Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Поиск точки минимума функции с помощью градиентного спуска

# Функции одного аргумента и их точки минимума

## Точки минимума функции

Точка *a* называется **локальным минимумом** функции *f(x)*, если существует окрестность точки *а*, что значение *f(a)<f(b)* для любой точки *b* из этой окрестности.

Точка *a* называется **глобальным минимумом** функции *f(x)*, если *f(a)<f(b)* для любой точки *b* из области определения функции.

Можно не уметь искать точки максимума, так как точки максимума функции *f(x)* являются точками минимума функции *-f(x)*. И наоборот.

Ищем точки минимума:

* Есть функция *f(x)*.
* Находим ее производную *f’(x)*.
* Решаем уравнение *f’(x)=0*.
* Корни этого уравнения – кандидаты на звание «локального минимума».

## Метод градиентного спуска

Обычный мячик под действием силы тяжести скатывается в самую глубокую область поверхности.

Выберем длину шага *h* (например, *h=0.01*). Сделав шаг такой длины наш шарик будет останавливаться и думать, в какую сторону ему дальше катиться.

Пусть шарик находится в точке с координатой *а*. Тогда новая *а* точка равна:



После этого процесс повторяется, мы получаем новую позицию для шарика и т. д.

В общем случае этот процесс можно описать формулой. Пусть *an* - координата шарика после *n* шагов. Тогда координата шарика на следующем шаге равна:

a_{n+1}:=a_n-h\cdot f'(a_n)

Стандартное значение шага *h=0.01.*

Когда нужно остановиться?

Когда значение *f’(an)* будет (примерно) равно *0*. Либо, когда пройдет заданное заранее максимальное число итераций.

Такой способ нахождения точки минимума называется **градиентным спуском (ГС)**. И он работает, даже когда школьный метод ломается.

Недостатки ГС:

* всё зависит от расположения начальной точки *a0*.
* слишком маленький шаг *h* плох;
* слишком большой шаг *h* плох;
* точность зависит от длины шага *h*;
* плохое поведение на плато функции *f(x)*.
* взрыв градиента, когда значение *f’(a)* очень велико.

Преимущества ГС:

* Других общих методов нахождения точек минимума у нас нет.
* Проблемы с выбором *h* вполне решаемы: мы можем изменять величину *h* на каждой итерации. На первых итерациях *h* может быть достаточно большим, и потом плавно уменьшаться.
* Метод может работать, даже когда производная неизвестна!

# Точки минимума функций многих переменных

## Функции многих переменных (ФМП)

Пример. «Площадь прямоугольника» зависит от длин двух его сторон:



А как можно представлять себе ФМП? График в виде кривой не годится.

Они представляются в виде поверхностей: двумерных (их еще можно нарисовать) или многомерных.

Точка *(a1,a2 )* является точкой **локального минимума** ФМП *f(x1,x2)*, если в некоторой окрестности этой точки выполнено неравенство



для всех точек *(x1,x2 )* **из данной окрестности*.***

Точка *(a1,a2 )* является точкой **глобального минимума** ФМП *f(x1,x2)*, если 

для всех точек *(x1,x2 )* **из области определения функции** *f(x1,x2 ).*

Точка *(a1,...,an )* является точкой **локального минимума** ФМП *f(x1,...,xn)*, если в некоторой окрестности этой точки выполнено неравенство



для всех точек *(x1,...,xn )* из данной окрестности*.*

Точка *(a1,...,an )* является точкой **глобального минимума** ФМП *f(x1,...,xn)*, если для всех точек *(x1,...,xn )* из области определения функции *f(x1,...,xn).*

Нужно уметь находить у ФМП глубокие минимумы. В идеале – найти глобальный минимум.

## Производные функций многих переменных

Пусть *f(x1,x2 ) -* функция от двух аргументов. Оказывается, у нее существует не одна, а целых две производных:

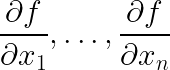
**частная производная** по аргументу *x1*,

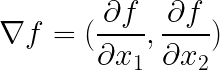


**частная производная** по аргументу *x2.*

Частная производная (ЧП) считается по обычным правилам **в предположении, что все остальные аргументы функции являются константами.**

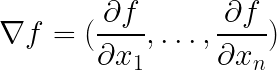
Если ФМП *f(x1,...,xn )* зависит от *n* аргументов, то у нее будет *n* частных производных (ЧП)



Если точка *(a1,a2 )* является точкой локального минимума ФМП *f(x1,x2),* то **все ЧП равны нулю в этой точке**! Для функций одного аргумента, это, конечно же, тоже выполнено.

