Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Технологии тренировки глубоких нейронных сетей

# Инициализация весов и нормализация данных

## Проблемы НС

Общая схема тренировки НС:

1. Задать архитектуру НС.
2. Выписать функцию потерь *L(w)* по заданной ТВ.
3. Найти точку минимума функции потерь, которая даст оптимальные значения весов НС.
4. После этого НС может предсказывать значения для новых объектов (которых уже нет в ТВ).

**Проблемы НС.** Начальная точка ГС может быть выбрана неудачно.

Мы можем оказаться сразу на плато или попасть в зону бесконечного спуска. Для борьбы с этим были придуманы нетривиальные алгоритмы. Например, с попаданием на плато борются алгоритмы инициализации весов и нормализация данных. С бесконечным спуском косвенно борется регуляризация.

**Проблемы глубоких НС:**

1. Затухание сигнала при прохождении через сеть. Методы борьбы: алгоритмы инициализации весов и нормализация по мини-батчам.
2. Переобучение: кода НС не пытается извлечь закономерность из данных, а тупо запоминает всю ТВ с её ответами (либо извлекает несуществующую закономерность). Метод борьбы: регуляризация и дропаут.

## Инициализация весов

В ГС нужно выбрать начальную точку для спуска, то есть начальные значения весов НС. Обычно эта точка выбирается случайно. Но от нее существенно зависит успех ГС. По этой причине веса нужно инициализировать правильно.

Казалось бы, задача невозможная:найти такие начальные значения весов, которые лежат вблизи глубокой точки минимума функции потерь, не используя информации о функции потерь.

От начальных значений весов действительно много что зависит.

Если начальные значения весов подобраны неудачно, то мы окажемся на плато, откуда ГС никуда не пойдёт.

Начальные значения весов должны быть далеко от плато!

Иначе на первых итерациях ГС мы столкнемся с затуханиями градиента и процесс самопроизвольно остановится.

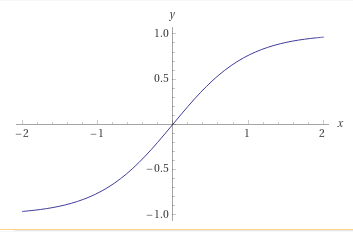
Во-вторых, метод инициализации весов должен хорошо работать для глубоких НС.

Нужно найти правило следующего рода: «генерировать начальные значения весов сети с помощью случайных чисел из интервала *[a,b]*» (числа *a, b* нужно будет найти).

Будем наблюдать за НС.Есть пара полносвязных слоев. Считаем, что в них одинаковое число нейронов *n*. Следовательно, в каждый нейрон правого слоя входит *n*связей (и у каждой свой вес).

Будем считать, что ФА *f* у всех нейронов одинаковая и удовлетворяет условиям:

Например, таким условиям удовлетворяет tanh(x):



Также будем считать, что **веса-смещения всех нейронов равны 0** (или близки к нему).

Какими же должны быть веса-связи между нейронами?

Допустим, что каждый из нейронов левого слоя **до** применения ФА имеет выходное значение в диапазоне *[-a,a]*.

Тогда из этих нейронов выходят значения из *[f(-a),f(a)]=[-f(a),f(a)]*.

Допустим, что веса связей между слоями принадлежат отрезку *[-b,b]* (число *b* мы как раз хотим найти). Тогда каждый из нейронов правого слоя получает на вход число из диапазона *[-nbf(a),nbf(a)].*

Поскольку смещения равны 0, то в итоге получаем, что при переходе между слоями сигнал из интервала ***[-a,a]*** переходит в сигнал из интервала ***[-nbf(a),nbf(a)]****.*

Какой отрезок больше и что это значит?

Из свойств ФА получаем, что нужно сравнивать отрезки: *[-a,a]* *[-nba,nba].*

Если *nba<a,* то это **плохо**! Диапазон сигналов нейронов уменьшается, и при достаточно глубокой сети последний слой будет выдавать константу 0.

