ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Производная, градиент и градиентная оптимизация.

Аннотация к занятию: обучающиеся познакомятся с алгоритмом градиентной оптимизации для поиска минимума функции и применением алгоритма для функций многих переменных. Обсудят, как задача минимизации функции связана с машинным обучением.

Цель занятия: сформировать у обучающихся представление о задаче поиска минимума функции и познакомить с алгоритмом градиентной оптимизации.

Задачи занятия:

- научить вычислять производную функции многих переменных;
- сформировать представление о частных производных и градиенте;
- обсудить задачу поиска минимума функции;
- познакомить с алгоритмом градиентной оптимизации.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	5 мин.	Друзья, здравствуйте! На предыдущем уроке мы познакомились с понятием производной функции, научились её вычислять. На этом уроке мы продолжим работать с производной	Приветствие. Создание в классе атмосферы психологического комфорта
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	7 мин.	**Вопрос для обсуждения** Ранее мы рассматривали производные функций одной переменной. Как найти производную функции многих переменных? **Возможные ответы обучающихся:** ● найти по каждой переменной; ● её невозможно найти. Сегодня научимся вычислять производные многих функций. Узнаем правила вычисления производных более сложных, составных функций	Способствовать обсуждению мотивационных вопросов
Изучение нового материала	50 мин.	Напомню, что такое функция многих переменных. На экране вы видите функцию f от двух переменных (x1 и x2) и её график. На графике обозначена точка,	Для справки:

		соответствующая значениям x1 равно минус двадцать и x2 равно минус двадцать. Варьируя значения x1 и x2, мы будем получать различные значения функции f. Бывают функции от трёх, четырёх и сколько угодно переменных. На самом деле, мы уже сталкивались с функцией многих переменных, когда говорили о линейной регрессии. **Вопрос для обсуждения** Давайте вспомним, что такое линейная регрессия. Это модель для решения задачи регрессии. Она ставит в соответствие каждому признаку датасета свой коэффициент. И затем для входящего элемента вычисляет ответ по формуле, которую вы видите на экране. В этой формуле xi — i-ый признак элемента. Например, для первого элемента датасета ответ линейной регрессии y1 с шапочкой выглядел бы вот так. Смотрите: формулу ответа модели для первого элемента датасета можно представить в виде функции f от шести переменных: k0, k1, k2, k3, k4 и k5. Меняя значения ki-ых, мы получаем разные модели линейной регрессии, которые выдают разные ответы на первый элемент. Нарисовать графики функций, у которых больше, чем две переменные, мы не можем. Понять, возрастают или убывают они в разных точках и с какой скоростью, можно только с помощью производных.	Сайт: https://habr.com/ru/post/413853/ Перед уроком рекомендуется ознакомиться с материалами, представленными на сайте.

		Так давайте научимся вычислять производные функций многих переменных. Вернёмся к нашей функции f от x1 и x2. Давайте представим, что x2 — это не переменная, а константа. Как будто у функции f только одна переменная — x1. У такой функции мы можем взять производную. Она будет равна шесть x1 плюс два x2. Это называется частной производной функции f по переменной x1. Или проще: производной f по x1. Записывается это как f' с нижним индексом x1 или как df по dx1. Обратите внимание, что в записи формулы буква d должна быть не совсем обычная, а немного закруглённая. **Важно:** несмотря на то, что мы вычисляли эту производную, представляя, что x2 — это не переменная, производная f' по x1 — это всё равно функция от двух переменных, x1 и x2. То есть сначала мы берём функцию f, представляем, что x2 — это не переменная, а константа, вычисляем так производную f по x1, и дальше снова начинаем считать, что x2 — это переменная. Обобщим понятие частной производной функции многих переменных. Пусть у нас есть функция f от n переменных. Чтобы вычислить частную производную функции f по переменной x1, нужно: 1. Представить, что все остальные переменные x2, x3 и так далее — константы. 2. .Вычислить производную f по x1.	Для справки: https://ru.wikipedia.org/wiki/Частная_производная