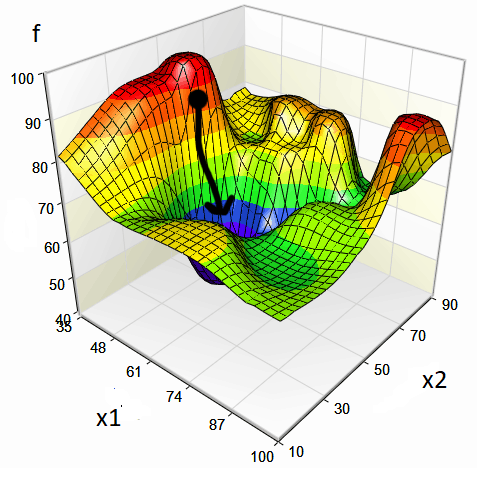
Вектор для ФМП *f(x1,x2 )* называется **градиентом**.

Для ФМП от *n* переменных и градиент подлиннее:



## Градиентный спуск для функций многих переменных

Мы будем снова моделировать спуск шарика. Но на этот раз движется по поверхности, а не по одномерному графику функции.



Глобальный минимум не обязательно будет достигнут при таком спуске.

Позиция шарика перед началом спуска существенно влияет на результат (это было справедливо и для функции одной переменной).

ГС для ФМП ничем **принципиально не отличается** от ГС для функции одного аргумента («Математика - сила!»). Вспомним ГС для одного аргумента.

Выберем длину шага *h*. Пусть шарик находится в точке с координатой *а*. Тогда новая *а* точка равна:



После этого процесс повторяется, мы получаем новую позицию для шарика и т.д.

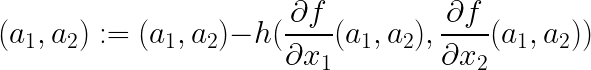
А теперь просто заменим числа на векторы в формуле для ГС:

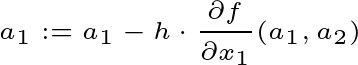


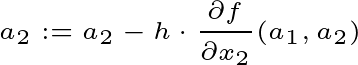


Будет так:

Выберем длину шага *h (*например, *h=0.01)*. Пусть шарик находится в точке с координатой . Тогда новые координаты равны:







После этого процесс повторяется, мы получаем новую позицию для шарика и т.д.

Если – позиция нашего шарика на *n*-м шаге, то новая позиция равна:



Стандартное значение шага *h=0.01.*

**Градиент (как вектор) показывает направление наибольшего возрастания функции.**

А мы специально идем в противоположном направлении, чтобы спуститься как можно глубже (из-за этого в формуле ГС стоит знак «минус»).

## Модификации градиентного спуска

### Модификации с шагом h

**Как понять, что где-то есть впадины поглубже?**

**Идея**: на первых итерациях градиентного спуска шаг *h* может быть большим, а потом нужно его плавно уменьшать.

По каким формулам уменьшают длину шага *h*?

**Первая формула:**

**Смысл:** Перед началом ГС у нас шаг *h0*. После каждой итерации шаг равномерно уменьшается**.**

**Сюрприз:** после *Т* итераций (*Т* - константа, задаваемая перед ГС) шаг станет равным *0*. ГС останавливается.

**В общем, у вас есть *(T+1)* шагов, чтобы найти хороший минимум.**

**Вторая формула:**

**Смысл:** Перед началом ГС у нас шаг *h0*. После каждой итерации шаг **плавно** уменьшается**.** Константа *Т* (задаваемая перед ГС) определяет скорость уменьшения длины шага *h*. Чем больше *Т*, тем медленнее убывает *h*.

### Метод Momentum

**Идея**: до сего момента шарик (который катится по поверхности) был без каких-либо физических характеристик. Так давайте припишем ему некоторую массу и после каждого шага будем вычислять физическую скорость шарика.

**Зачем**: шарик с большой массой и скоростью будет легко миновать неглубокие минимумы и по инерции пролетать плато. Осталось лишь выразить всю эту физику с помощью формулы.

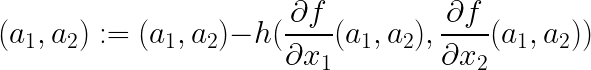
**Пусть , тогда вычисление новой позиции шарика при ГС происходит по формуле:** .

**Смысл** : это как бы масса (инерция) шарика. Этот параметр определяет, насколько сильно новое направление движения зависит от пройденного пути.

Если , то мы имеем **классический** ГС.