Получили пока одно условие: – **запомним его.**

Теперь посмотрим, какие значения ЧП при back-prop возникают в нашей сети. Пусть все y*(x)* – значение, выдаваемое некоторым нейроном N y из левого слоя; *y1(x),...,yn(x)* – выходы из нейронов правого слоя.

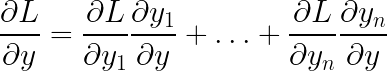
Значения *y1(x),...,yn(x)* зависят от y*(x):*

*yi (x)=f(...+y(x)wi+...)*

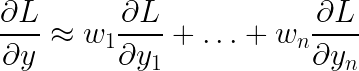
где *wi –* вес связи между нейроном N и *i*-м нейроном правого слоя.

И поэтому:

Продемонстрируем это на примере.

Из выбора ФА *f* получаем:

Тогда ЧП функции потерь *L* равны:



Их можно оценить как

Веса *wi* принадлежат интервалу *[-b,b].* Пусть ЧП с правого слоя принадлежат некоторому интервалу *[-c,c]*.

Тогда

Итак, ЧП с правого слоя принадлежат интервалу *[-c,c],* а ЧП с левого слоя принадлежат интервалу *[-nbc,nbc].*

Если *nbc<c,* то это **плохо**! Это означает, что, когда алгоритм back-prop дойдёт до первого слоя сети, значения ЧП будут почти равны нулю (затухание градиента). Таким образом, мы снова получаем такое же условие



**Вывод:** начальные значения весов-связи между нейронами полносвязного слоя нужно задавать случайными числами из диапазона *[-b,b]*, где ,

*n* - количество нейронов в каждом из двух полносвязных слоев.

В частности, (и для простоты) **можно выбрать диапазон** *[-1/n,1/n].*

А если в слоях разное число нейронов? В этом случае начальные значения весов-связей между нейронами полносвязного слоя нужно задавать случайными числами из диапазона *[-1/n,1/n]*, где *n=(n1+n2 )/2* – среднее арифметическое числа нейронов в соседних слоях.

Выше была объяснена (весьма неформально) суть инициализации Ксавье весов НС: веса-связи инициализируются случайными числами из интервала [*-1/n,1/n*], веса-смещения инициализируются нулями.

На самом деле веса генерируются не из интервала *[-1/n,1/n],* а с помощью равномерного распределения с дисперсией *1/n.*

Практика показывает, что инициализация Ксавье улучшает обучение НС и повышает точность предсказания.

## Нормализация данных

Нормализация – это приведение данных к одному масштабу.

Например, часть данных у вас выражена в кг с диапазоном значений [0,100], а часть данных – в градусах Цельсия c возможными значениями [-30,40]. После нормализации все данные становятся безразмерными, и принадлежащими одному общему диапазону.

Оказывается, данные ТВ нужно нормализировать перед тренировкой НС.

Необходимость нормализации следует из наших рассуждений об алгоритме инициализации весов.

Там мы предполагали, что нейроны некоторого слоя НС до применения ФА выдают значения из диапазона *[-a,a]*.

Это предположение должно быть верным для любого слоя НС, в том числе и для входного слоя.

Но позвольте! Признаки объектов *xi* могут иметь разный масштаб и природу (например, *x1={пол человека}, x2={рост}*). Как их загнать в один диапазон?

Идея нормализации состоит в том, чтобы все признаки объекта привести к одному масштабу. Нормализация осуществляется с помощью «умных» формул, ниже будут указаны два самых популярных способа нормализации данных.

По сути нормализация – это преобразование нецелевых столбцов в таблице с данными.

**1-й способ нормализации**

Для каждого столбца X делаем следующее:

Пусть *X={x1 ,...,xm } –* все данные из столбца *X*, *a=min{x1 ,...,xm}, b=max{x1,...,xm } -* максимальное и минимальное значение в столбце.

Тогда нормализованный столбец состоит из значений:



После нормализации значения в столбце *X* принадлежат интервалу *[0,1].*

**2-й способ нормализации**

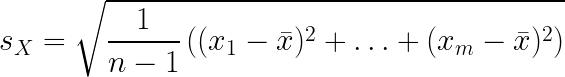
Этот способ требует знания некоторых понятий из математической статистики.