		3. Снова начать считать x2, x3 и так далее переменными. Всё. Так мы получаем функцию f' с индексом x1 от n переменных. Это и будет частная производная f' по x1. Аналогично вычисляются остальные частные производные f' по x2, f' по x3 и так далее. Всего у функции с n переменными будет n частных производных. Вернёмся к нашей функции f от двух переменных. Мы уже вычислили для нее f' по x1. Так как частные производные — это тоже функции, мы можем вычислять значения производных в различных точках. Посмотрим на знакомую нам точку x1 равно x2 равно 20. Обозначим её буквой A. На слайде вы видите её положение на графике функции. Значение частной производной f' по x1 в этой точке равно минус сто шестьдесят. Давайте обсудим, что это значит. Смысл частных производных на самом деле такой же, как и у обычной производной функции одной переменной — о ней мы говорили на прошлых занятиях. Значение частной производной f' по x1 отражает то, что будет происходить со значением функции f, если мы из точки A сдвинемся по x1 на бесконечно малое значение delta x вправо. То есть перейдём в точку (−20 + delta x −20).	

		Ещё проще: смотрите, наше значение производной f’ по x1 в точке A равно минус сто шестьдесят. Это значение меньше нуля. Это значит, что функция f в точке A убывает по x1. Это видно по графику: если мы из точки A при неизменном значении x2 будем двигаться вправо по оси x1, то значение нашей функции будет уменьшаться. Посмотрим на производную f’ по x2 в точке A. F’ по x2 равно два x1. Подставив значение x1 равно минус двадцать, получим, что f’ по x2 в точке A равно минус 40. Тоже отрицательное значение. Это значит, что в точке A функция f по переменной x2 убывает. Это заметно и на графике: двигаясь параллельно оси x2 от точки A в сторону увеличения значения x2, значение функции уменьшается. Замечу, что f’ по x2 в точке A меньше по модулю, чем f’ по x1 в точке A. Значит, функция f в точке A по x1 убывает быстрее, чем по x2. Это тоже видно на графике: склон функции при движении вдоль оси x1 гораздо круче, чем при движении вдоль оси x2. Таким образом, мы увидели, что смысл частных производных такой же, как и смысл производной функции одной переменной. Частные производные отражают характер поведения функции в точке по отдельным переменным. Иначе говоря, частная производная f’ по x в точке показывает нам, в какую сторону нам нужно сдвинуться из точки по переменной x, чтобы значение функции уменьшилось.	Для справки:

		Теперь мы можем ввести понятие градиента. Градиент функции в точке x — это просто вектор значений частных производных функции в точке x. Сколько у функции переменных, столько и будет элементов в векторе градиента. Например, для нашей функции f её градиент в точке A — это вектор минус сто шестьдесят, минус сорок. Обозначается градиент значком, который называется nabla. Он выглядит как перевернутый треугольник. У градиента есть смысл. Он содержит всю информацию о том, возрастает или убывает функция f в точке по всем переменным, и насколько быстро. Получается, вычислив градиент функции в точке, мы можем понять, в какую сторону нам нужно сдвинуться из этой точки по каждой из переменных. Так, чтобы значение функции уменьшилось. Например, для нашей функции f и точки A нам нужно увеличить значения обеих переменных x1 и x2, чтобы значение функции уменьшилось. На графике движение по обеим переменным x1 и x2 в сторону увеличения показано красной стрелкой. Видно, что, увеличивая x1 и x2, мы уменьшаем значение f. Градиентная оптимизация — это инструмент, который позволяет решить задачу минимизации функции. Подробнее остановимся на том, что же такое задача минимизации функции.	Сайт: https://ru.abcdef.wiki/wiki/Gradient

