n}\sum_{i=1}^{n}[y_i = 1]\sigma(w \cdot x_i) + [y_i = -1](1 - \sigma(w \cdot x_i)) \to \min_w$$

У этого функционала есть существенный недостаток: он не сильно штрафует модель в случае серьезной ошибки. Если на положительном объекте модель уверена, что это отрицательный объект и сигмоида выдает значение, равное нолю, то штраф будет равен только единице. Если бы вероятность на этом объекте была равна единице, то функция потерь на этом объекте была бы минимальной и равной -1. В этом случае первое слагаемое в функции потерь было бы активировано, и мы бы взяли сигмоиду с минусом и значением -1. Однако в нашем случае сигмоида равна нулю, поэтому функция потерь также равна нулю. Соответственно, штраф равен единице (разность между 0 и -1).

Для ужесточения штрафа в формулу функционала вводят логарифм, получая логарифмический функционал:

$$\frac{1}{n}\sum_{i=1}^{n}[y_i = 1]\log(\sigma(w \cdot x_i)) + [y_i = -1](1 - \log(\sigma(w \cdot x_i))) \to \min_w$$

В этом случае мы получим логарифм нуля, если модель крайне уверена в неправильном ответе. Логарифм нуля равен $-\infty$ и, умноженный на -, даст в итоге $+\infty$, ужесточая таким образом штраф за уверенность в неправильном ответе.

После введения логарифма в функцию потерь над логарифмической функцией производят ряд предобразований, получая в итоге логистическую функцию потерь:

$$L(M) = \log(1 + e^{-M})$$

Далее стандартным алгоритмом градиентного спуска находят минимум этого функционала ошибки.

2.5 Оценка качества алгоритма

Для оценки качества алгоритма логистической регрессии часто используют следующие метрики:

- **accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **precision:** $\frac{TP}{TP+FP}$
- **recall:** $\frac{TP}{TP+FN}$
- **F1:** $2 * \frac{precision*recall}{precision+recall}$

2.6 Интерпретация признаков с помощью алгоритма

Интерпретация признаков в логистической регрессии заключается в понимании того, как каждый признак влияет на зависимую переменную в модели. Это может помочь понять, какие признаки наиболее важны для прогнозирования целевой переменной и как они связаны с ней.

Наиболее простым способом оценить влияние того или иного признака - посмотреть на весовой коэффициент при признаке. Коэффициенты признаков в логистической регрессии представляют величину и направление влияния каждого признака на зависимую переменную. Если значение признака большое, можно сказать, что его значение также большое (при условии нормализованных данных). Если коэффициент положительный, то увеличение значения признака будет приводить к увеличению значения зависимой переменной. Если коэффициент отрицательный, то увеличение значения признака будет приводить к уменьшению значения зависимой переменной.

2.7 Модификации алгоритма

Существует несколько модификаций алгоритма логистической регрессии, которые могут быть применены в зависимости от задачи и данных. Некоторые из них:

- **добавление регуляризации L2:** добавление регуляризации L2 в функцию потерь для борьбы с переобучением. Это позволяет уменьшить веса модели, чтобы она была менее склонна к переобучению и более устойчива к шуму в данных.
- **добавление регуляризации L1:** добавление регуляризации L1 в функцию потерь, чтобы стимулировать разреженность весов модели. Это означает, что некоторые веса будут установлены на ноль, что может помочь идентифицировать наиболее важные признаки.
- **Мультиклассовая логистическая регрессия:** алгоритм логистической регрессии может быть расширен на случай многоклассовой классификации с помощью подходов One-vs-All, One-vs-One, softmax + cross-entropy

2.8 Область применения алгоритма

лгоритм логистической регрессии широко используется в различных областях:

- Медицина: например, для определения вероятности заболевания у пациента на основе их медицинских данных.
- Финансы: например, для определения вероятности дефолта заемщика на основе его кредитной истории.
- Маркетинг: например, для прогнозирования вероятности покупки товара на основе поведенческих данных покупателя.
- Обработка естественного языка: например, для классификации текстовых документов или определения тональности текста.
- Компьютерное зрение: например, для классификации изображений.
- Рекомендательные системы: например, для предсказания рейтинга товара на основе истории покупок и поведения пользователя.
- Информационная безопасность: например, для определения вероятности атаки на систему на основе ее характеристик.