Пусть *X={x1 ,...,xm } –* все данные из столбца *X.* Тогда величина называется средним значением признака Х.

**Задачка**: если из всех значений признака *Х* вычесть его среднее значение, а потом вычислить среднее значение нового признака, то оно будет равно **0**.

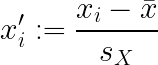
Пусть *X={x1 ,...,xm } –* все данные из столбца *X.*

**Величина**



**называется отклонением признака *Х*.**

Тогда нормализованный столбец состоит из значений:



Можно доказать, что нормализованный столбец имеет среднее значение 0 и отклонение 1.

С вероятностью 95-99% значения нормализованного столбца лежат в диапазоне *[-3,3]* (**правило трёх сигм**).

# Регуляризация

## Постановка проблемы

После обучения НС у нее могут оказаться **астрономически большие**

(по абсолютной величине) веса.

**Почему это плохо?**

- когда у функции *FNN(x)* большие веса *wi* , предсказание НС становится неустойчивым. Небольшое изменение в значениях нецелевых признаков приводит к большому изменению предсказанного признака.

- (философский) большие числа сами по себе менее вероятны для описания природы, чем небольшие числа.

- если задача предсказания допускает несколько решений, то нужно выбирать максимально простое.

Как запретить НС в процессе ГС получать слишком большие веса?

И главное не переусердствовать в этом! Возможно, в конкретной задаче большие веса действительно нужны.

**1-й совет**: ГС нужно начать с маленьких весов.

Для этого использовать:

а) **нормализацию данных**;

б) **инициализацию Ксавье**.

**2-й совет**: применить эту самую регуляризацию.

Итак, нужно запретить НС искать слишком большие веса. Это требование нужно **вписать в функцию потерь**. Но как?

Пусть *L(w)* – «обычная» функция потерь (то есть сумма квадратов ошибок).

Мы приделаем ей «хвост» и получим регуляризированную функцию потерь:

*Lreg(w)=L(w)+C(w12+...+wk2),*

где *С* – некоторая **неотрицательная константа** (её надо самим выбрать), а в скобках стоит сумма квадратов всех весов (веса-связи и смещения) нашей НС. Затем как обычно ищем минимум функции *Lreg(w).*

**Константа *С* –** это вес весов нейронной сети.

Она отражает **компромисс** между двумя противоположными задачами:

* искать минимум «старой» функции потерь L(w);
* уменьшать веса сети.

Чем больше *С*, тем выше приоритет второй задачи. Чем ближе к нулю *С*, тем выше приоритет первой задачи.

В частности, при *С=0* мы получаем обычную процедуру тренировки НС.

## Второй вид регуляризации

Оказывается, есть и другой вид регуляризации.

Ранее мы изучали регуляризированную функцию потерь ***Lreg(w)=L(w)+C(w12+...+wk2)*** *–* такой вид регуляризации называется **L2-регуляризацией**.

Но можно составить такую функцию потерь: ***Lreg(w)=L(w)+C(|w1|+...+|wk|)*** *–* это **L1-регуляризация.**

L1- и L2-регуляризации по-разному уменьшают величины весов НС:

* в L2-регуляризации с ростом константы С веса **плавно стремятся к нулю**;
* в L1-регуляризации с ростом константы С все **веса постепенно зануляются**.

# Дропаут и нормализация по мини-батчам

## Дропаут

**Дропаут** – это преднамеренная деактивация части нейронов на шаге обучения. Это реинкарнация идеи **ансамбля** – очень популярной вещи в машинном обучении.

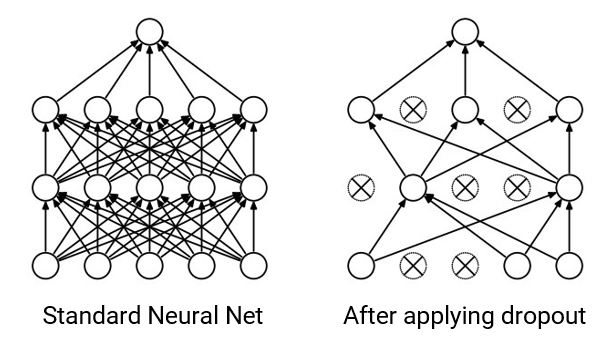
Если у вас есть один мозг (биологический или искусственный), то он может иногда ошибаться в принятии решений.