		Пусть у нас есть функция f(x). Задача минимизации этой функции состоит в нахождении её точки минимума. Мы помним, что минимумы функции бывают локальные и глобальные. Алгоритмы поиска минимума функций чаще всего ищут точки локальных минимумов функций. Для функции многих переменных найти точку минимума значит найти такой набор значений параметров, что значение функции для них будет минимально. На слайде вы видите функцию от двух переменных и её график. Её точка минимума — при значениях x1 и x2, равных нулю. При любых других значениях x1 и x2 значение функции f будет больше нуля. Задачу минимизации функции называют и по-другому. Например, задачей поиска минимума функции или задачей оптимизации функции. Почему поиск точек минимума функций важен? Дело в том, что обучение многих моделей машинного обучения как раз и состоит в том, чтобы искать точки минимума метрик качества. Так, к примеру, обучаются нейросети. Более подробно об этом вы узнаете в следующих модулях, когда будете изучать логистическую регрессию и нейросети. **Вопрос для обсуждения** Итак, как же решить такую задачу? Как по данной функции f найти её точку минимума? **Возможный ответ обучающихся**	

		Найти производную. Здесь нам и пригодятся производные. Рассмотрим первый способ нахождения точек минимума — аналитический. Вспомним одно из свойств производной. Производная в точках экстремума функции равна нулю. Поэтому идея поиска точек минимума функции такая: • берём функцию f, вычисляем её производную f'; • затем решаем уравнение f'(x) = 0; • находим значения x — корни этого уравнения. Так как при этих значениях x производная f'(x) будет равна нулю, то корни уравнения являются точками экстремумов функции f — то есть, точками минимума и максимума; • после этого остаётся последний шаг — понять, какие из найденных точек — точки минимума. Эта идея хороша, но, к сожалению, работает далеко не для всех функций. Убедимся в этом на примере знакомой функции f(x), график которой вы видите на экране. Её производная f' равна четыре x в третьей плюс пятнадцать x в квадрате минус десять. Получается, чтобы найти точки минимума, нужно решить уравнение f'(x) равно нулю. f' — это многочлен третьей степени, и решить такое уравнение непросто. А что если функция f была бы десятой или двадцатой степени? Не говоря уж о случае, когда f — функция многих переменных.	

		Получается, такой способ нахождения точек минимума функции работает не всегда. Для многих функций f второй пункт этого плана сложно выполним, если вообще выполним. Что же делать? К сожалению, никакого другого способа, который бы точно находил все точки минимума у любой функции, не существует. Мы рассмотрим другой способ, который позволяет находить некоторые точки локального минимума функций. Он основан на производной и называется алгоритмом градиентной оптимизации. Возьмём произвольную точку функции f. Например, точку x равно минус пять. Значение функции в этой точке равно пятидесяти. Вычислим значение производной в этой точке: оно равно минус ста тридцати пяти. Вспомним, о чем говорит нам знак производной: о том, что функция f в точке минус пять убывает. Что это значит: раз функция в точке минус пять убывает, значит, где-то правее этой точки находится локальный минимум. Если мы сдвинемся из точки минус пять вправо, то, скорее всего, подвинемся чуть ближе к точке локального минимума. Давайте сдвинемся вправо на шаг delta x равно 1 и попадём в точку x равно минус четыре. Значение функции действительно стало меньше — минус двадцать четыре, — а мы стали ближе к локальному минимуму.	

		Значение производной в этой точке равно минус двадцати шести: всё ещё меньше нуля. Значит, чтобы найти точку локального минимума, нужно всё ещё двигаться вправо и увеличивать x. Хорошо. Двигаемся дальше: ещё на единицу. Попадаем в точку x равно минус трём. Считаем значение функции и производную: они равны минус двадцати четырём и семнадцати. Смотрите: производная стала больше нуля, а значение функции не изменилось. Это значит, что мы перешагнули точку минимума и попали в место, где функция начала возрастать. Получается, теперь, чтобы попасть в точку минимума, нужно идти левее: уменьшать x. Но если мы уменьшим x на значение delta x, равное единице, мы снова перешагнём точку минимума и попадём обратно в точку минус четыре, где уже были. Значит, шаг delta x нужно уменьшить. Возьмём delta x равно ноль целых пять десятых и сдвинемся влево: получим точку минус три с половиной. Значение функции станет равно около минус двадцати девяти: оно уменьшилось по сравнению с предыдущим шагом, то есть, мы стали ближе к точке минимума. И эта точка, кстати, уже очень близка к реальной точке минимума минус три целых пятьдесят пять сотых. Производная осталась положительной. Значит, мы всё ещё находимся правее точки минимума, и нам нужно снова двигаться влево, чтобы её найти.	