- Биоинформатика: например, для анализа генетических данных и определения вероятности развития заболевания.

В целом, алгоритм логистической регрессии может быть применен в любой области, где необходимо решать задачи бинарной или многоклассовой классификации на основе набора признаков.

2.9 Плюсы и минусы алгоритма

Алгоритм логистической регрессии имеет ряд преимуществ и недостатков.
Плюсы:

- Простота реализации и понимания. Это один из самых простых алгоритмов машинного обучения.
- Может использоваться для задач классификации и оценки влияния признаков.
- Эффективен на больших выборках и может быстро обучаться на большом количестве признаков.
- Хорошо интерпретируется и может помочь понять важность каждого признака.

Минусы:

- Чувствительность к выбросам и шуму в данных.
- Требует выполнения предположения о линейной зависимости между признаками и целевой переменной, что может быть проблематично в некоторых случаях.
- Не учитывает взаимодействие между признаками, что может привести к недооценке важности некоторых признаков.
- Может быть склонен к переобучению, если в модели слишком много признаков или коэффициенты признаков сильно отличаются друг от друга.
- Неустойчив к мультиколлинеарности. Алгоритм логистической регрессии может быть неустойчив к мультиколлинеарности между признаками, что может привести к неопределенности в оценке весов модели.

В целом, алгоритм логистической регрессии имеет много преимуществ, таких как простота реализации, хорошая обобщающая способность и возможность вероятностной интерпретации, но также имеет ограничения, связанные с линейностью модели и чувствительностью к выбросам и мультиколлинеарности.

2.10 Реализация алгоритма в Python

В библиотеке Scikit-learn в модуле linear_model также реализован класс **LogisticRegression** для логистической регрессии.

Класс LogisticRegression имеет следующие параметры:

- **penalty:** строка, указывающая тип регуляризации (‘l1’, ‘l2’, ‘elasticnet’, None). По умолчанию равен ’l2’, что означает регуляризацию L2.
- **dual:** булевый параметр, указывающий, должна ли быть решена двойственная задача оптимизации. По умолчанию равен False.
- **tol:** вещественное число, задающее критерий остановки. По умолчанию равен False.
- **C:** коэффициент регуляризации, положительное вещественное число. По умолчанию равен 1e-4.

- **fit_intercept:** булевый параметр, указывающий, должно ли применяться смещение (intercept/bias). По умолчанию равен True.
- **intercept_scaling:** вещественное число, указывающее, должен ли масштаб смещения быть умножен на C. По умолчанию равен 1.
- **class_weight:** строка или словарь, указывающий, должно ли быть использовано взвешивание по классам (вес задается в зависимости от относительной частоты класса). По умолчанию равен None.
- **random_state:** число или объект RandomState, указывающий, должен ли использоваться случайный генератор для воспроизводимости результатов работы модели. По умолчанию равен None.
- **n_jobs:** количество параллельных задач, которые используются для расчета. По умолчанию равен None, что означает использование всех доступных процессоров.

Класс LogisticRegression имеет методы:

- **fit(X, y)** для обучения модели на данных X и y
- **predict(X)** для предсказания целевых значений для новых данных X
- **score(X, y)** для получения точности модели
- **get_params()** для получения параметров модели соответственно
- **predict_proba(X)**, который возвращает вероятности принадлежности классу для новых данных X.

2.11 Вопросы для самопроверки

С помощью какой функции в логистической регрессии происходит отображение вещественных значений на отрезок (0, 1)?

1. тангенс
2. сигмоида
3. логистическая функция

Правильные ответы: 2

Какой функционал ошибки может использоваться для обучения логистической регрессии?

1. MSE
2. Логистический
3. Логарифмический

Правильные ответы: 2, 3

Выберите плюсы алгоритма логистической регрессии:

1. Эффективен на больших выборках и может быстро обучаться на большом количестве признаков.
2. Чувствительность к выбросам и шуму в данных.
3. Хорошо интерпретируется и может помочь понять важность каждого признака.
4. Требует выполнения предположения о линейной зависимости между признаками и целевой переменной, что может быть проблематично в некоторых случаях.

