Тятя! Тятя! Нейросети заменили продавца!

Ппилиф Ульянкин

Аннотация

В этой виньетке собрана коллекция ручных задачек про нейросетки. Вместе с Машей можно попробовать по маленьким шажкам с ручкой и бумажкой раскрыть у себя в теле несколько чакр и немного глубже понять модели глубокого обучения[1].

Вместо введения

> Я попала в сети, которые ты метил, я самая счастливая на всей планете.
>
> *Юлианна Караулова*

Однажды Маша услышала про какой-то Машин лёрнинг. Она сразу же смекнула, что именно она — та самая Маша, кому этот лёрнинг должен принадлежать. Ещё она смекнула, что если хочет владеть лёрнингом по праву, ни одна живая душа не должна сомневаться в том, что она шарит. Поэтому она постоянно изучает что-то новое.

Её друг Миша захотел стать адептом Машиного лёрнинга, и спросил её о том, как можно за вечер зашарить алгоритм обратного распространения ошибки. Тогда Маша открыла свою коллекцию учебников по глубокому обучению. В каких-то из них было написано, что ей никогда не придётся реализовывать алгоритм обратного распространения ошибки, а значит и смысла тратить время на его формулировку нет[2]. В каких-то она находила слишком сложную математику, с которой за один вечер точно не разберёшься.[3] В каких-то алгоритм был описан понятно, но оставалось много недосказанностей[4].

Маша решила, что для вечерних разборок нужно что-то более инфантильное. Тогда она решила поскрести по лёрнингу и собрать коллекцию ручных задачек, прорешивая которую, новые адепты Машиного лёрнинга могли бы открывать у себя диплернинговые чакры. Так и появилась эта виньетка.

[1] Ахахах глубже глубокого, ахахах

[2] Франсуа Шолле, Глубокое обучение на Python

[3] Goodfellow I., Bengio Y., Courville A. Deep learning. – MIT press, 2016.

[4] Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей - Санкт-Петербург, 2018.

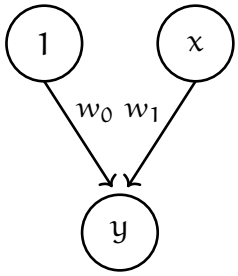
Содержание

Листочек 1: всего лишь функция

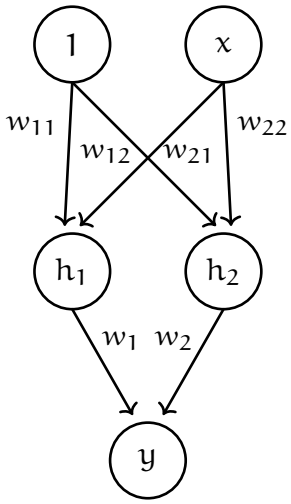
> Ты всего лишь машина, только имитация жизни. Робот сочинит симфонию? Робот превратит кусок холста в шедевр искусства?
>
> *Из фильма «Я, робот» (2004)*

Упражнение 1 (от регрессии к нейросетке)

Однажды вечером, по пути с работы[5] Маша зашла в свою любимую кофейню на Тверской. Там, на стене, она обнаружила очень интересную картину:



Хозяин кофейни, Добродум, объяснил Маше, что это Покрас-Лампас так нарисовал линейную регрессию, и её легко можно переписать в виде формулы: $y_i = w_0 + w_1 \cdot x_i$. Пока Добродум готовил кофе, Маша накидала у себя на бумажке новую картинку:



Как такая функция будет выглядеть в виде формулы? Правда ли, что y будет нелинейно зависеть от x? Если нет, как это исправить и сделать зависимость нелинейной?

[5] она работает рисёрчером.

Решение:

Когда мы переписывали картинку в виде уравнения регрессии, мы брали вход из кругляшей, умножали его на веса, написанные около стрелок и искали сумму. Сделаем ровно то же самое для Машиной картинки. Величины h_i внутри кругляшей скрытого слоя будут считаться как:

$$h_1 = w_{11} \cdot 1 + w_{21} \cdot x$$
$$h_2 = w_{12} \cdot 1 + w_{22} \cdot x$$

Итоговый y будет получаться из этих промежуточных величин как

$$y = w_1 \cdot h_1 + w_2 \cdot h_2.$$

Подставим вместо h_i их выражение через x и получим уравнение, которое описывает картинку Маши

$$\begin{aligned} y = w_1 \cdot h_1 + w_2 \cdot h_2 = \\ = w_1 \cdot (w_{11} + w_{21} \cdot x) + w_2 \cdot (w_{12} + w_{22} \cdot x) = \\ = \underbrace{(w_1 w_{11} + w_2 w_{12})}_{\gamma_1} + \underbrace{(w_1 w_{21} + w_2 w_{22})}_{\gamma_2} \cdot x. \end{aligned}$$

Когда мы раскрыли скобки, мы получили ровно ту же самую линейную регрессию. Правда мы зачем-то довольно сложно параметризовали γ_1 и γ_2 через шесть параметров. Чтобы сделать зависимость нелинейной, нужно немного преобразить каждую из h_i, взяв от них какую-нибудь нелинейную функцию. Например, сигмоиду:

$$f(h) = \frac{1}{1 + e^{-h}}.$$

Тогда формула преобразиться:

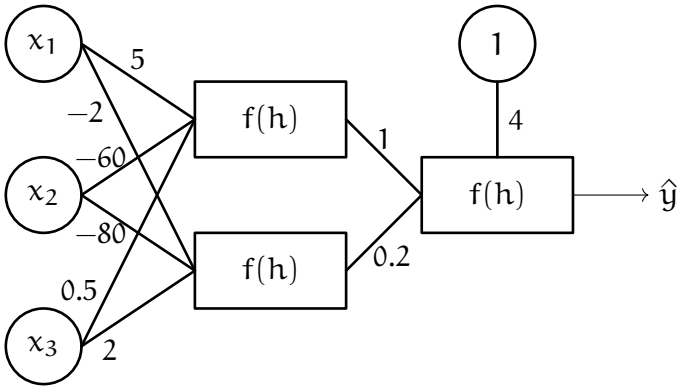
$$y = w_1 \cdot f(w_{11} + w_{21} \cdot x) + w_2 \cdot f(w_{12} + w_{22} \cdot x).$$

~~Смерти~~ Линейности больше нет. **Только что на ваших глазах произошло чудо. Регрессия превратилась в нейросеть.** Можно использовать вместо сигмоиды любую другую функцию активации. Например, ReLU (Rectified Linear Unit)

$$\mathrm{ReLU}(h) = \max(0, h).$$

Упражнение 2 (из картинки в формулу)

Добродум хочет понять, насколько сильно будет заполнена кофейня в следующие выходные. Для этого он обучил нейросетку. На вход она принимает три фактора: температуру за окном, x_1, факт наличия на Тверской митинга, x_2 и пол бариста на смене, x_3. В качестве функции активации Добродум использует ReLU.



а. В эти выходные за барной[6] стойкой стоит Агнесса. Митинга не предвидится, температура будет в районе 20 градусов. Спрогнозируйте, сколько человек придёт в кофейню к Добродуму?

б. На самом деле каждая нейросетка — это просто-напросто какая-то нелинейная сложная функция. Запишите нейросеть Добродума в виде функции.

Решение:

Будем постепенно идти по сетке и делать вычисления. Подаём все значения в первый нейрон, получаем:

$$h_1 = \max(0, 5 \cdot 20 + (-60) \cdot 0 + 0.5 \cdot 1) = \max(0, 100.5) = 100.5$$

Ровно то же самое делаем со вторым нейроном:

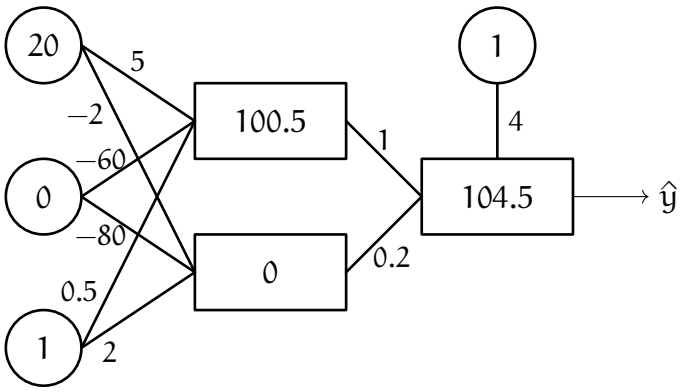
$$h_2 = \max(0, -2 \cdot 20 + (-80) \cdot 0 + 2 \cdot 1) = \max(0, -38) = 0$$

Дальше результат скрытых нейронов идёт во второй слой:

$$\hat{y} = \max(0, 1 \cdot 100.5 + 0.2 \cdot 0 + 4 \cdot 1) = 104.5$$

Это и есть итоговый прогноз.

[6] барной... конечно, кофейня у него...



Теперь по мотивам наших вычислений запишем нейронку как функцию. Начинать будем с конца:

$$\hat{y} = f(1 \cdot h_1 + 0.2 \cdot h_2 + 4 \cdot 1)$$

Подставляем вместо h_1 и h_2 вычисления, которые происходят на первом слое нейронки:

$$\begin{aligned}\hat{y} = f(1 \cdot f(5 \cdot x_1 - 60 \cdot x_2 + 0.5 \cdot x_3) + 0.2 \cdot f(-2 \cdot x_1 - 80 \cdot x_2 + 2 \cdot x_3) + 4 \cdot 1) = \\ = \max(0, \max(0, 5 \cdot x_1 - 60 \cdot x_2 + 0.5 \cdot x_3) + 0.2 \cdot \max(0, -2 \cdot x_1 - 80 \cdot x_2 + 2 \cdot x_3) + 4).\end{aligned}$$

Обучение нейронной сетки, на самом деле, эквивалентно обучению такой сложной нелинейной функции.