		Этот процесс можно продолжать и дальше. Таким образом мы будем все ближе и ближе подходить к точке локального минимума. Обобщим алгоритм, который мы только что придумали. Итак, у нас есть функция f. Чтобы найти её точку минимума, нужно сделать следующее: 1. Выбираем случайную точку функции x и фиксируем значение шага delta x. 2. Понимаем, в какую сторону нужно идти из точки x, чтобы приблизиться к точке минимума. Для этого вычисляем производную f'(x) и смотрим на знак. Если производная больше нуля, двигаемся влево, уменьшаем значение x. Если производная меньше нуля, двигаемся вправо, к точке x плюс delta x. 3. Снова вычисляем производную функции и смотрим, изменился ли её знак. Если не изменился, всё хорошо: мы идём в верном направлении. Если изменился, значит, мы перешагнули точку минимума: сделали слишком большой шаг delta x. Тогда уменьшим delta x и продолжим идти к точке минимума: снова вычислим производную, поймём, в какую сторону двигаться, и так далее. Понятно, что, двигаясь таким образом, мы будем всё ближе и ближе подходить к точке локального минимума. Действительно: как только мы перешагиваем через точку минимума, мы уменьшаем шаг, начинаем двигаться аккуратнее.	

		Однако ровно в точку минимума мы вряд ли когда-нибудь попадём. Поэтому этот алгоритм ищет точку минимума приближённо: ищет точку, которая будет рядом с точкой минимума. Как, например, на прошлом слайде: мы нашли точку минус три с половиной, которая была довольно близка к минимуму. Понятно, что этот алгоритм находит только одну точку локального минимума: ту, которая ближе всего к начальной точке. Можно попытаться найти несколько точек минимума, проделав этот алгоритм из нескольких разных начальных точек, надеясь, что они приведут к разным точкам минимума. Однако в случае, когда у вас нет доступа к графику функции и вы не знаете, сколько у функции вообще точек минимума, вы не сможете гарантированно найти их все. Несмотря на эти недостатки, алгоритм хорош тем, что может работать с любой функцией, у которой можно взять производную. Всё, что мы делаем, — вычисляем производную функции и смотрим на её знак. Никаких уравнений решать не нужно. Стоит заметить, что с помощью этого алгоритма можно искать и точки максимумов функций. Нужно просто делать шаги на delta x в противоположном направлении: не в сторону убывания функции, а в сторону возрастания. Алгоритм такого вида называется градиентной оптимизацией функции. Градиентная она потому, что мы в процессе используем производную — градиент.	

		Вопрос для обсуждения К этому моменту у вас, наверное, возник вопрос: а как понять, когда в этом алгоритме остановиться? Ведь если мы никогда не попадем в точку минимума, мы можем вечно ходить вокруг нее, всё приближаясь к ней, но никогда не достигая. **Ответ прост:** чаще всего в таких алгоритмах останавливаются тогда, когда значение delta x становится маленьким. Например, одна тысячная. **Смотрите:** если delta x маленькое, это значит, что мы очень близко подобрались к точке минимума: настолько, что при чуть большем значении delta x мы уже перешагнем через неё. На этом можно остановиться: сказать, что мы нашли точку x, которая находится очень близко к точке минимума, и нам этого достаточно. Тогда итоговый алгоритм градиентной оптимизации будет выглядеть так: те же пункты, что на предыдущих слайдах, и к пункту пять добавим оговорку. Мы двигаем точку x до тех пор, пока delta x не станет меньше некоторого числа epsilon. Для начала немного поправим алгоритм градиентной оптимизации. Алгоритм представлен на слайде. Он прекрасно работает и ищет точки рядом с точками минимума, но есть один нюанс. В этом алгоритме мы до начала движения из точки x фиксировали начальное значение delta x. Например, для этой функции в прошлом мы брали delta x, равное	