5. Простота реализации и понимания. Это один из самых простых алгоритмов машинного обучения.
6. Не учитывает взаимодействие между признаками, что может привести к недооценке важности некоторых признаков.
7. Может использоваться для задач классификации и оценки влияния признаков.
8. Может быть склонен к переобучению, если в модели слишком много признаков или коэффициенты признаков сильно отличаются друг от друга.

Правильные ответы: 1, 3, 5, 7

2.12 Резюме по разделу

Алгоритм логистической регрессии - это метод машинного обучения, который используется для классификации объектов. Алгоритм предсказывает вероятность принадлежности объекта к определенному классу, основываясь на линейной комбинации значений признаков и коэффициентов регрессии, а также сигмоидальной функции, ккоторая переводит вещественные значения на отрезок (0, 1).

Коэффициенты регрессии - это параметры модели, которые определяют веса признаков в предсказании класса объекта. Чтобы определить коэффициенты, используется функционал ошибок, называемый **логистической функцией потерь**.

Основными преимуществами логистической регрессии являются простота и интерпретируемость модели, а также возможность использования для классификации бинарных и многоклассовых задач. Недостатками алгоритма являются чувствительность к выбросам и нелинейным зависимостям между переменными.

3 Метод опорных векторов

Цель занятия: ученик может применить метод опорных векторов для решения задач классификации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Описание алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Интерпретация признаков с помощью алгоритма
- Модификации алгоритма
- Область применения алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

3.1 Визуальная демонстрация алгоритма

Метод опорных векторов (англ. Support Vector Machine, SVM) - это алгоритм машинного обучения, использующийся для классификации. Он основывается на поиске гиперплоскости в пространстве признаков, которая наилучшим образом разделяет данные на два класса. В случае SVM мы ищем гиперплоскость, которая максимизирует зазор (отступ между двумя классами данных). Зазор (опроный вектор) в контексте SVM - это расстояние от гиперплоскости до ближайшей точки каждого класса. Визуальная демонстрация алгоритма представлена на иллюстрации 9.

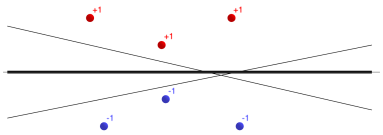
Давайте разберемся с тем, что изображено на иллюстрации 9. На рисунках 9.1-9.2 отображена суть SVM - поиск гиперплоскости в пространстве признаков, которая наилучшим образом разделяет данные на два класса (оптимальный отступ). Однако такая безусловная оптимизация (Hard-margin SVM) не приспособлена для работы с объектами, которые классифицируются неправильно. Для таких случаев вводятся дополнительные переменные ϵ_i, которые разрешают на некоторых объектах делать отступ меньше единицы и коэффициент C, которые позволяет выбирать, что важнее: сделать отступ шире либо сделать точность модели лучше (Рисунки 9.3)-9.4. На рисунках 9.5-9.6 показан так называемый Kernel trick в SVM: для линейно неразделимых выборок можно увеличить размерность данных для того, чтобы данные стали линейно разделимыми.

3.2 Описание алгоритма

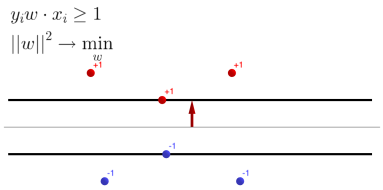
Дадим формальное определение алгоритма SVM. SVM (англ. Support Vector Machine) - это алгоритм машинного обучения для задач классификации и регрессии. Он находит оптимальную гиперплоскость, которая разделяет два класса данных с наибольшим зазором (margin) между ними и наименьшим количеством ошибок при классификации.

Рассмотрим SVM для линейно разделимых выборок.

