Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Графы вычислений и алгоритм обратного распространения ошибки

# Сложные функции одного аргумента

Пусть *f(x), g(x)* – две функции аргумента *x*. Тогда функция *h(x)=f(g(x))* называется **сложной функцией** (или **суперпозицией** функций *f,g*).

Функция *h(x)* получается из *f(x)* заменой каждого вхождения буквы *x* на выражение *g(x)*. Такой вот копипаст.

Например, если *f(x)=xsin(x), g(x)=x2*, то *h(x)=f(g(x))=x2sin(x2).*

Со школы известно правило: если *h(x)=f(g(x))*, то *h’(x)=f’(g(x))g’(x).*

Например, есть функция

Чтобы найти ее производную, надо представить ее как сложную функцию: *h(x)=f(g(x))*, где *f(x)=ex, g(x)=x2*. Тогда получим: *f’(x)=ex, g’(x)=2x,* и поэтому 

Аналогично для  получаем *h(x)=f(g(x))*, где *f(x)=1/x, g(x)=sin x.* Тогда *f’(x)=-1/x2, g’(x)=cos x,* и поэтому 

Если в выражении *h(x)=f(g(x))* функция *g(x)=g1(g2(x))* снова является сложной, то процесс рекурсивно повторяется

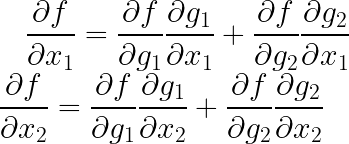
*h’(x)=f’(g(x))g’(x)=f’(g1(g2(x)))g’1(g2(x))g’2(x)*

Таким образом, вычислять производную сложной функции можно с помощью вычисления производных всё более и более глубоких выражений.

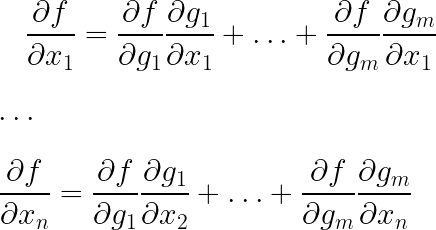
# Сложные функции многих переменных

Пусть *f(x1,x2 ), g1(x1,x2 ), g2(x1,x2)* – функции аргументов *x1,x2*. Тогда функция *f(g1(x1,x2 ),g2(x1,x2 ))* называется **сложной функцией** (или **суперпозицией** функций *f,g1,g2*.

**Формула для ЧП сложной функции *h(x1,x2):***



**Если , то**

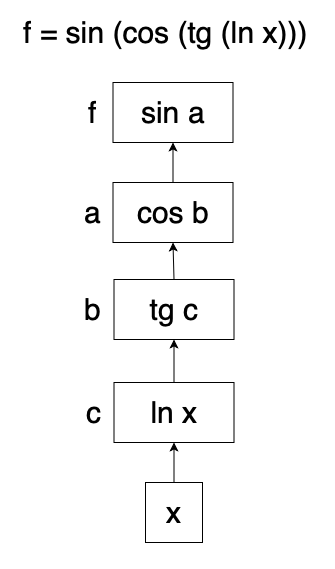


# Графы вычислений

Как вы видели, ФМП являются сложными выражениями, одни фрагменты выражения вкладываются в другие. Как визуально представлять процесс вычисления значения функции? В виде **графа!** Он будет называться **граф вычислений**.

У графа есть узлы и стрелки между ними. В каждом узле стоит операция, применяемая к выражениям, входящим в узел. **Узлы, в которые не входят стрелки, называются листьями,** и в них стоят переменные. Еще есть **корень –** единственная вершина, из которой ничего не выходит.

**Пример графа вычислений.** Аргумент х – в листе, корень соответствует всей функции. Граф определяет выражение *f(x)=sin(a)=sin(cos(b))=sin(cos(tg(c)))=sin(cos(tg(ln(x)))).*



# Алгоритм обратного распространения

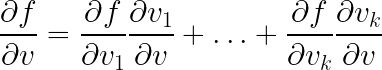
По графу вычислений можно вычислить не только значение функции в точке, а найти выражения для всех её ЧП.

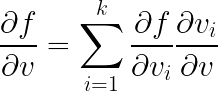
Для этого в графе нужно **идти от корня к листьям**. Это моделирует вычисления производной по формуле дифференцирования сложной функции.

Этот алгоритм называется **алгоритмом обратного распространения (back propagation, backprop).**

Основные правила **алгоритма backprop:**

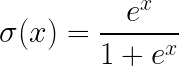
* Начать с корня, посчитать ЧП по всем буквам, входящим в корень.
* Взять узел графа *v* такой, что для всех его соседних узлов сверху *v1,...,vk* ЧП уже подсчитаны.
* Тогда ЧП считается по формуле:

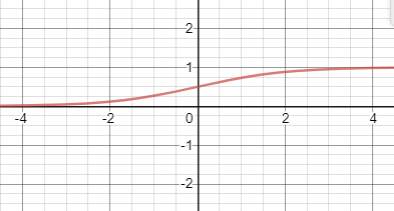




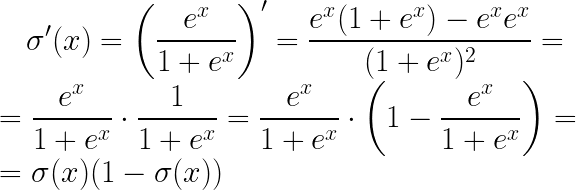
# Сигмоида

В теории нейронных сетей целая эпоха связана с этой функцией.

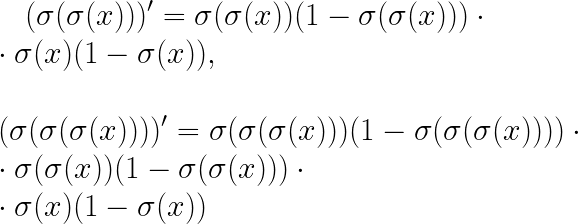




С помощью школьных правил дифференцирования получим:

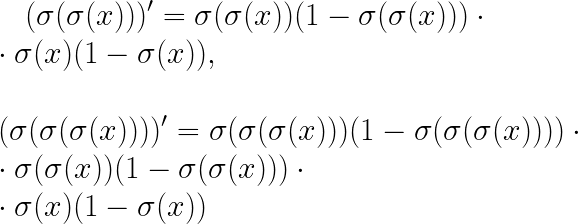


Дифференцируя как сложную функцию, получим:



Чем больше суперпозиция, тем больше множителей в производной.

Но все эти множители положительны и строго меньше 1, а произведение таких чисел стремится к 0 с возрастанием числа множителей.



**Если для почти всех значений *a* выполнено ,** то это означает трудности с нахождением минимума функции с помощью ГС.

Иными словами, функция *f(x)* почти всюду имеет относительно ровное плато. В этом случае ГС не сможет существенно изменять положение точки, так как

Из последующих глав будет понятно, почему:

* сигмоиду сейчас не используют в построении искусственных нейронных сетей;
* тем не менее её когда-то начинали использовать в нейронных сетях, и в те времена считалось, что сигмоида – это круто!