		единице. Затем, когда перешагивали через точку минимума, начинали delta x уменьшать. **Вопрос для обсуждения** Какое начальное значение delta x стоит брать? Можно всегда брать delta x, равное единице. Но тогда мы с таким delta x будем очень-очень долго идти к точке минимума. Например, рассмотрим функцию f равно x в четвёртой степени. Её точка минимума — ноль. И пусть мы хотим с помощью нашего алгоритма найти эту точку минимума, двигаясь из точки x равно двадцать. И пусть delta x у нас равно единице. Видно, что прежде чем мы подойдём достаточно близко к точке минимума и уменьшим значение delta x, пройдёт как минимум двадцать шагов. Это довольно долго. А что будет, если мы стартуем из точки x равно двести или x равно две тысячи? Кажется, delta x, равное единице, было не лучшим выбором для шага в этом случае. Другой пример: та же функция и стартовая точка x равно ноль целых две десятых. Казалось бы, эта точка близка к точке минимума: достаточно несколько раз аккуратно сдвинуться на небольшое delta x налево, и мы окажемся практически в минимуме. Но если мы будем идти по алгоритму, то сначала сдвинемся в точку минус ноль целых восемь десятых, затем уменьшим delta x и попадём в точку минус ноль целых три десятых, затем в точку ноль целых две десятых и так далее. Будем блуждать	

		около точки минимума, постоянно перепрыгивая через неё. Иногда придуманный нами алгоритм работает не совсем оптимально. Хочется придумать, как выбирать delta x так, чтобы быстрее приходить в точку минимума: чтобы на каждом шаге сдвигаться как можно ближе к минимуму. Давайте подумаем, как можно улучшить выбор delta x. Вспомним о ещё одном свойстве производной: модуль производной функции в точке показывает скорость роста функции в этой точке. А теперь посмотрим на график функции f равно x в четвёртой. Заметим вот что: чем дальше точка находится от точки минимума, тем быстрее растёт или убывает функция в этой точке. Получается, для точек, близких к минимуму, модуль производной будет мал. А для точек, далёких от минимума, модуль чаще всего будет большим. Это правило работает не всегда, но для большинства точек это верно. Можно использовать это свойство, чтобы понимать, насколько большой шаг в сторону минимума мы можем совершить, находясь в той или иной точке. Правило такое: чем больше модуль производной, тем больший шаг по направлению минимума мы можем сделать. Однако модуль производной не говорит нам о том, какой именно по величине шаг нужно выбрать. Шаг, равный модулю производной, не подойдет: он будет слишком большим. Смотрите, в точке x равно 5 производная функции f равна пятистам. Если мы	

		двинемся на шаг delta x, равный пятистам, то улетим далеко по другую сторону от минимума. Решение такое: сдвигаться от точки на величину, равную модулю производной, умноженной на некое маленькое число alpha. Обычно alpha берут равным одной тысячной или ещё меньше. Тогда, если мы каждый раз будем сдвигаться из точки x в сторону минимума на шаг alpha умножить на модуль производной, мы будем двигаться на больший шаг из далёких от минимума точек и на меньший шаг из близких к минимуму точек. По мере приближения к точке минимума модуль производной будет становиться меньше, и шаг delta x тоже будет уменьшаться. Например, при alpha равном одной тысячной, из точки пять мы сдвинемся влево на шаг delta x, равный ноль целым пяти десятым, а из точки двадцать — на delta x, равный тридцати двум. Конечно, такой выбор delta x тоже не идеален. Иногда мы всё ещё будем перепрыгивать точки минимума и тратить большое количество шагов на то, чтобы подойти близко к минимуму. Например, здесь мы из точки двадцать, сдвинувшись на delta x, равное тридцати двум, перепрыгнули минимум. На практике получается так, что в среднем для всех функций такой выбор delta x ускоряет алгоритм градиентной оптимизации. На этом слайде вы видите процесс работы алгоритма для функции 2x умножить на синус x из точки x равной	