Упражнение 3 (из формулы в картинку)

Маша написала на бумажке функцию:

$$y = \max(0, 4 \cdot \max(0, 3 \cdot x_1 + 4 \cdot x_2 + 1) + 2 \cdot \max(0, 3 \cdot x_1 + 2 \cdot x_2 + 7) + 6)$$

Теперь она хочет, чтобы кто-нибудь из её адептов нарисовал её в виде нейросетки. Нарисуйте.

Решение:

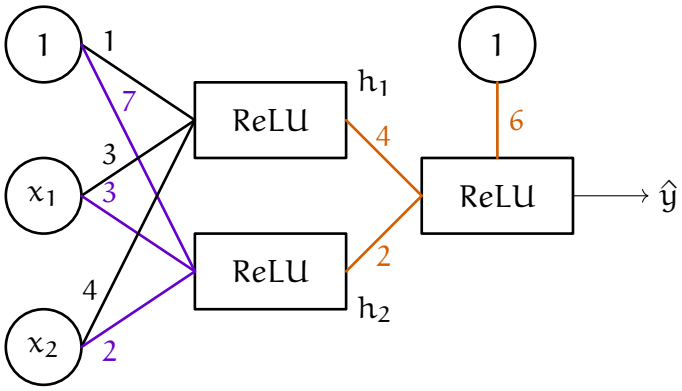
Начнём рисовать картинку с конца. На выход выплёвывается либо 0, либо комбинация из двух входов:

$$\hat{y} = \mathrm{ReLU}(4 \cdot h_1 + 2 \cdot h_2 + 6)$$

Каждый из входов — это снова либо 0, либо комбинация из двух входов.

$$y = \max(0, 4 \cdot \underbrace{\max(0, 3 \cdot x_1 + 4 \cdot x_2 + 1)}_{h_1} + 2 \cdot \underbrace{\max(0, 3 \cdot x_1 + 2 \cdot x_2 + 7)}_{h_2} + 6)$$

Получается, что на первом слое находится два нейрона, которые передают свои выходы в третий:

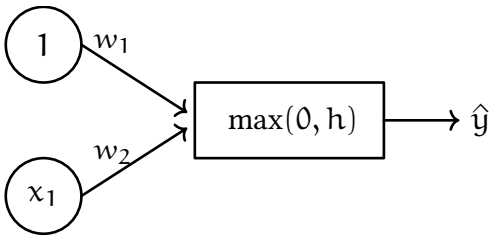


Упражнение 4 (армия регрессий)

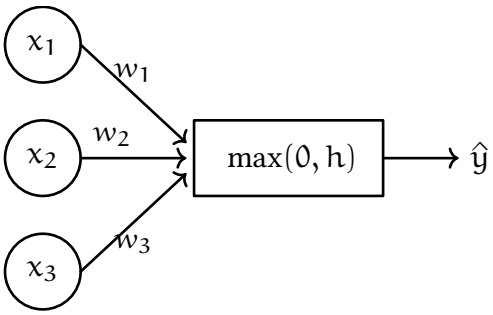
Парни очень любят Машу,[7] а Маша с недавних пор любит собирать персептроны и думать по вечерам об их весах и функциях активации. Сегодня она решила разобрать свои залежи из персептронов и как следует упорядочить их.

а. В ящике стола Маша нашла персептрон с картинки 1 Маша хочет подобрать веса так, чтобы он реализовывал логическое отрицание, то есть превращал $x_1 = 0$ в $y = 1$, а $x_1 = 1$ в $y = 0$.

Картинка 1



Картинка 2



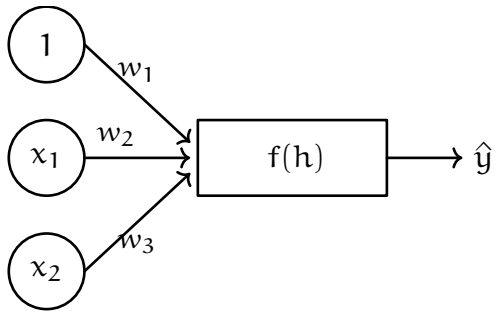
б. В тумбочке, среди носков, Маша нашла персептрон, с картинки 2, Маша хочет подобрать такие веса w_i, чтобы персептрон превращал x из таблички в соответствующие y:

x_1	x_2	x_3	y
1	1	2	0.5
1	−1	1	0

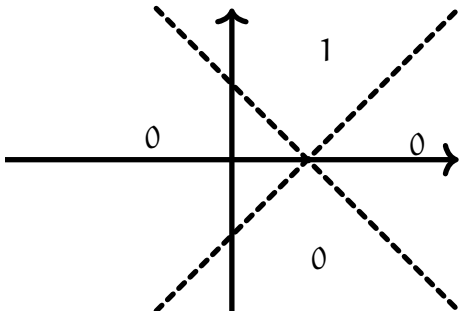
[7] когда у тебя есть лёрнинг, они так и лезут

в. Оказывается, что в ванной всё это время валялась куча персептронов с картинки 3 с неизвестной функцией активации.

Картинка 3



Картинка 4



Маша провела на плоскости две прямые: $x_1 + x_2 = 1$ и $x_1 - x_2 = 1$. Она хочет собрать из персептронов нейросетку, которая будет классифицировать объекты с плоскости так, как показано на картинке 4. В качестве функции возьмите единичную ступеньку (Функцию Хевисайда).

Решение:

а. Начнём с первого пункта. Чтобы было легче запишем нейрон в виде уравнения:

$$\hat{y} = \max(0, w_1 + w_2 \cdot x_1).$$

Нам нужно, чтобы

$$\max(0, w_1 + w_2 \cdot 1) = 0$$
$$\max(0, w_1 + w_2 \cdot 0) = 1$$

На второе уравнение w_2 никак не влияет, а $w_1 = 1$. Для того, чтобы в первом уравнении получить ноль, нужно взять любое $w_2 \leqslant -1$. Нейрон готов.

б. Снова выписываем несколько уравнений:

$$\max(0, w_1 + w_2 + 2 \cdot w_3) = 0.5$$
$$\max(0, w_1 - w_2 + w_3) = 0$$

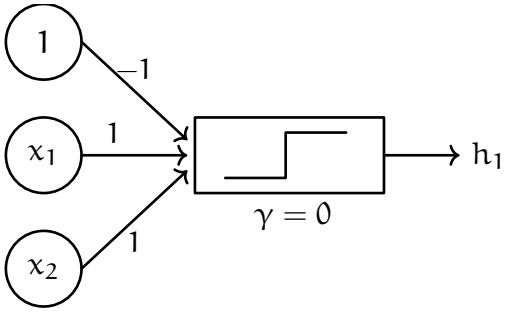
Тут решений может быть довольно много. Одно из них — это занулить w_1 и w_3 в первом уравнении, а w_2 поставить 0.5. Тогда во втором уравнении мы сразу же будем оказываться в отрицательной области и ReLU заботливо будет отдавать нам 0.

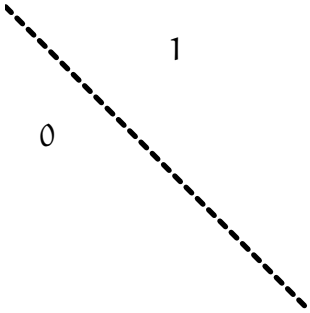
в. Единичная ступенька выглядит как

$$f(h) = \begin{cases} 1, h > 0 \\ 0, h \leqslant 0 \end{cases}.$$

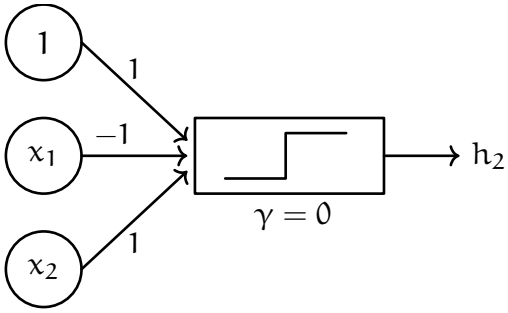
Один нейрон — это одна линия, проведённая на плоскости. Эта линия отделяет один

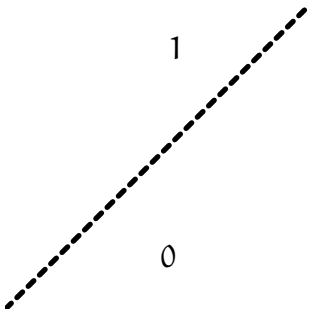
класс от другого. Например, линию $x_1 + x_2 - 1 = 0$ мог бы описать нейрон





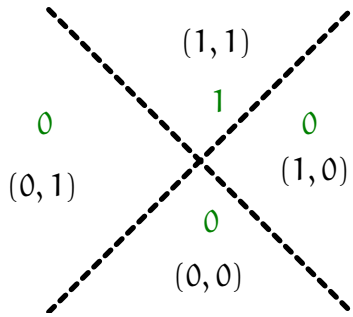
Порог γ для кусочной функции в каком-то смысле дублирует константу. Они взаимосвязаны. Будем всегда брать его нулевым. Видим, что если мы получили комбинацию x_1, x_2 и 1, большую, чем ноль, мы оказались справа от прямой. Если хочется поменять метки 0 и 1 местами, можно умножить все коэффициенты на -1. **Наш персептрон понимает по какую сторону от прямой мы оказались,** то есть задаёт одну линейную разделяющую поверхность. По аналогии для второй прямой мы можем получить следующий результат.