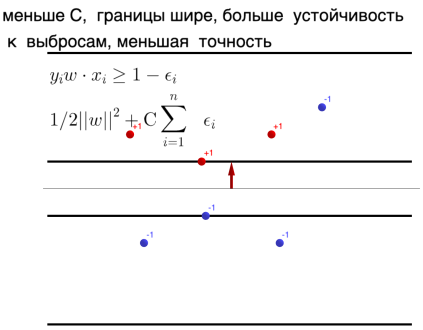
Hard-margin SVM - это метод машинного обучения, который находит линейную гиперплоскость, разделяющую два класса данных в линейно разделимом случае. Он предполагает, что существует идеальная разделяющая гиперплоскость, которая не допускает никаких ошибок классификации на обучающем наборе данных.



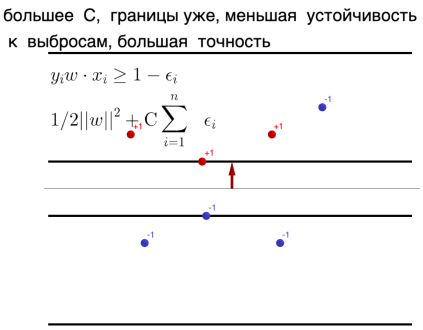
(1)



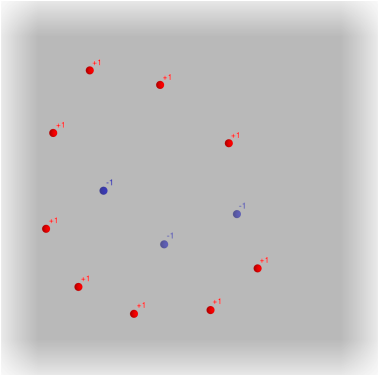
(2)



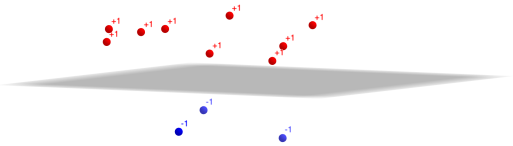
(3)



(4)



(5)



(6)

Рисунок 9: Визуальная демонстрация метода опорных векторов

Формально, пусть дано обучающее множество X с n элементами, где каждый элемент x представлен вектором p признаков, и метки классов y_i , где y_i принимает значение 1 или -1, обозначающее класс, к которому относится i-й элемент.

Тогда, Hard-margin SVM ищет линейную гиперплоскость $f(x) = w^T x$, которая разделяет два класса данных с наибольшим зазором (margin) между ними и отступом: $y_i(w \cdot x_i + w_0) > 0$. Зазор определяется как расстояние от гиперплоскости до ближайших элементов каждого

класса. Отступ от гиперплоскости вычисляется по формуле:

$$\frac{|w \cdot x_i + w_0|}{||w||}$$

.

Соответственно, зазор вычисляется как:

$$\min_{x_i \in X} \frac{|w \cdot x_i + w_0|}{||w||}$$

.

После этого происходит нормирование весов путем деления их на минимальное значение отступа. После такой нормировки минимальное значение отступа будет равно 1:

$$\min_{x_i \in X} \frac{|w \cdot x_i + w_0|}{||w||} = 1$$

.

Можно вернуться к зазору классификатора. Так как $||w||$ не зависит от i, $||w||$ выносится за знак минимума, $\min\limits_{x_i \in X} |w \cdot x_i + w_0|$ равен 1 после нормировки, соответственно зазор равен:

$$\min_{x_i \in X} \frac{|w \cdot x_i + w_0|}{||w||} = \frac{\min\limits_{x_i \in X} |w \cdot x_i + w_0|}{||w||} = \frac{1}{||w||}$$

Соответственно, у нас появляется ряд требований к модели, на основе которых можно определить условную оптимизацию для модели:

- $y_i(w \cdot x_i + w_0) > 0$
- максимальный зазор для классификатора: $\max\limits_{w} \frac{1}{||w||}$ при условии, что $|w \cdot x_i + w_0| \geq 1$

Максимизация $\max\limits_{w} \frac{1}{||w||}$ эквивалентна минимизации $\min\limits_{w} ||w||$.