		двум с половиной. Здесь альфа равно пяти тысячным. Альфа достаточно мало, чтобы мы не перескакивали через точку локального минимума, а аккуратно приближались к точке минимума с левой стороны. Чем меньше alpha, тем ближе наша функция в итоге сможет подойти к точке минимума. Но при очень малых alpha каждый шаг получается маленьким, алгоритм работает долго. Поэтому нужно выбирать alpha исходя из того, что важнее: точность или скорость. Видно, что в местах, где функция убывает быстрее, алгоритм ускоряется: делает большие шаги от точки к точке по направлению минимума. Причина в том, что в этих местах модуль производной функции выше и шаг delta x получается выше. Кроме того, видно, что в начальной точке два с половиной, которая далека от минимума, модуль производной не такой большой, как в более близких точках, например, около x, равного четырём. Получается, что правило «чем дальше точка от минимума, тем быстрее убывает функция и тем производная больше по модулю» выполняется не всегда. Для многих точек функций это верно, поэтому наш алгоритм градиентной оптимизации в целом работает хорошо и подбирает оптимальные размеры шага delta x. На этой гифке хорошо видно, что наш алгоритм в процессе работы на каждом шаге вычисляет производную функции и спускается точка за точкой к точке минимума. Поэтому этот алгоритм также называется градиентным спуском.	

		Подведём итог и сформулируем обновленный алгоритм градиентного спуска. Суть его осталась той же. Выбираем начальную точку x: мы будем двигаться от неё к точке минимума. Но теперь мы не выбираем значение delta x, а фиксируем значение alpha. Далее на каждом шаге мы будем вычислять производную функции f в текущей точке x, смотреть на знак этой производной и сдвигать точку x в сторону минимума на величину, равную alpha умножить на модуль производной. Повторяем эти пункты, пока шаг, на который мы сдвигаем нашу точку, не станет очень маленьким. То есть, пока мы не будем близки к минимуму. В алгоритме стало на один пункт меньше, чем было до этого. Нам больше не нужно изменять delta x вручную, когда мы перепрыгнули через точку минимума. Delta x теперь зависит от модуля производной функции в каждой точке. Формулу движения x в пункте 2 можно записать проще, в одну строчку. Запишем это так: новое значение x равно текущее значение x минус альфа умножить на значение производной в текущей точке. Значение производной без модуля.	

		Действительно, если f’(x) больше нуля, то значение x уменьшится на величину альфа умножить на модуль производной. То есть, мы сдвинем точку влево, где и находится точка минимума, так как знак производной больше нуля. А если f’(x) меньше нуля, то значение x, наоборот, увеличится на величину альфа умножить на модуль производной. Такая формула пересчёта координаты точки x эквивалентна тому, что было написано на предыдущем слайде. Вы можете подумать и убедиться в этом сами. Мы получили алгоритм градиентной оптимизации, который используется в математике и машинном обучении. Давайте узнаем, как применять такой алгоритм для функции нескольких переменных. Тут, на самом деле, всё просто. Алгоритм будет точно таким же. Вот как он будет выглядеть для функции двух переменных. Фиксируем начальную точку x — координаты x1 и x2. Они могут быть любыми. На каждом шаге вычисляем значения частных производных по x1 и x2 в текущей точке. Меняем значения координат x1 и x2 как новое значение равно старое минус альфа умножить на значение частной производной в старой точке. Повторяем так до тех пор, пока изменения обеих координат не станут очень малы. Как мы помним, частная производная функции по переменной x1 показывает, убывает или возрастает функция по этой переменной. Получается, алгоритм	