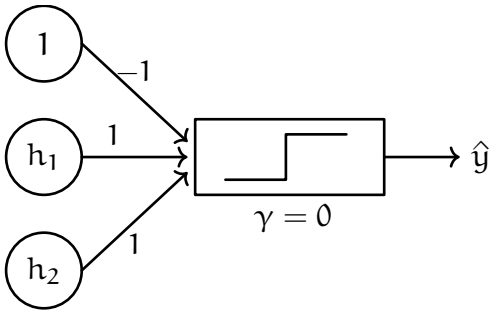




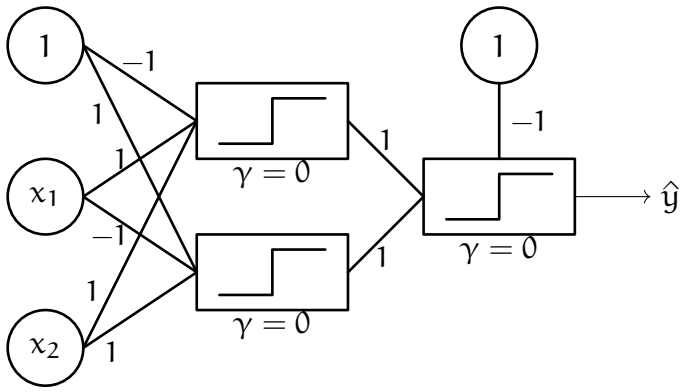
Итак, первый персептрон выбрал нам позицию относительно первой прямой, второй относительно второй. Остаётся только объединить эти результаты. Нейрон для скрепки должен реализовать для нас логическую функцию, которую задаёт табличка ниже. Там же нарисованы примеры весов, которые могли бы объединить выхлоп первого слоя в итоговый прогноз.

h_1	h_2	$\hat{y}$
1	1	1
1	0	0
0	1	0
0	0	0





Теперь мы можем нарисовать итоговую нейронную сеть, решающую задачу Маши. Она состоит из двух слоёв. Меньше не выйдет, так как каждый персептрон строит только одну разделяющую линию.



Кстати говоря, если бы мы ввели для нашей нейросетки дополнительный признак $x_1 \cdot x_2$, у нас бы получилось обойтись только одним персептроном. В нашей ситуации **нейросетка сама сварила на первом слое признак $x_1 \cdot x_2$, которого ей не хватало.** Другими словами говоря, нейросетка своим первым слоем превратила сложное пространство признаков в более простое, а затем вторым слоем, решила в нём задачу классификации.

Упражнение 5 (логические функции)

Маша вчера поссорилась с Пашей. Он сказал, что у неё нет логики. Чтобы доказать Паше обратное, Маша нашла теорему, которая говорит о том, что с помощью нейросетки можно аппроксимировать почти любую функцию, и теперь собирается заняться аппроксимацией логических функций. Для начала она взяла самые простые, заданные следующими таблицами истинности:

x_1	x_2	$x_1 \cap x_2$
1	1	1
1	0	0
0	1	0
0	0	0

x_1	x_2	$x_1 \cup x_2$
1	1	1
1	0	1
0	1	1
0	0	0

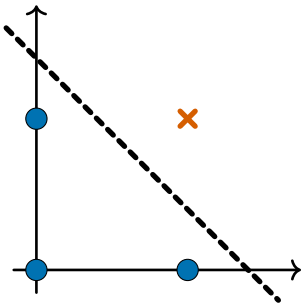
x_1	x_2	x_1 XoR x_2
1	1	0
1	0	1
0	1	1
0	0	0

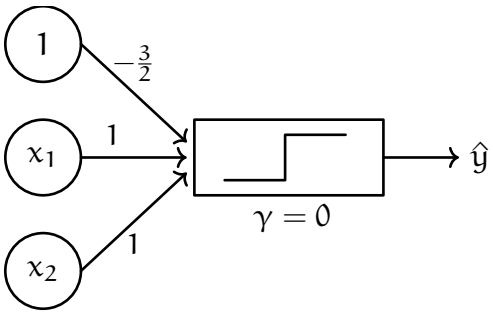
Первые два столбика идут на вход, третий получается на выходе. Первая операция — логическое "и вторая — "или". Операция из третьей таблицы называется "исключающим или (XoR). Если внимательно приглядеться, то можно заметить, что XoR — это то же самое что и $[x_1 \neq x_2]$[8].

Решение:

На самом деле, в предыдущем упражнении, мы уже построили нейрон для пересечения. Он располагался на последнем слое нейросети. Посмотрим на тот же нейрон под другим углом:

[8]Тут квадратные скобки обозначают индикатор. Он выдаёт 1, если внутри него стоит правда и 0, если ложь. Такая запись называется скобкой Айверсона. Попробуйте записать через неё единичную ступеньку Хевисайда.

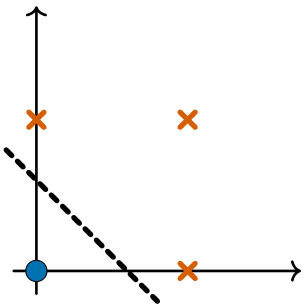


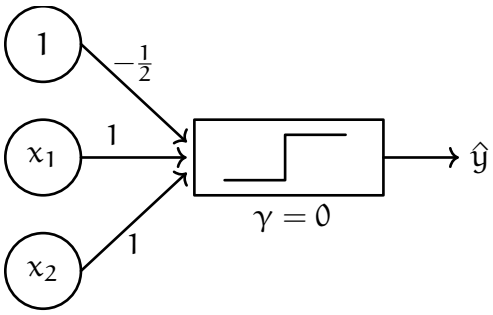


Если нарисовать все наши четыре точки на плоскости, становится ясно, что мы хотим отделить точку $(1, 1)$ от всех остальных. Сделать это можно большим числом способов. Например, в нейроне выше задана линия $x_2 = 1.5 - x_1$. Подойдёт и любая другая линия, отделяющая $(1, 1)$ от остальных точек. Пропустим ради приличия точки через наш нейрон и убедимся, что он работает корректно:

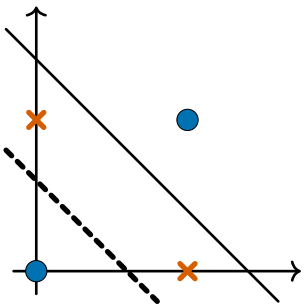
$$[-1.5 + 1 + 1 > 0] = [0.5 > 0] = 1$$
$$[-1.5 + 0 + 0 > 0] = [-1.5 > 0] = 0$$
$$[-1.5 + 0 + 1 > 0] = [-0.5 > 0] = 0$$
$$[-1.5 + 1 + 0 > 0] = [-0.5 > 0] = 0$$

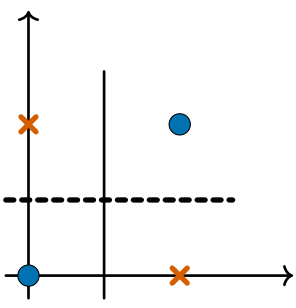
С объединением та же ситуация, только на этот раз линия должна пройти чуть ниже. Подойдёт $x_2 = 0.5 - x_1$.





С третьей операцией, исключающим или, начинаются проблемы. Чтобы разделить точки, нужно строить две линии. Сделать это можно многими способами. Но линий всегда будет две. То есть мы попадаем в ситуацию из прошлой задачки. Надо посмотреть первым слоем нейросетки, где мы оказались относительно каждой из линий, а вторым слоем соединить результаты.





Если немного пофантазировать, можно даже записать эту нейросеть через объединение и пересечение:

$$\hat{y} = [1 \cdot (x_1 \cup x_2) - 1 \cdot (x_1 \cap x_2) - 0.5 > 0]$$

Нейрон $(x_1 \cup x_2)$ выясняет по какую сторону от сплошной линии мы оказались, нейрон $x_1 \cap x_2$ делает то же самое для пунктирной линии. А дальше мы просто объединяем результат.

Упражнение 6 (ещё немного про XoR)

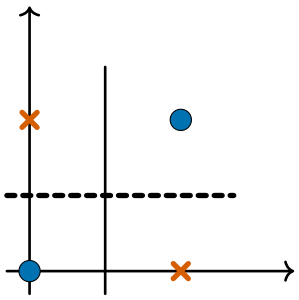
Маша заметила, что на XoR ушло очень много персептронов. Она поняла, что первые два персептрона пытаются сварить для третьего нелинейные признаки, которых нейросетке не хватает. Она решила самостоятельно добавить персептрону вход $x_3 = x_1 \cdot x_2$ и реализовать XoR одним персептроном. Можно ли это сделать?

Решение:

Маша обратила внимание на очень важную штуку. Нам не хватает признаков, чтобы реализовать XoR за один нейрон. Поэтому первый слой нейросетки сам их для нас придумывает. Чем глубже нейросетку мы построим, тем более сложные и абстрактные признаки она будет выделять из данных. Если добавить ко входу $x_3 = x_1 \cdot x_2$, мы сделаем за нейросетку часть её работы и сможем обойтись одним нейроном. Например, вот таким:

$$\hat{y} = [x_1 + x_2 - 2 \cdot x_1 \cdot x_2 - 0.5 > 0]$$

Такая линия как раз будет задавать две скрещивающиеся прямые.



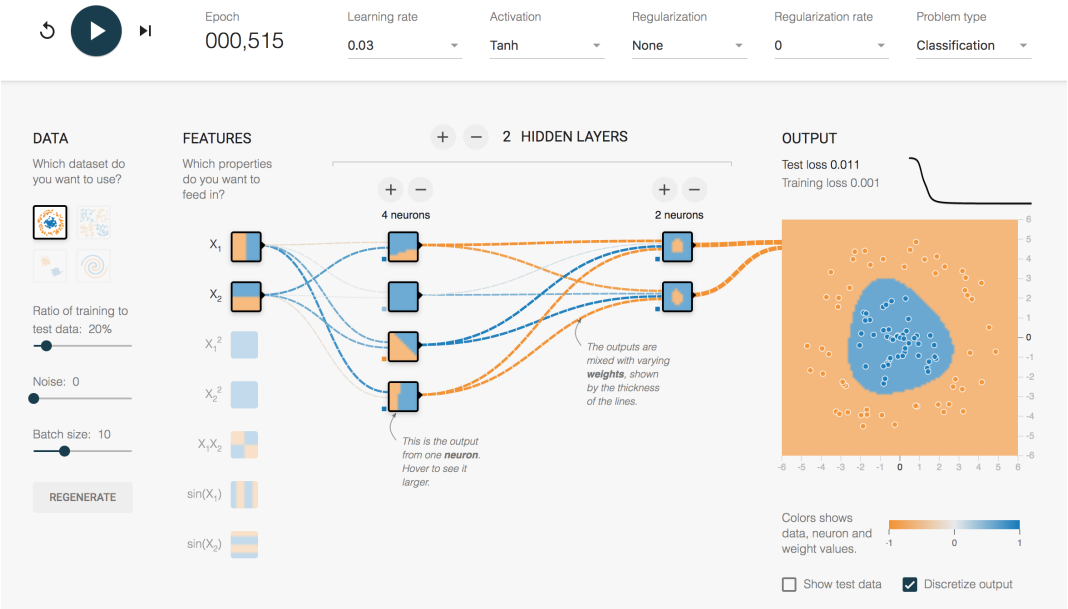
Это легко увидеть, если немного поколдовать над уравнением:

$$\begin{aligned}
&x_1 + x_2 - 2x_1x_2 - 0.5 = 0 \\
&2x_1 + 2x_2 - 4x_1x_2 - 1 = 0 \\
&2x_1(1 - 2x_2) + 2x_2 - 1 = 0 \\
&(1 - 2x_2) \cdot (2x_1 - 1) = 0
\end{aligned}$$

Получаем два решения. Прямую $x_2 = 0.5$ и прямую $x_1 = 0.5$.