А вот если у вас будет целый **ансамбль мозгов**, то можно выслушать ответ каждого мозга и в качестве окончательного решения взять «усредненный ответ». Точность ансамбля выше точности каждого из его членов.

На идее ансамбля построены многие классические модели МО и ИИ. Например, **RandomForest**. Там идея такая: у вас есть деревья решений, а вы из них составляете ансамбль (в данном случае: лес).

Лес принимает коллегиальное решение. Каждое дерево в лесу строится по случайному подмножеству нецелевых признаков и случайному подмножеству объектов ТВ.

Мы можем условно разделить НС на части. Каждая часть будет формально считаться автономным мозгом. Ответ всей НС будет фактически усредненным значением нескольких ее подсетей.



**Дропаут** – это преднамеренная деактивация части нейронов на шаге обучения (а ещё это деталь велосипеда).

Перед тем, как произвести один шаг ГС, мы случайным образом **временно исключаем** из НС часть нейронов. Формально возникает новая сеть с новой функцией *FNN(x)* и новой функцией потерь *L(w)*. По этой функции потерь осуществляется один шаг ГС и пересчитываются веса сети.

**Примечание**: если после исключения части нейронов НС **перестает быть связной**, то есть в ней нет пути от входного слоя к выходу, то мы формируем новое множество нейронов для исключения (пока сеть без этих нейронов не станет связной).

**Фиксируется число *p*** – вероятность «смерти» нейрона (обычно *р* берут из интервала [0.2, 0.5] – слишком большие и слишком малые значения *p* – плохо). На каждой итерации ГС для каждого нейрона НС проводится **случайное испытание** – исключать его или нет.

Формируется НС из «выживших» нейронов, для неё выписывается своя функция потерь. После этого осуществляется **одна** итерация ГС и дропаут повторяется.

Когда НС будет натренирована с помощью дропаута, и мы захотим проверить предсказание сети на новом объекте *А*, то надо не забыть следующее:выход каждого нейрона домножается на число *р* (вероятность исключения этого нейрона).

На входном слое тоже можно (и нужно) применять дропаут. Фактически – это **выбор случайного подмножества признаков для обучения** на каждой итерации ГС.

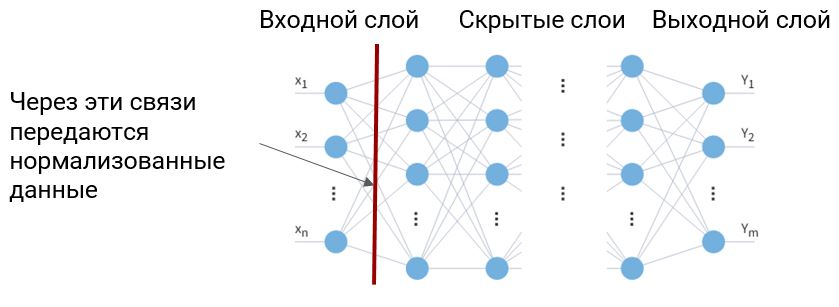
А дальше ГС на каждой итерации минимизирует либо функцию потерь *L1(w)* либо *L2(w)* (в зависимости от того, какой вход удаляется)*.*

Идея тренировать модель машинного обучения по **случайному подмножеству признаков** не нова. Это применяется в классической модели Random Forest.

## Нормализация данных во внутренних слоях сети

В предыдущих лекциях говорилось, что данные для входа НС нужно **нормализовать** (привести к одному масштабу).