		градиентной оптимизации для функции двух переменных на каждом шаге понимает для каждой координаты отдельно, в какую сторону и насколько её нужно сдвинуть, чтобы приблизиться к точке минимума функции. Визуализацию градиентного спуска для функции двух переменных вы видите на слайде. Здесь альфа равно пяти сотым. По мере приближения к точке минимума функция начинает убывать медленнее, шаг алгоритма становится всё меньше. Последний шаг: алгоритм градиентного спуска для функции многих переменных можно переписать в терминах градиента. Градиент — это вектор частных производных функции. Поэтому алгоритм можно записать в таком виде, как на слайде. Мы разобрали градиентную оптимизацию или градиентный спуск для функций одной и многих переменных. На практическом занятии мы постараемся применить её на языке Python. В заключение скажу пару слов о том, как градиентный спуск применяется для обучения моделей машинного обучения. Вспомним линейную регрессию и то, как она получает ответ на элемент датасета. Мы уже говорили о том, что функцию получения ответа регрессии можно рассматривать как функцию от n+1 переменных, где n — количество признаков в данных. На слайде n равно пяти,	

		и линейная регрессия выражает функцию от шести переменных: k1, k2, k3 и так далее. Дальше. В модуле о линейной регрессии мы говорили о том, что задача модели — подобрать такие коэффициенты k, чтобы значение метрики качества MSE на обучающем датасете было минимально. Давайте запишем формулу метрики MSE для первого элемента датасета на экране. MSE от ответа модели y1 с шапочкой и правильного значения ответа y1 равно y1 минус y1 с шапочкой в квадрате. Подставим вместо y1 с шапочкой формулу для неё. Получим, что MSE — это функция от тех же шести переменных k1, k2 и так далее. Задача линейной регрессии — минимизировать MSE, то есть найти такие k1, k2, k3 и так далее, чтобы значение MSE было минимальным. А это ровно та задача минимизации функции многих переменных, которая решается с помощью алгоритма градиентного спуска. Конечно, здесь есть нюансы. При обучении модели мы хотим, чтобы MSE было минимальным в среднем на всех элементах датасета, а не только на одном, как тут. Как этого добиться, мы узнаем в следующих модулях, когда будем разбирать градиентный спуск для нескольких моделей машинного обучения.	

Закрепление изученного материала	15 мин.	**Вопросы для обсуждения:** • Как вычислять производные функций многих переменных? • Что такое градиентная оптимизация? • Как задача минимизации функции связана с машинным обучением?	Педагог организует беседу по вопросам
Этап подведения итогов занятия (рефлексия)	8 мин.	**Вопросы для обсуждения** • Чему я научился? • С какими трудностями я столкнулся? • Каких знаний мне не хватает для более глубокого понимания изученного материала? • Достиг ли я поставленных целей и задач?	Педагог способствует размышлению обучающихся над вопросами
Информация о домашнем задании, инструктаж по его применению	5 мин.	В этом домашнем задании вам предстоит поработать с понятием производной и градиента, а также написать градиентный спуск и его вариации. **Основный пункт задания** В этом задании Вы должны реализовать функцию grad_descent_v1 для нахождения минимума функции с помощью градиентного спуска. Вход фукнции. Функция func, которую нужно оптимизировать Ее производная deriv. В данном пункте производная вам дана, вычислять ее самостоятельно не нужно. (*) Начальная точка start. Выход функции -- точка локального минимума.	

		Для вашего удобства мы написали функцию для отрисовки траектории градиентного спуска. В реализации градиентного спуска можете предполагать, что на вход подаются функции с единственным, глобальным минимумом. Перед тем, как писать код, ответьте себе на следующие вопросы: Как понять, что пора остановиться? Это может зависеть от градиента или расстояния между двумя соседними шагами алгоритма, так и от числа уже выполненных итераций. Как правильно менять величину шага (learning rate) от итерации к итерации?	

Рекомендуемые ресурсы для дополнительного изучения:

1. Частная производная. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Частная_производная.
2. Градиент. [Электронный ресурс] – Режим доступа: https://ru.abcdef.wiki/wiki/Gradient.
3. Обзор градиентных методов в задачах математической оптимизации. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/413853/.
4. Основы линейной регрессии. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/514818/.