Упражнение 7 (избыток)

На сайте `http://playground.tensorflow.org` Маша стала играться с простенькими нейросетками и обучила для решения задачи классификации трёхслойного монстра.



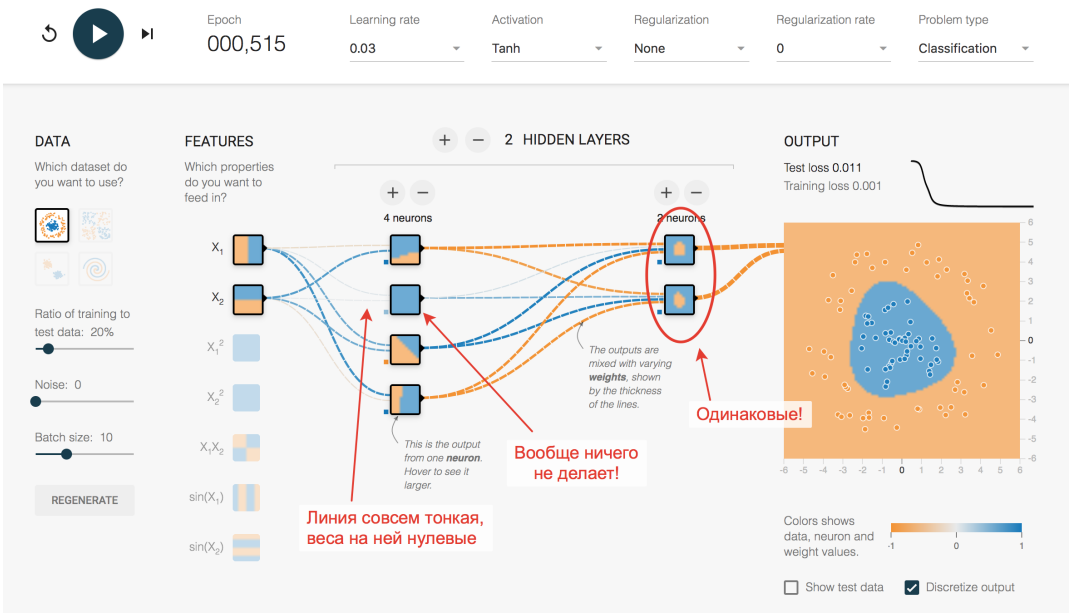
Голубым цветом обозначен первый класс, рыжим второй. Внутри каждого нейрона визуализирована та разделяющая поверхность, которую он выстраивает. Так, первый слой ищет разделяющую линию. Второй слой пытается из этих линий выстроить более сложные фигуры и так далее. Чем ярче связь между нейронами, тем больше весовой коэффициент, относящейся к ней. Синие связи — положительные, рыжие — отрицательные. Чем тусклее связь, тем он ближе к нулю.

Маша заметила, что с её архитектурой что-то не так. Какие у неё проблемы?

Решение:

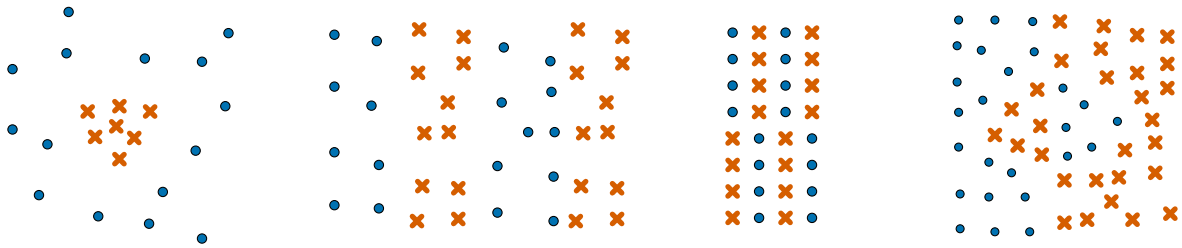
Нейросетка Маши оказалась избыточной. Во-первых, можно увидеть, что на первом слое есть нейрон, который вообще ничего не делает. Связи, которые идут к нему от входов настолько тусклые (коэффициенты при них равны нулю), что их даже не видно на картинке. От этого нейрона смело можно избавиться и сделать архитектуру проще. Во-вторых, можно заметить, что на последнем слое у нас есть два одинаковых нейрона. Один из них смело можно выбрасывать.

Для решения такой простой задачи классификации подойдёт более простая модель. Сколько минимально нужно нейронов, чтобы её решить вам и Маше предстоит выяснить в следующей задаче.



Упражнение 8 (минималочка)

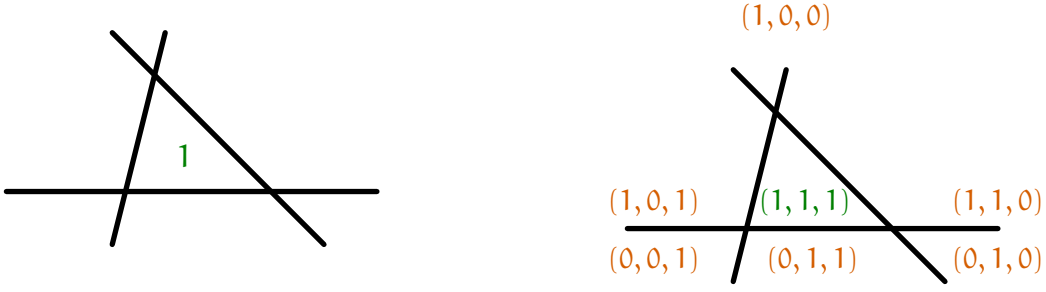
Шестилетняя сестрёнка ворвалась в квартиру Маши и разрисовала ей все обои:



Маша по жизни оптимистка. Поэтому она увидела не дополнительные траты на ремонт, а четыре задачи классификации. И теперь в её голове вопрос: сколько минимально нейронов нужно, чтобы эти задачи решить?

Решение:

а. Нам с помощью нейросетки надо выделить треугольник. Всё, что внутри будет относится к первому классу. Получается, что на первом слое надо три нейрона. Каждый из них настроим так, что если мы попадаем внутрь треугольника, он выдаёт 1. Тогда на втором слое будет достаточно одного нейрона, который удостоверится, что все три результата с первого слоя оказались равны 1. Вернитесь к предыдущей задаче, сходите на сайт с демкой и постройте оптимальную нейросетку.



б. Первый слой должен построить нам три линии. Это три нейрона. Второй слой должен

принять решение в какой из полос мы оказались. Будем считать, что если мы попали направо, нейрон выдаёт единицу. Если мы попали налево, ноль. В качестве функции активации используем единичную ступеньку.

Вопрос в том, хватит ли нам на втором слое одного нейрона для того, чтобы обработать все четыре возможные ситуации. Нам нужно, чтобы выполнялись следующие условия

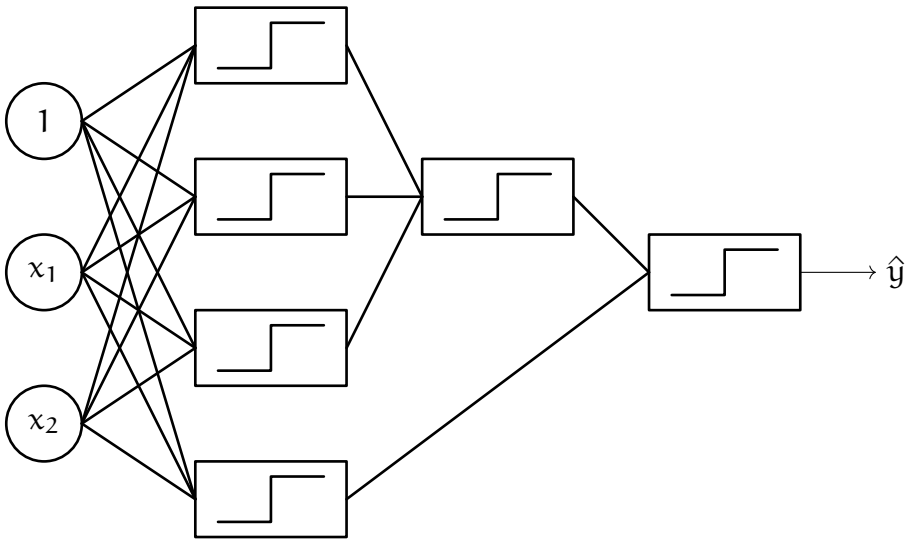
$$\begin{cases} f(1 \cdot w_1 + 1 \cdot w_2 + 1 \cdot w_3) = 1 \\ f(1 \cdot w_1 + 0 \cdot w_2 + 0 \cdot w_3) = 0 \\ f(1 \cdot w_1 + 1 \cdot w_2 + 01 \cdot w_3) = 0 \\ f(0 \cdot w_1 + 0 \cdot w_2 + 0 \cdot w_3) = 0 \end{cases}.$$

Для того, чтобы со вторым уравнением всё было хорошо, возьмём $w_1 = 1$. Тогда вес w_2 надо взять отрицательным, а w_3 положительным, например, $w_2 = -2$, а $w_3 = 4$. Тогда один нейрон на внешнем слое решит нашу задачу. Выходит, что всего надо задействовать 4 нейрона.