В итоге получается задача условной оптимизации для SVM (Формула 13).

$$\begin{cases} ||w||^2 \to \min\limits_{w} (||w||^2 \text{ берется в квадрате для облегчения вычисления производных}) \\ |w \cdot x_i + w_0| \geq 1 \end{cases} \tag{13}$$

Решение этой задачи оптимизации находит вектор весов w и смещение b , которые определяют разделяющую гиперплоскость и минимизируют норму вектора весов.

Таким образом, Hard-margin SVM ищет идеальную гиперплоскость, которая разделяет данные на два класса, и не допускает никаких ошибок классификации на обучающем наборе данных. Однако, такой подход может быть чувствителен к выбросам и не работать в случае, когда данные линейно неразделимы.

В случае, если выборка линейно неразделима, условие $|w \cdot x_i + w_0| \geq 1$ невалидно. Для этого случая используется Soft-margin SVM.

Soft-margin SVM — это модификация классического Hard-margin SVM, которая позволяет решать задачи классификации, учитывая наличие шумовых или выбросовых данных в обучающей выборке. В отличие от классического SVM, где решающая граница строго разделяет классы, в Soft-margin SVM допускаются некоторые ошибки классификации за счет допуска нарушения ограничений на расстояние от объектов до разделяющей границы (margin).

Формально, Soft-margin SVM решает задачу оптимизации, минимизируя следующий функционал (Формула 14):

$$\begin{cases} y_i w \cdot x_i \geq 1 - \epsilon_i \\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \epsilon_i \to \min\limits_{w} \end{cases}$$

где $y_i \in \{-1, 1\}$ - класс объекта, x_i - вектор его признаков,

w - вектор весов, ϵ_i - дополнительные переменные, обозначающие степень

нарушения ограничений на margin для каждого объекта,

C - гиперпараметр, определяющий вес ошибки классификации в оптимизационной задаче.

(14)

Таким образом, в Soft-margin SVM допускаются объекты, которые находятся внутри margin или даже на неправильной стороне разделяющей границы, при этом степень нарушения ограничений контролируется параметром C. Большее значение C приводит к более строгой классификации, тогда как меньшее значение C допускает большее количество ошибок классификации.

3.3 Подготовка данных для алгоритма

Перед применением алгоритма SVM необходимо выполнить следующие шаги подготовки данных:

- **Очистка данных:** удаление неполных или некорректных записей, заполнение пропущенных значений и преобразование данных в формат, подходящий для анализа.
- **Нормализация данных:** приведение данных к общему масштабу, чтобы каждый признак имел одинаковый вклад в анализ. Нормирование признаков является важным шагом в предобработке данных перед использованием SVM. Это связано с тем, что SVM оценивает веса (коэффициенты) каждого признака в модели, чтобы предсказать целевую переменную. Если масштабы разных признаков значительно отличаются друг от друга, то оценка весов может быть смещена в сторону признаков с большими значениями. Это может привести к неверным выводам и плохому качеству предсказания. Обычно используются два основных способа нормирования:
  - Стандартизация: признаки масштабируются так, чтобы их среднее значение было равно 0, а стандартное отклонение было равно 1.
  - Нормализация: признаки масштабируются так, чтобы их значения находились в диапазоне от 0 до 1.

  После нормирования признаков, каждый из них будет иметь примерно одинаковый вклад в предсказание целевой переменной, что позволит SVM более точно оценивать их веса и делать более точные прогнозы.
- **Разбиение данных на обучающую и тестовую выборки:** обучающая выборка используется для обучения модели, а тестовая выборка - для оценки ее точности.

3.4 Процесс обучения

Обучение SVM с использованием градиентного спуска включает в себя следующие шаги:

1. **Формирование обучающего набора данных.** Подготавливаем обучающий набор данных, содержащий объекты, каждый из которых имеет некоторые признаки и относится к одному из двух классов.
2. **Выбор функции потерь.** В SVM обычно используют функцию потерь hinge loss, которая выражается следующим образом: $L(y_i, \mathbf{w}) = \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x_i}))$ где y_i - метка класса для объекта i, $\mathbf{w}$ - параметры гиперплоскости, а $\mathbf{x_i}$ - вектор признаков объекта i.
3. **Определение оптимальных параметров.** SVM имеет гиперпараметр C, который определяет, насколько много ошибок классификации допускает SVM. В данном случае его можно интерпретировать как коэффициент регуляризации, который позволяет контролировать сложность модели. Параметр C может быть настроен с помощью кросс-валидации.
4. **Определение градиента.** В случае hinge loss градиент выражается следующим образом: $\frac{\partial L(y_i, \mathbf{w})}{\partial \mathbf{w}} = \begin{cases} -y_i \mathbf{x_i}, & \mathbf{w}^T \mathbf{x_i} > 1 \\ 0, & \text{otherwise} \end{cases}$
5. **Обновление параметров.** Для обновления параметров $\mathbf{w}$ и b используется формула градиентного спуска: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial L}{\partial \mathbf{w}}$ где α - скорость обучения, которая определяет, как быстро меняются параметры.
6. Определение условия останова. Обучение останавливается, когда достигается некоторый критерий останова, например, когда достигается максимальное количество итераций или когда функция потерь перестает уменьшаться на заданном уровне точности.
7. **Повторение шагов 4-6.** Шаги 4-6 повторяются до тех пор, пока модель не достигнет заданного уровня точности или не будет выполнен критерий останова.
8. **Построение гиперплоскости.** После обучения SVM, гиперплоскость, разделяющая два класса, может быть построена из параметров $\mathbf{w}$.

3.5 Оценка качества алгоритма

Для оценки качества алгоритма SVM часто используют следующие метрики:

- **accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **precision:** $\frac{TP}{TP+FP}$
- **recall:** $\frac{TP}{TP+FN}$
- **F1:** $2 * \frac{precision * recall}{precision + recall}$

3.6 Интерпретация признаков с помощью алгоритма

Интерпретация признаков в SVM заключается в понимании того, как каждый признак влияет на зависимую переменную в модели. Это может помочь понять, какие признаки наиболее важны для прогнозирования целевой переменной и как они связаны с ней.

Наиболее простым способом оценить влияние того или иного признака - посмотреть на весовой коэффициент при признаке. Коэффициенты признаков в SVM представляют величину и направление влияния каждого признака на зависимую переменную. Если значение признака большое, можно сказать, что его значение также большое (при условии нормализованных данных). Если коэффициент положительный, то увеличение значения признака будет приводить к увеличению значения зависимой переменной. Если коэффициент отрицательный, то

увеличение значения признака будет приводить к уменьшению значения зависимой переменной.

Важно отметить, что интерпретация признаков в SVM должна быть осуществлена с осторожностью, поскольку влияние каждого признака на принятие решения может зависеть от остальных признаков и от их взаимодействия. Кроме того, при использовании ядерных SVM, интерпретация признаков может быть затруднена, поскольку признаковое пространство может быть преобразовано в другое пространство, где трудно понять, какие именно признаки оказывают влияние на решение.

3.7 Модификации алгоритма

Существует несколько модификаций алгоритма SVM, которые могут быть применены в зависимости от задачи и данных. Некоторые из них:

- **Soft-margin SVM:** Soft-margin SVM разрешает некоторым объектам попадать внутрь границы разделения классов (маргинала), что снижает требования к разделимости данных. Это позволяет создать более гибкую границу разделения, которая может лучше обобщать на новые данные, которые могут быть не идеально разделимыми. Вместо того, чтобы минимизировать только ошибки классификации, как в Hard-margin SVM, Soft-margin SVM минимизирует ошибки классификации и штрафует за нарушение границы маргинала.
- **SVM с ядрами:** SVM с ядрами позволяет перевести нелинейно разделимые данные в более высокую размерность пространства, где данные могут быть линейно разделимыми. Вместо того, чтобы искать гиперплоскость, разделяющую данные в исходном пространстве признаков, SVM с ядрами ищет гиперплоскость в новом пространстве, созданном путем применения функции ядра к исходным данным. Некоторые из наиболее распространенных ядер SVM включают в себя:
- – Линейное ядро: простейшее ядро, которое используется для линейной разделимости классов.
  – Полиномиальное ядро: используется для увеличения размерности пространства данных и построения нелинейных границ.
  – RBF (Radial Basis Function) ядро: самое распространенное ядро, которое создает нелинейную разделяющую гиперплоскость.
  – Sigmoid ядро: используется для задач классификации с нелинейной разделимостью.
- **Multi-class SVM:** Multi-class SVM позволяет обрабатывать данные с несколькими классами. Одним из подходов является использование метода "один против всех в котором SVM обучается для каждого класса, чтобы отличать его от всех остальных классов.
- **Incremental SVM:** Incremental SVM позволяет обучать SVM на больших объемах данных, постепенно добавляя новые объекты и/или признаки к обучающей выборке. Это удобно, если данные поступают постепенно или если вычислительные ресурсы ограничены.
- **Online SVM:** Online SVM позволяет обучать SVM на потоке данных, который может быть бесконечным или динамическим. Это достигается путем последовательного обновления параметров SVM по мере получения новых данных.