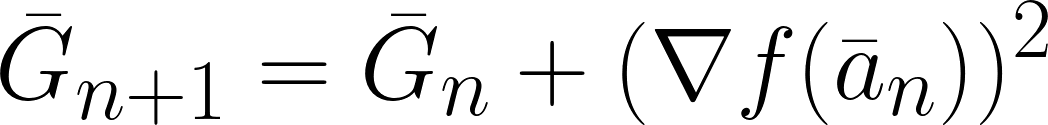
### Adagrad

**Проблема** («градиентный коммунизм») по всем координатам градиент умножается на один и тот же коэффициент *h*.



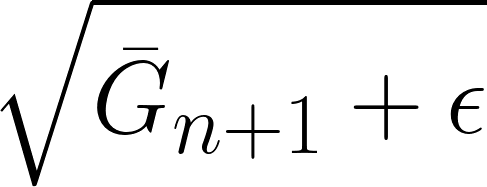
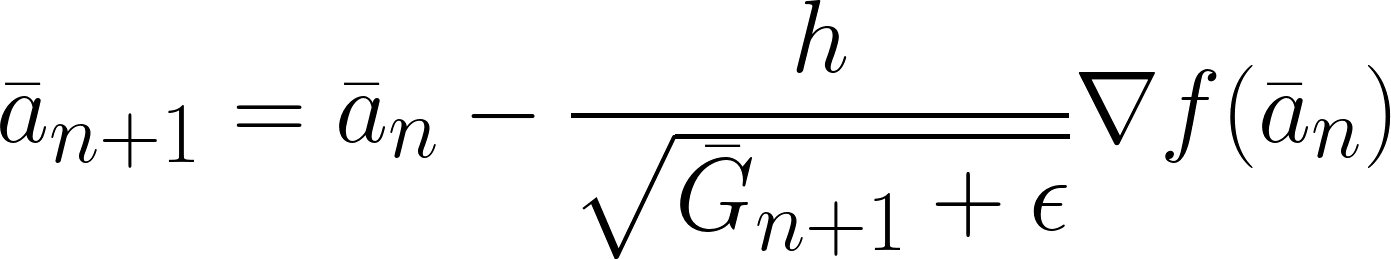
Это не всегда оправдано, поскольку по одной координате можно сдвигаться более длинными шагами, а по другим координатам – нужны более мелкие шаги.

**Идея**: градиент умножать на разные числа, в зависимости от направления.

Мы накапливаем информацию о градиентах со всех предыдущих позициях шарика **(здесь покоординатное возведение в квадрат).**

Короче говоря, это сумма квадратов градиентов с прошлых итераций.

**Новая позиция шарика:**

****

(здесь градиент покоординатно делится на вектор )

Здесь ε>0 – маленькое число, нужное для избегания деления на 0.

### Adam

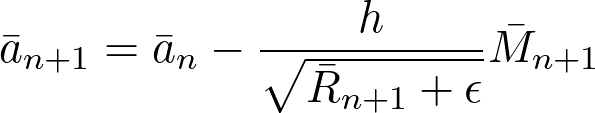
**Недостаток Adagrad**: **градиенты за все прошлые шаги вносят одинаковый вклад в сумму**  .

Но это не всегда логично. Градиенты с далёких шагов могут быть нерелевантны в текущей точке.

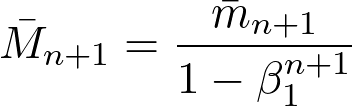
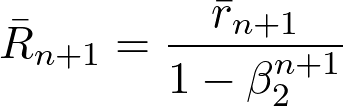
Вы же не используете свой давний опыт альпинизма при перемещении по городской улице!

**Второй недостаток:** **выражение неограниченно растёт, слишком быстро уменьшая шаг.**

**Идея**: а давайте запоминать **взвешенные суммы** градиентов (и их квадратов) с прошлых шагов.



Как будет видно дальше, вектора *Mn+1 (Rn+1)*хранят информацию о предыдущих градиентах (квадратов градиентов).

, 

где – числа-константы, их смысл см. далее.

\bar{m}_{n+1}=\beta_1\bar{m}_{n}+(1-\beta_1)\nabla f(\bar{a}_n)

\bar{r}_{n+1}=\beta_2\bar{r}_{n}+(1-\beta_2)(\nabla f(\bar{a}_n))^2

\bar{m}_0=\bar{r}_0=\bar{0}

Т.е. вектора *mn+1,rn+1* накапливают информацию о взвешенном среднем градиента (квадрата градиента) с предыдущих позиций ГС.

Числа *𝛃1, 𝛃2* показывают, **насколько сильно забываются давние значения** градиента и квадрата градиента. Чем меньше *𝛃1, 𝛃2*, тем быстрее старые значения забываются. Если *𝛃1, 𝛃2=0,* то мы вообще не используем никакую историю.

Обычно берут: *𝛃1=0.9*, *𝛃2=0.999*.