в. Оценим число нейронов сверху. Перед нами две XoR задачи, которые лежат рядом с друг-другом. Для решения каждой надо 3 нейрона. Чтобы объединить получившиеся решения нужен ещё один нейрон. Получается трёхслойная сетка с 7 нейронами.

Если мы попробуем подойти к задаче также, как в предыдущем пункте, на втором слое мы получим несовместимую систему из уравнений. То есть третьего слоя точно не избежать.

Можно первым слоем построить 3 линии, вторым решить задачу из предыдущего пункта, а на третьем добавить информацию о том, выше горизонтальной линии мы оказались или ниже. Тогда мы потратим 6 нейронов. Нейросетка получится неполносвязной.



г. Думайте, рассуждайте, а автор умывает руки.

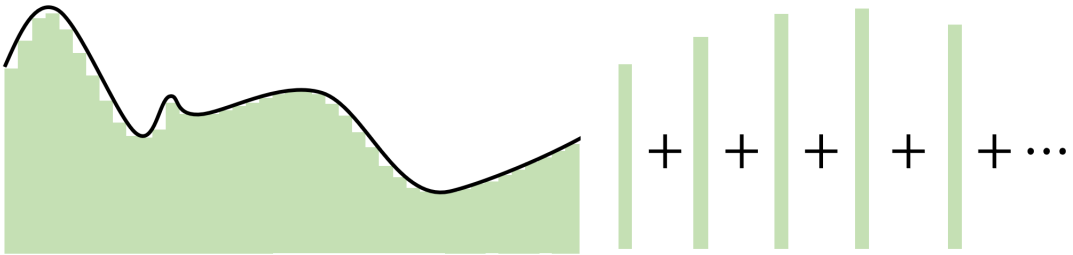
Упражнение 9 (универсальный регрессор)

Маша доказала Паше, что у неё всё в полном порядке с логикой. Теперь она собирается доказать ему, что с помощью двухслойной нейронной сетки можно приблизить любую непрерывную функцию от одного аргумента $f(x)$ со сколь угодно большой точностью[9].

Hint: Вспомните, что любую непрерывную функцию можно приблизить с помощью кусочно-линейной функции (ступеньки). Осознайте как с помощью пары нейронов можно описать такую ступеньку. Соедините все ступеньки в сумму с помощью выходного нейрона.

Решение:

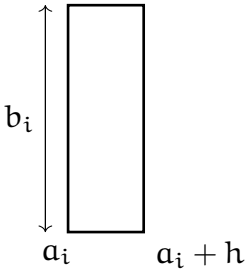
Не нужно воспринимать эту задачку как строгое доказательство. Скорее, это показательство. Мы хотим приблизить функцию $f(x)$ с какой-то точностью. Будем делать это с помощью кусочно-линейных ступенек. Чем выше точность, тем больше будем рисовать ступенек.



Высоту ступеньки определяют по-разному. Чаще всего как значение функции в середине выбранного отрезка $b_i = f(\frac{a_i + a_{i+1}}{2})$. Тогда всю функцию целиком можно приблизить суммой

$$f(x) \approx \sum_{i=1}^{n} f\left(\frac{a_i + a_{i+1}}{2}\right) \cdot [a_i \geqslant x < a_{i+1}].$$

Давайте попробуем описать с помощью нейрона одну из ступенек. Пуст высота этой ступеньки равна b_i. Шагать по оси x мы будем с фиксированным шагом h, поэтому $a_{i+1} = a_i + h$.



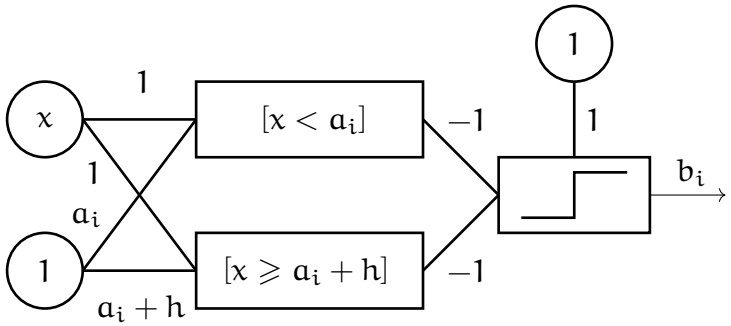
Если x, для которого мы ищем $f(x)$ попадает в полуинтервал, на котором задана наша ступенька, мы будем приближать $f(x)$ этой ступенькой. Ступенька состоит из двух линий.

[9] http://neuralnetworksanddeeplearning.com/chap4.html

Выходит, что она будет описываться двумя нейронами. Если мы внутри ступеньки, значит $a_i \leqslant x < a_i + h$. Пара нейронов должна сравнить x с a_i и $a_i + h$ и на основе этого принять решение. Можно записать попадание x в ступеньку следующим образом:

$$1 - [x < a_i] - [x \geqslant a_i + h]$$

Если оба условия — неправда, получаем 1. Мы в ступеньке. Если хотя бы одно из них выполнено — мы вылетаем за ступеньку. Оба сразу выполниться они не могут. Нарисуем это в виде нейрона. В качестве функции активации используем единичную ступеньку.



Нарисуем такую сетку для каждой ступеньки. Если мы попали в ступеньку, сетка будет выплёвывать со второго слоя единичку. Там мы будем умножать её на b_i и посылать на внешний слой. Мы всегда будем попадать только в одну из ступенек, значит только один из слоёв выдаст нам 1. Все остальные выдадут 0. На внешнем слое нам остаётся только просуммировать всё, что к нам пришло и выдать ответ. Чем больше ступенек мы добавляем в модель, тем точнее наша апроксимация.

Упражнение 10 (число параметров)

Та, кому принадлежит машин лёрнинг собирается обучить полносвязную нейронную сеть для решения задачи регрессии, На вход в ней идёт 12 переменных, в сетке есть 3 скрытых слоя. В первом слое 300 нейронов, во втором 200, в третьем 100. Сколько параметров предстоит оценить Маше?

Решение:

Нам нужно решить довольно простую комбинаторную задачку. Связи в нашей сетке проведены между всему нейронами. Не забываем учесть, что у каждого нейрона есть константа. Получается, что всего параметров будет

$$(12 + 1) \cdot 300 + (300 + 1) \cdot 200 + (200 + 1) \cdot 100 + (100 + 1) \cdot 1.$$

Листочек 2: что выплёвывает нейросеть

Плюют в душу обычно те, кому не удалось в неё влезть.

Пацанский паблик категории Б

Упражнение 11 (про сигмоиду)

Любую s-образную функцию называют сигмоидой. Наиболее сильно прославилась под таким названием функция $f(t) = \frac{e^t}{1+e^t}$. Слава о ней добралась до Маши и теперь она хочет немного поисследовать её свойства[10].

а. Что происходит при $t \to +\infty$? А при $t \to -\infty$?

б. Как связаны между собой $f(t)$ и $f(-t)$?

в. Как связаны между собой $f'(t)$ и $f'(-t)$?

г. Как связаны между собой $f(t)$ и $f'(t)$?

д. Найдите $f(0)$, $f'(0)$ и $\ln f(0)$.

е. Найдите обратную функцию $f^{-1}(t)$

ж. Как связаны между собой $\frac{d \ln f(t)}{dt}$ и $f(-t)$?

з. Постройте графики функций $f(t)$ и $f'(t)$.

и. Говорят, что сигмоида — это гладкий аналог единичной ступеньки. Попробуйте построить на компьютере графики $f(t)$, $f(10 \cdot t)$, $f(100 \cdot t)$, $f(1000 \cdot t)$. Как они себя ведут?

Упражнение 12 (про logloss)

У Маши три наблюдения, первое наблюдение — кит, остальные — муравьи. Киты кодируются $y_i = 1$, муравьи — $y_i = 0$. В качестве регрессоров Маша берёт номера наблюдений $x_i = i$. После этого Маша оценивает логистическую регрессию с константой. В качестве функции потерь используются логистические потери.

а. Выпишите для данной задачи функцию потерь, которую минимизирует Маша.

б. При каких оценках коэффициентов логистической регрессии эта функция достигает своего минимума?

Упражнение 13 (про softmax)

Маша чуть внимательнее присмотрелась к своему третьему наблюдению и поняла, что это не кит, а бобёр. Теперь ей нужно решать задачу классификации на три класса. Она решил использовать для этого нейросеть с softmax-слоем на выходе.

Маша уже обучила нейронную сетку и хочет построить прогнозы для двух наблюдений. Слой, который находится перед softmax выдал для этих двух наблюдений следующий резуль-

[10] Часть задач украдена отсюда: https://github.com/bdemeshev/mlearn_pro

тат: $(1, -2, 0)$ и $(0.5, -1, 0)$.

а. Чему равны вероятности получить кита, муравья и бобра для этих двух наблюдений?

б. Пусть первым был кит, а вторым бобёр. Чему будет равна logloss-ошибка?

в. Пусть у Маши есть два класса. Она хочет выучить нейросеть. Она может учить нейронку с одним выходом и сигмоидой в качестве функции активации либо нейронку с двумя выходами и softmax в качестве функции активации. Как выходы этих двух нейронок взаимосвязаны между собой?

г. Объясните, почему softmax считают сглаженным вариантом максимума.

Упражнение 14 (про разные выходы)

Та, в чьих руках находится лёрнинг, решила немного поэкспериментировать с выходами из своей сетки.

а. Для начала Маша решила, что хочет решать задачу классификации на два класса и получать на выходе вероятность принадлежности к первому. Что ей надо сделать с последним слоем сетки?

б. Теперь Маша хочет решать задачу классификации на K классов. Что ей делать с последним слоем?