3.8 Область применения алгоритма

Некоторые из основных областей применения алгоритма SVM включают в себя:

- **Классификация текстов:** SVM может быть использован для классификации текстовых данных, таких как электронные письма или отзывы на продукты, на положительные и отрицательные классы.
- **Биоинформатика:** SVM может быть использован для классификации белков, распознавания генов и прогнозирования свойств биомолекул.
- **Компьютерное зрение:** SVM может быть использован для обнаружения объектов на изображении, распознавания лиц, классификации изображений и т.д.
- **Финансовый анализ:** SVM может быть использован для прогнозирования цен на акции и классификации инвестиционных продуктов.
- **Интернет-реклама:** SVM может быть использован для классификации пользователей в зависимости от их поведения в интернете и предсказания того, какие рекламные объявления будут наиболее эффективны для конкретного пользователя.
- **Анализ данных:** SVM может быть использован для кластеризации и классификации данных, таких как данные о клиентах, данные о продажах и данные о производственных процессах.
- **Медицинская диагностика:** SVM может быть использован для классификации медицинских данных, таких как изображения CT и MRI, для определения наличия или отсутствия определенного заболевания.

В целом, алгоритм SVM является мощным инструментом машинного обучения, который может использоваться для решения различных задач классификации в различных областях.

3.9 Плюсы и минусы алгоритма

Алгоритм SVM имеет ряд преимуществ и недостатков.
Плюсы:

- Эффективность: SVM хорошо справляется с выборкой большого объема и с большим количеством признаков.
- Хорошая обобщающая способность: SVM может обучаться на небольшом количестве данных, не переобучаясь.
- Гибкость: SVM может использоваться с различными типами ядер, что позволяет адаптироваться к различным задачам.
- Высокая точность: SVM может давать высокую точность при решении задач классификации.

Минусы:

- Чувствительность к выбросам: SVM чувствителен к выбросам в данных, что может привести к неправильным результатам.
- Не подходит для несбалансированных данных: если данные несбалансированы, то SVM может давать неправильные результаты.
- Выбор ядра: выбор подходящего ядра для конкретной задачи может быть сложной задачей.
- Не подходит для большого числа классов: SVM не всегда подходит для задач с большим числом классов.

В целом, алгоритм SVM является мощным инструментом машинного обучения, который имеет свои преимущества и недостатки. При выборе алгоритма необходимо учитывать особенности конкретной задачи и обеспечивать подходящую настройку параметров SVM.

3.10 Реализация алгоритма в Python

В библиотеке Scikit-learn также реализован класс **SVC** для SVM. Класс SVC имеет следующие параметры:

- **C:** коэффициент регуляризации. Меньшие значения C создают более широкие разделяющие гиперплоскости, позволяющие большему количеству наблюдений нарушать ограничения. По умолчанию C=1.
- **kernel:** ядро, используемое в SVM. Поддерживаются линейное ядро, радиальное ядро базисной функции (RBF), полиномиальное ядро и другие. По умолчанию используется RBF.
- **degree:** степень полиномиального ядра. Используется только если kernel='poly'. По умолчанию degree=3.
- **gamma:** коэффициент ядра RBF. Большие значения gamma приводят к более точному соответствию обучающим данным, но могут приводить к переобучению. По умолчанию gamma='scale', что означает, что оно будет равно 1 / (n_features * X.var()), где X - обучающая выборка.
- **coef0:** свободный член, используемый в ядрах полинома и сигмоиды. По умолчанию coef0=0.0.
- **shrinking:** булевый параметр, указывающий, должен ли использоваться алгоритм уменьшения границ. По умолчанию shrinking=True.
- **probability:** булевый параметр, указывающий, должны ли выдаваться оценки вероятности для классификации. По умолчанию probability=False.
- **tol:** критерий останова оптимизации. По умолчанию tol=1e-3.
- **max_iter:** максимальное количество итераций. По умолчанию max_iter=-1, что означает, что нет ограничения на количество итераций.