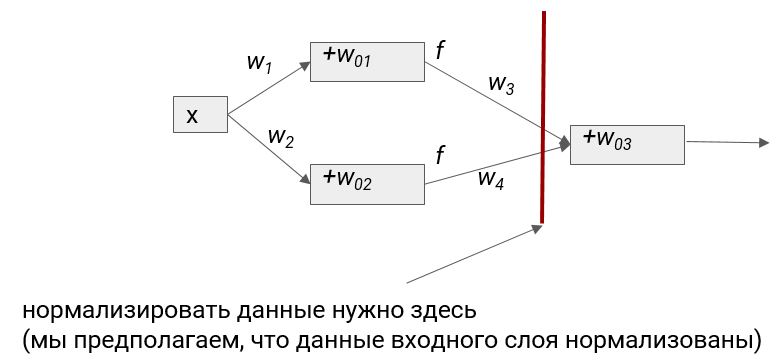
Иными словами, данные, передаваемые между входным слоем и первым внутренним слоем **нормализованы**.



А почему бы не нормализовать данные, передаваемые между **каждой парой соседних слоев?**

1. Какая принципиальная разница между входным слоем и внутренними слоями сети? Первый внутренний слой можно считать входным слоем при преобразованных данных.
2. Это следует из инициализации Ксавье. Перед ГС веса генерируются из **единого интервала** *[-a,a]*. Следовательно, сигнал, передаваемый по всем связям должен быть одного масштаба. Иначе НС будет долго обучаться, ей потребуется много времени, чтобы самой подогнать веса к масштабу данных.

**Общее правило**: нужно нормализовать данные, входящие в каждый слой НС.



## Тонкая настройка нормализации

Почему мы нормализуем данные, делая их среднее значение равным 0 и отклонение равным 1? **Почему числа 0 и 1** **такие особенные**?

Это обычные числа, поэтому можно нормализовать данные к другим значениям – если сильно хочется. Но к каким?

Введем два **настраиваемых** параметра НС: *c,s*, и входы каждого нейрона в сети будут нормироваться «средним» *с* и «отклонением» *s*.

Параметры *c,s* попадут в выражение для FNN(x) и в выражение для функции потерь L(w).

Когда будет найдена точка минимума функции потерь, **мы получим оптимальные значения** *c,s.* И мы поймём, какое среднее и отклонение в данных оптимально для нашей задачи. Это неформальное объяснение.

Пусть *x’* - отнормированное входное значение нейрона. Выражение *x’* участвует в *FNNN(x)* и функции потерь.

А теперь давайте на вход слою НС подавать не *x’*, а выражение  *ax’+b,* где *a,b* - новые параметры сети, оптимальное значение которых будет искаться во время обучения.

**Справка**: если среднее значение и отклонение величины *х’* были равны 0 и 1, то среднее значение и отклонение величины *ax’+b* будут равны *b* и *a* соответственно.

Модифицируем изученный ранее пример с использованием параметров *a,b.*

## Нормализация по мини-батчам

Нормализация данных во всех слоях НС приводит к **сильному усложнению** вида функции потерь.

Более того, *LN(w)* содержит выражения для среднего и отклонения по объектам ТВ. Эти выражения являются суммами с числом слагаемых равным объёму ТВ. И эти выражения **встречаются в LN(w) много раз**.

Если ТВ огромная, то вычислить градиент функции потерь **чрезвычайно трудоемко**.

Допустим, вам нужно вычислить **средний рост людей на планете.** Измерять абсолютно всех – трудоемко и нереализуемо.

Альтернатива - выборочный метод. Нужно измерить небольшую группу **случайно выбранных** людей, и их средний рост **не будет сильно отличаться** от среднего роста всего человечества.

**Идея:** если трудно считать среднее и отклонение по всей ТВ, то можно взять небольшой набор случайно выбранных объектов из ТВ и подсчитать среднее и отклонение по ним. Такой случайно выбранный набор объектов называется **мини-батчем**.

Нормализация по мини-батчам оказалось суперуспешной технологией для тренировки больших НС. В большинстве ситуаций если используется **батч-нормализация**, то можно не применять ни **дропаут** ни **регуляризацию**.

В общем, современные нейронные сети – это:

**L** – **loss function** (функция потерь);

**G** – **gradient descent** (градиентный спуск);

**B** – **batch-normalization** (батч-нормализация);

**T** – **training set** (тренировочная выборка).