в. Новые вводные! Маша хочет спрогнозировать рейтинг фильма на "Кинопоиске". Он измеряется по шкале от 0 до 10 и принимает любое непрерывное значение. Как Маша может приспособить для этого свою нейронку?

г. У Маши есть куча новостей. Каждая новость может быть спортивной, политической или экономической. Иногда новость может относится сразу к нескольким категориям. Как Маше собрать нейросетку для решения этой задачи? Как будет выглядеть при этом функция ошибки?

д. У Маши есть картинки с уточками и чайками. Маша хочет научить нейросеть искать на картинке птицу, обводить её в прямоугольник (bounding box), а затем классифицировать то, что попало в прямоугольник. Как должен выглядеть выход из такой нейросети? Как должна выглядеть функция потерь?

е. Маша задумалась, как можно спрогнозировать число людей в кафе так, чтобы на выходе сетка всегда прогнозировала целое число. Надо ли как-то при этом менять функцию потерь?

Hint: вспомните про пуассоновскую регрессию.

Листочек 3: пятьдесят оттенков градиентного спуска

Повторять до сходимости — это как жарить до готовности

Неизвестный студент Вышки

Упражнение 15 (50 оттенков спуска)

Маша Нестерова, хозяйка машин лёрнинга[11], собрала два наблюдения: $x_1 = 1, x_2 = 2$, $y_1 = 2, y_2 = 3$ и собирается обучить линейную регрессию $y = w \cdot x$. Маша очень хрупкая девушка, и ей не помешает помощь.

а. Получите теоретическую оценку методом наименьших квадратов.

б. Сделайте три шага градиентного спуска. В качестве стартовой точки используйте $w_0 = 0$. В качестве скорости обучения возьмите $\eta = 0.1$.

в. Сделайте четыре шага стохастического градиентного спуска. Пусть в SGD сначала попадает первое наблюдение, затем второе.

г. Если вы добрались до этого пункта, вы поняли градиентный спуск. Маша довольна. Начинаем заниматься тупой технической бессмыслицей. Сделайте два шага Momentum SGD. Возьмите $\alpha = 0.9, \eta = 0.1$

д. Сделайте два шага Momentum SGD с коррекцией Нестерова.

е. Сделайте два шага RMSprop. Возьмите $\alpha = 0.9, \eta = 0.1$

ж. Сделайте два шага Adam. Возьмём $\beta_1 = \beta_2 = 0.9, \eta = 0.1$

Решение:

а. Найдём теоретическую оценку стандартным МНК. Минимизируем MSE:

$$\text{MSE} = \frac{1}{2} \cdot ((2 - w)^2 + (3 - 2w)^2) \to \min_w$$

Берём производную, решаем уравнение и получаем ответ:

$$-(2 - w) - 2(3 - 2w) = 0 \Rightarrow \hat{w} = \frac{8}{5} = 1.6$$

б. Чтобы сделать три шага градиентного спуска, нужно найти градиент. Наша функция потерь выглядит как

$$L(w) = (y - wx)^2.$$

[11] Лёрнинг ей папа подарил

Мы подбираем один параметр, значит градиентом в данном случае будет просто одно число — производная по этому параметру.

$$\nabla L(w_0, x, y) = \frac{\partial L}{\partial w} = -2x(y - wx).$$

Стартовая точка $w_0 = 0$. Мы хотим сделать шаг

$$w_1 = w_0 - \eta \cdot \nabla L(w_0)$$

Посчитаем градиент в точке w_0 по всей выборке:

$$\nabla L(w_0) = \frac{1}{n} \cdot \sum_{i=1}^{n} \nabla L(w_0, x_i, y_i) = \frac{1}{2} \cdot (-2(2 - w_0) - 2 \cdot 2(3 - 2w_0)) = -8$$

Делаем **первый шаг**:

$$w_1 = 0 + 0.1 \cdot 8 = 0.8$$

По аналогии, **второй шаг**:

$$\nabla L(w_1) = \frac{1}{2} \cdot (-2(2 - w_1) - 2 \cdot 2(3 - 2w_1)) = -4$$
$$w_2 = w_1 - \eta \cdot \nabla L(w_1) = 0.8 + 0.1 \cdot 4 = 1.2$$

По аналогии, **третий шаг**:

$$\nabla L(w_2) = \frac{1}{2} \cdot (-2(2 - w_2) - 2 \cdot 2(3 - 2w_2)) = -2$$
$$w_3 = w_2 - \eta \cdot \nabla L(w_2) = 1.2 + 0.1 \cdot 2 = 1.4$$

в. Теперь то же самое, на градиентный спуск стохастический. Мы будем считать $\nabla(w_t)$ не как среднее по всей выборке, а как значение градиента в одной случайно выбранной точке.

Первый шаг:

$$\nabla L(w_0) = -2(2 - w_0 \cdot 1) = -4$$
$$\beta_1 = w_0 - \eta \cdot \nabla L(w_0) = 0 + 0.1 \cdot 4 = 0.4$$

Второй шаг:

$$\nabla L(w_1) = -2 \cdot 2 \cdot (3 - w_1 \cdot 2) = -8.8$$
$$w_2 = w_1 - \eta \cdot \nabla L(w_1) = 0.4 + 0.1 \cdot 8.8 = 1.28$$

Третий шаг:

$$\nabla L(w_2) = -2(2 - w_1 \cdot 1) = -1.44$$
$$w_3 = w_2 - \eta \cdot \nabla L(w_2) = 1.28 + 0.144 = 1.424$$

Четвёртый шаг:

$$\nabla L(w_3) = -2 \cdot 2(3 - w_3 \cdot 2)$$
$$w_4 = w_3 - \eta \cdot \nabla L(w_4) = 1.424 + 0.1 \cdot 0.608 = 1.4848$$

г. Автор не очень хочет расписывать решение дальнейших четырёх пунктов. Но вы обязательно это сделайте.

Упражнение 16 (логистическая регрессия)

Маша решила, что нет смысла останавливаться на обычной регрессии, когда она знает, что есть ещё и логистическая:

$$z = w \cdot x \qquad p = P(y = 1) = \frac{1}{1 + e^{-z}}$$
$$\text{logloss} = -[y \cdot \ln p + (1 - y) \cdot \ln(1 - p)]$$

Запишите формулу, по которой можно пересчитывать веса в ходе градиентного спуска для логистической регрессии.

Оказалось, что $x = -5$, а $y = 1$. Сделайте один шаг градиентного спуска, если $w_0 = 1$, а скорость обучения $\gamma = 0.01$.

Решение:

Сначала нам надо найти $\text{logloss}'_{\hat{p}}$. В принципе в этом и заключается вся сложность задачки. Давайте подставим вместо $\hat{p}$ в logloss сигмоиду.

$$\text{logloss} = -1 \left(y \cdot \ln \left(\frac{1}{1 + e^{-z}} \right) + (1 - y) \cdot \ln \left(1 - \frac{1}{1 + e^{-z}} \right) \right)$$

Теперь подставим вместо z уравнение регрессии:

$$\text{logloss} = -1 \left(y \cdot \ln \left(\frac{1}{1 + e^{-w \cdot x}} \right) + (1 - y) \cdot \ln \left(1 - \frac{1}{1 + e^{-w \cdot x}} \right) \right)$$

Это и есть наша функция потерь. От неё нам нужно найти производную. Давайте подготовимся.

Делай раз, найдём производную logloss по $\hat{p}$:

$$\text{logloss}'_{\hat{p}} = -1 \left(y \cdot \frac{1}{\hat{p}} - (1 - y) \cdot \frac{1}{(1 - p)} \right)$$

Делай два, найдём производную $\frac{1}{1+e^{-wx}}$ по w:

$$\left(\frac{1}{1+e^{-wx}}\right)'_w = -\frac{1}{(1+e^{-wx})^2}\cdot e^{-wx}\cdot(-x) = \frac{1}{1+e^{-wx}}\cdot\frac{e^{-wx}}{1+e^{-wx}}\cdot x = \\ = \frac{1}{1+e^{-wx}}\cdot\left(1-\frac{1}{1+e^{-wx}}\right)\cdot x$$

По-другому это можно записать как $\hat{p}\cdot(1-\hat{p})\cdot x$.

Делай три, находим полную производную:

$$\text{logloss}'_\beta = -1\left(y\cdot\frac{1}{\hat{p}}\cdot\hat{p}\cdot(1-\hat{p})\cdot x - (1-y)\cdot\frac{1}{(1-\hat{p})}\cdot\hat{p}\cdot(1-\hat{p})\cdot x\right) = \\ = -y\cdot(1-\hat{p})\cdot x + (1-y)\cdot\hat{p}\cdot x = (-y+y\hat{p}+\hat{p}-y\hat{p})\cdot x = (\hat{p}-y)\cdot x$$

Найдём значение производной в точке $w_0 = 1$ для нашего наблюдения $x = -5, y = 1$:

$$\left(\frac{1}{1+e^{-1\cdot(-5)}} - 1\right)\cdot(-5) \approx 4.96$$

Делаем шаг градиентного спуска:

$$w_1 = 1 - 0.01\cdot 4.96 \approx 0.95$$

Упражнение 17 (вопросики)

Убедитесь, что вы можете дать ответы на следующие вопросы:

- Как вы думаете, почему считается, что SGD лучше работает для оптимизации функций, имеющих больше одного экстремума?
- Предположим, что у функции потерь есть несколько локальных минимумов. Как можно адаптировать градиентный спуск так, чтобы он находил глобальный минимум чаще?
- Что будет происходить со стохастическим градиентным спуском, если длина его шага не будет уменьшаться от итерации к итерации?

Упражнение 18 (скорости обучения)

В стохастическом градиентном спуске веса изменяются по формуле

$$w_t = w_{t-1} - \eta_t\cdot\nabla L(w_{t-1}, x_i, y_i),$$

где наблюдение i выбрано случайно, скорость обучения зависит от номера итерации.