Класс SVC также имеет методы **fit(X, y)** для обучения модели на данных X и y, а также метод **predict(X)** для предсказания целевых значений для новых данных X. Кроме того, класс SVC имеет методы **score(X, y)** и **get_params()** для получения оценки точности модели и параметров модели соответственно.

3.11 Вопросы для самопроверки

Какое главное отличие между SVM и логистической регрессией?

1. SVM является линейным классификатором, в то время как логистическая регрессия - нелинейным
2. SVM стремится найти гиперплоскость, максимально разделяющую классы, в то время как логистическая регрессия моделирует вероятности принадлежности к классам
3. SVM может работать только с двумя классами, в то время как логистическая регрессия может работать с многоклассовыми задачами
4. SVM может использовать различные ядра для нелинейного разделения классов, в то время как логистическая регрессия не имеет такой возможности

Правильные ответы: 2

Какую роль играет параметр C в SVM?

1. Он определяет ширину зазора между гиперплоскостью и ближайшими объектами разных классов.
2. Он определяет, сколько объектов может оказаться на неверной стороне гиперплоскости.
3. Он определяет, насколько сильно модель будет штрафовать за ошибки на обучающей выборке.
4. Он определяет, какую функцию потерь будет использовать модель.

Правильные ответы: 3

Выберите плюсы алгоритма SVM:

1. Хорошая масштабируемость на больших датасетах.
2. Высокая точность классификации.
3. Способность работать с данными, не являющимися линейно разделимыми с помощью ядер
4. Возможность качественно работать с большим количеством признаков.
5. Подходит для решения задач регрессии.
6. Не требует предварительной обработки данных.

Правильные ответы: 1, 2, 3, 4

3.12 Резюме по разделу

Алгоритм SVM (Support Vector Machine) - это метод машинного обучения, который используется для классификации. Алгоритм строит гиперплоскость в пространстве признаков, которая разделяет объекты разных классов.

Опорные вектора (Support Vectors) - это объекты обучающей выборки, которые лежат ближе всего к разделяющей гиперплоскости. Опорные вектора используются для определения гиперплоскости и оптимизации функции потерь.

Основными преимуществами SVM являются эффективность в высокоразмерных пространствах, хорошая обобщающая способность и возможность работы с нелинейными данными. Недостатками алгоритма являются сложность выбора правильного ядра и параметров модели, а также чувствительность к выбросам.

Также существуют модификации алгоритма, такие как SVM с ядровыми функциями, которые позволяют работать с нелинейными данными, SVM с мягким зазором (Soft-margin SVM), которая позволяет работать с неидеальными данными и SVM с несколькими классами, которые позволяют классифицировать объекты на более чем два класса.

4 Резюме по модулю

В этом модуле мы рассмотрели такие темы, как линейная классификация, логистическая регрессия, а также с метод опорных векторов. Линейная классификация отличается от задачи регрессии тем, что в ней необходимо предсказать метку класса или вероятность класса, а не вещественное число. Суествует две разновидности задач классификации: бинарная классификация и многоклассовая классификация. Мы также рассмотрели функции потерь для задач классификации: логистическая функция потерь, логарифмическая функция потерь, функция потерь Хинджа.

Также были рассмотрены две модели линейной классификации: логистическая регрессия и метод опорных векторов. В логистической регрессии используется сигмоида для преобразования вещественного скалярного произведения вектора весов на вектор признаков в диапазон (0, 1). В методе опорных векторов (SVM) мы ищем гиперплоскость, которая максимизирует зазор (отступ между двумя классами данных).