Условия Роббинса-Монро гарантируют сходимость алгоритма к оптимуму для выпуклых

дифференцируемых функций. Они говорят, что ряд из скоростей $\sum_{t=0}^{\infty} \eta_t$ должен расходиться, а ряд $\sum_{t=0}^{\infty} \eta_t^2$ сходиться. То есть скорость спуска должна падать не слишком медленно, но и не слишком быстро. Какие из последовательностей, перечисленных ниже, можно использовать для описания изменения скорости алгоритма?

а. $\eta_t = \frac{1}{t}$

б. $\eta_t = \frac{0.1}{t^{0.3}}$

в. $\eta_t = \frac{1}{\sqrt{t}}$

г. $\eta_t = \frac{1}{t^2}$

д. $\eta_t = e^{-t}$

е. $\eta_t = \lambda \cdot \left(\frac{s_0}{s_0 + t}\right)^p$, где λ, p и s_0 — параметры

Листочек 4: алгоритм обратного распространения ошибки

К толковому выбору приводит опыт,а к нему приводит выбор бестолковый.

JSON Стэтхэм

Упражнение 19 (граф вычислений)

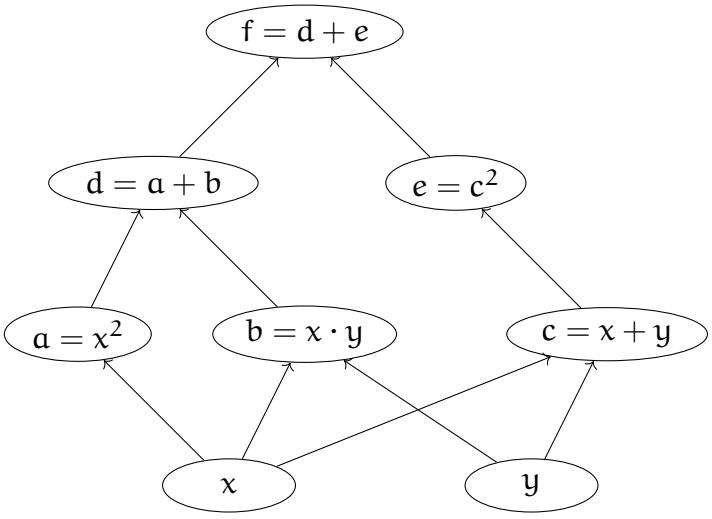
Как найти производную a по b в графе вычислений? Находим не посещённый путь из a в b, перемножаем все производные на рёбрах получившегося пути. Добавляем это произведение в сумму. Так делаем для всех путей. Маша хочет попробовать этот алгоритм на функции

$$f(x, y) = x^2 + xy + (x + y)^2.$$

Помогите ей нарисовать граф вычислений и найти $\frac{\partial f}{\partial x}$ и $\frac{\partial f}{\partial y}$. В каждой вершине графа записывайте результат вычисления одной элементарной операции: сложений или умножения[12].

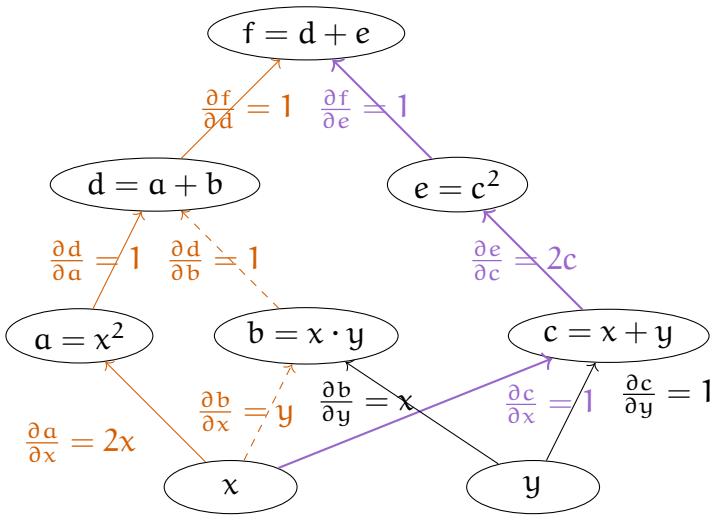
Решение:

Нарисуем граф вычислений.



Каждому ребру припишем производную выхода по входу. Например, ребру между x и a будет соответствовать $\frac{\partial a}{\partial x} = 2x$.

[12]По мотивам книги Николенко "Глубокое обучение"(стр. 79)



Теперь пройдём по всем траекториям из x в f и перемножим производные на рёбрах. После просуммируем получившиеся множители

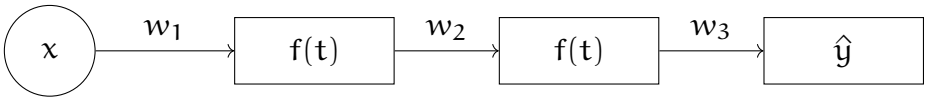
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} \cdot \frac{\partial a}{\partial x} + \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial b} \cdot \frac{\partial b}{\partial x} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial x} = \\ = 1 \cdot 1 \cdot 2x + 1 \cdot 1 \cdot y + 1 \cdot 2c \cdot 1 = 2x + y + 2(x + y).$$

По аналогии найдём производную по траекториям из y в f:

$$\frac{\partial f}{\partial y} = 1 \cdot 1 \cdot x + 1 \cdot 2c \cdot 1 = x + 2(x + y).$$

Упражнение 20 (придумываем backpropagation)

У Маши есть нейросеть с картинки ниже, где w_k — веса для k слоя, $f(t)$ — какая-то функция активации. Маша хочет научиться делать для такой нейронной сетки градиентный спуск.



а. Запишите Машину нейросеть, как сложную функцию.

б. Предположим, что Маша решает задачу регрессии. Она прогоняет через нейросетку одно наблюдение. Она вычисляет знчение функции потерь $L(w_1, w_2, w_3) = \frac{1}{2} \cdot (y - \hat{y})^2$. Найдите производные функции L по всем весам w_k.

в. В производных постоянно повторяются одни и те же части. Постоянно искать их не очень оптимально. Выделите эти части в прямоугольнички цветными ручками.

г. Выпишите все производные в том виде, в котором их было бы удобно использовать для алгоритма обратного распространения ошибки, а затем, сформулируйте сам алгоритм. Нарисуйте под него удобную схемку.

Решение:

Чтобы записать нейросеть как сложную функцию, нужно просто последовательно применить все слои

$$\hat{y}_i = f(f(f(x_i \cdot w_1) \cdot w_2) \cdot w_3.$$

Запишем функцию потерь и аккуратно найдём все производные

$$L(w_1, w_2, w_3) = \frac{1}{2} \cdot (y - \hat{y})^2 = \frac{1}{2} \cdot (y - f(f(x \cdot w_1) \cdot w_2) \cdot w_3)^2.$$

Делаем это по правилу взятия производной сложной функции. Как в школе.

$$\begin{aligned}
\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_3} = (y - \hat{y}) \cdot f(f(x \cdot w_1) \cdot w_2) \\
\frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = (y - \hat{y}) \cdot w_3 \cdot f'(f(x \cdot w_1) \cdot w_2) \cdot f(x \cdot w_1) \\
\frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} = (y - \hat{y}) \cdot w_3 \cdot f'(f(x \cdot w_1) \cdot w_2) \cdot w_2 \cdot f'(x \cdot w_1) \cdot x
\end{aligned}$$

Выделим в прямоугольники части, которые каждый раз считаются заново, хотя могли бы переиспользоваться.

$$\begin{aligned}
\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_3} = \boxed{(y - \hat{y})} \cdot f(f(x \cdot w_1) \cdot w_2) \\
\frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = \boxed{(y - \hat{y})} \cdot \boxed{w_3 \cdot f'(f(x \cdot w_1) \cdot w_2)} \cdot f(x \cdot w_1) \\
\frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} = \boxed{(y - \hat{y})} \cdot \boxed{w_3 \cdot f'(f(x \cdot w_1) \cdot w_2)} \cdot w_2 \cdot f'(x \cdot w_1) \cdot x
\end{aligned}$$

Если бы слоёв было бы больше, переиспользования возникали бы намного чаще. Градиентный спуск при таком подходе мы могли бы сделать точно также, как и в любых других моделях

$$\begin{aligned}
w_3^t &= w_3^{t-1} - \eta \cdot \frac{\partial L}{\partial w_3}(w_3^{t-1}) \\
w_2^t &= w_2^{t-1} - \eta \cdot \frac{\partial L}{\partial w_2}(w_2^{t-1}) \\
w_1^t &= w_1^{t-1} - \eta \cdot \frac{\partial L}{\partial W_1}(w_1^{t-1}).
\end{aligned}$$

Проблема в том, что такой подход из-за постоянных перевычислений будет работать долго. Алгоритм обратного распространения ошибки помогает более аккуратно считать производную и ускорить обучение нейросетей.

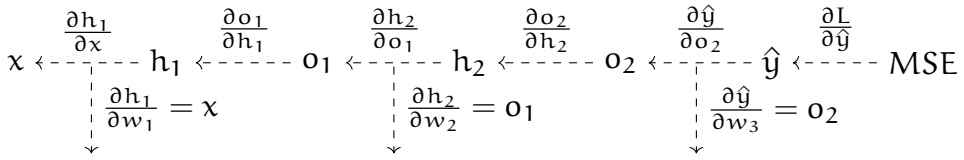
Выпишем алгоритм обратного распространения ошибки. Договоримся до следующих обозначений. Буквами h^k будем обозначать выход k—го слоя до применения функции активации. Буквами o^k будем обозначать всё то же самое после применения функции активации. Например, для первого слоя:

$$h_i^1 = w_1 \cdot x_i$$
$$o_i^1 = f(h_i^1).$$

Сначала мы делаем прямой проход по нейросети (forward pass):

$$x \xrightarrow{w_1} h_1 \xrightarrow{f} o_1 \xrightarrow{w_2} h_2 \xrightarrow{f} o_2 \xrightarrow{w_3} \hat{y} \longrightarrow L(y, \hat{y})$$

Наша нейросеть — граф вычислений. Давайте запишем для каждого ребра в рамках этого графа производную.



Мы везде работаем со скалярами. Все производные довольно просто найти по графу, на котором мы делаем прямой проход. Например,

$$\frac{\partial h_2}{\partial w_2} = \frac{\partial (o_2 \cdot w_2)}{\partial w_2} = o_2.$$

Если в качестве функции активации мы используем сигмоиду

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z},$$

тогда

$$\frac{\partial \sigma}{\partial z} = \left(\frac{e^z}{1 + e^z}\right)' = \frac{e^z}{1 + e^z} - \frac{e^z}{(1 + e^z)^2} \cdot e^z = \frac{e^z}{1 + e^z}\left(1 - \frac{e^z}{1 + e^z}\right) = \sigma(z)(1 - \sigma(z)).$$

Получается, что

$$\frac{\partial o_2}{\partial h_2} = \sigma'(h_2) = \sigma(h_2) \cdot (1 - \sigma(h_2)) = o_2 \cdot (1 - o_2).$$

Осталось только аккуратно записать алгоритм. В ходе прямого прохода мы запоминаем все промежуточные результаты. Они нам пригодятся для поиска производных при обратном проходе. Например, выше, в сигмоиде, при поиске производной, используется результат прямого прохода o_2.

Заведём для накопленного значения производной переменную d. На первом шаге нам надо найти $\frac{\partial L}{\partial w_3}$. Сделаем это в два хода

$$d = \frac{\partial L}{\partial \hat{y}}$$
$$\frac{\partial L}{\partial w_3} = d \cdot o_2.$$

Для поиска производной $\frac{\partial L}{\partial w_2}$ переиспользуем значение, которое накопилось в d. Нам надо найти

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2} = d \cdot \boxed{\frac{\partial \hat{y}}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_2}} \cdot \frac{\partial h_2}{\partial w_2}.$$

Часть, выделенную в прямоугольник мы будем переиспользовать для поиска $\frac{\partial L}{\partial w_1}$. Хорошо бы дописать её в d для этого. Получается, вторую производную тоже надо найти в два хода

$$d = d \cdot \frac{\partial \hat{y}}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_2}$$
$$\frac{\partial L}{\partial w_2} = d \cdot o_1.$$

Осталась заключительная производная $\frac{\partial L}{\partial w_1}$. Нам надо найти

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} = d \cdot \frac{\partial h_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}.$$

Снова делаем это в два шага

$$d = d \cdot \frac{\partial h_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_1}$$
$$\frac{\partial L}{\partial w_1} = d \cdot x.$$

Если бы нейросетка была бы глубже, мы смогли бы переиспользовать d на следующих слоях. Каждую производную мы нашли ровно один раз. **Это и есть алгоритм обратного распространения ошибки.** В случае матриц происходит всё ровно то же самое, но дополнительно

надо проследить за всеми размерностями и более аккуратно перемножить матрицы.

Упражнение 21 (сигмоида)

В неглубоких сетях в качестве функции активации можно использовать сигмоиду

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z},$$

Маша хочет использовать сигмоиду внутри нейросети. Предполагается, что после прямого шага, наши вычисления будут использованы в другой части нейросети. В конечном итоге, по выходу из нейросети мы вычислим какую-то функцию потерь L.

У сигмоиды нет параметров. Чтобы обучить нейросеть, Маше понадобится производная $\frac{\partial L}{\partial z}$. Выпишите её в матричном виде через производные $\frac{\partial L}{\partial \sigma}$ и $\frac{\partial \sigma}{\partial z}$.

Решение:

При решении предыдущей задачи мы выяснили, что $\sigma'(z) = \sigma(z)\cdot(1-\sigma(z))$. Тогда по цепному правилу

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \sigma}\cdot\frac{\partial \sigma}{\partial z} = \frac{\partial L}{\partial \sigma}\cdot\sigma(z)\cdot(1-\sigma(z)).$$

Получается, при прямом проходе мы вычисляем сигмоиду по формуле из условия. При обратном проходе мы умножаем пришедшую к нам производную на производную сигмоиды. Если на вход приходит матрица, мы берём сигмоиду от каждого её элемента. Если на вход приходит матрица $Z_{[n\times k]}$, на выходе мы получаем матрицу $\Sigma_{[n\times k]}$.

Когда мы берём сигмоиду от матрицы, мы применяем функцию к каждому её элементу. из-за этого, в производной все умножения мы делаем поэлементно, то есть матрица $\Sigma * (1-\Sigma)$ останется размера $[n \times k]$. Когда мы применяем цепное правило, под $\frac{\partial L}{\partial \Sigma}$ мы подразумеваем производную функции потерь L по каждому элементу матрицы Σ. Получается, что это матрица размера $[n\times k]$. Её мы поэлементно умножаем на $\Sigma*(1-\Sigma)$ и снова получаем матрицу размера $[n \times k]$. Все размерности оказываются соблюдены.

Упражнение 22 (линейный слой)

Маша знает, что главный слой в нейронных сетях — линейный. В матричном виде его можно записать как $Z = XW$.

Маша хочет использовать этот слой внутри нейросети. Предполагается, что после прямого шага наши вычисления будут использованы в другой части нейросети. В конечном итоге, по выходу из нейросети мы вычислим какую-то функцию потерь L.

Чтобы обучить нейросеть, Маше понадобятся производные $\frac{\partial L}{\partial X}$ и $\frac{\partial L}{\partial W}$. Аккуратно найдите их и запишите в матричном виде[13]. Предполагается, что

[13] https://web.eecs.umich.edu/~justincj/teaching/eecs442/notes/linear-backprop.html

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \qquad W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$Z = XW = \begin{pmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \end{pmatrix} = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} & x_{11}w_{12} + x_{12}w_{22} & x_{11}w_{13} + x_{12}w_{23} \\ x_{21}w_{11} + x_{22}w_{21} & x_{21}w_{12} + x_{22}w_{22} & x_{21}w_{13} + x_{22}w_{23} \end{pmatrix}$$

Решение:

При обратном распространении ошибки мы предполагаем, что производная $\frac{\partial L}{\partial Z}$ у нас уже есть. Так как Z — это матрица размера 2×3, эта производная будет выглядеть как

$$\frac{\partial L}{\partial Z} = \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix}.$$

По цепному правилу мы можем использовать $\frac{\partial L}{\partial Z}$ для поиска интересующих нас градиентов

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Z} \cdot \frac{\partial Z}{\partial X} \qquad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z} \cdot \frac{\partial Z}{\partial W}.$$

Нужно, чтобы у матриц совпали размерности. Производные $\frac{\partial Z}{\partial X}$ и $\frac{\partial Z}{\partial W}$ — это матрицы Якоби нашего линейного слоя. Пусть W это параметры, а X аргумент функции. Функция $f(X) = XW$ бьёт из пространства матриц $X_{[2\times 2]}$ в пространство матриц $Z_{[2\times 3]}$. Нам надо взять производную от каждого элемента матрицы Z по каждому элементу из матрицы X. Всего получится 24 производных. По правилам из матана мы должны будем записать их в виде четырёхмерной матрицы[14]. Это жутко неудобно.

К счастью, многие производные будут нулевыми. Поэтому мы можем схитрить, сначала найти $\frac{\partial L}{\partial X}$,

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \quad \Rightarrow \quad \frac{\partial L}{\partial X} = \begin{pmatrix} \frac{\partial L}{\partial x_{11}} & \frac{\partial L}{\partial x12} \\ \frac{\partial L}{\partial x_{21}} & \frac{\partial L}{\partial x_{22}} \end{pmatrix},$$

а затем написать удобные формулы в общем виде. Найдём $\frac{\partial L}{\partial x_{11}}$ с помощью цепного правила

$$\frac{\partial L}{\partial x_{11}} = \sum_{i=1}^{n} \sum_{j=1}^{d} \frac{\partial L}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial x_{11}} = \langle \frac{\partial L}{\partial Z}, \frac{\partial Z}{\partial x_{11}} \rangle.$$

Работать с суммами неудобно. Мы помним, что $\frac{\partial L}{\partial Z}$ и $\frac{\partial Z}{\partial x_{11}}$ — матрицы из производных. Поэтому сумму можно записать в виде скалярного произведения матриц. Мы должны в нём умножить элементы матриц друг на друга, а затем сложить. Давайте найдём производную

[14]Про это можно более подробно почитать в разделе про матричные производные.

матрицы Z по x_{11}

$$Z = XW = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} & x_{11}w_{12} + x_{12}w_{22} & x_{11}w_{13} + x_{12}w_{23} \\ x_{21}w_{11} + x_{22}w_{21} & x_{21}w_{12} + x_{22}w_{22} & x_{21}w_{13} + x_{22}w_{23} \end{pmatrix}.$$

Переменная x_{11} фигурирует только в первой строке

$$\frac{\partial Z}{\partial x_{11}} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ 0 & 0 & 0 \end{pmatrix}.$$

Выходит, что

$$\frac{\partial L}{\partial x_{11}} = \left\langle \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix}, \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ 0 & 0 & 0 \end{pmatrix} \right\rangle = \frac{\partial L}{\partial z_{11}} \cdot w_{11} + \frac{\partial L}{\partial z_{12}} \cdot w_{12} + \frac{\partial L}{\partial z_{13}} \cdot w_{13}.$$

По аналогии мы можем найти оставшиеся три производные. Например,

$$\frac{\partial L}{\partial x_{21}} = \left\langle \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ w_{11} & w_{12} & w_{13} \end{pmatrix} \right\rangle = \frac{\partial L}{\partial z_{21}} \cdot w_{11} + \frac{\partial L}{\partial z_{22}} \cdot w_{12} + \frac{\partial L}{\partial z_{23}} \cdot w_{13}.$$

Попробуем выписать $\frac{\partial L}{\partial X}$ через $\frac{\partial L}{\partial Z}$ и W

$$\begin{aligned} \frac{\partial L}{\partial X} &= \begin{pmatrix} \frac{\partial L}{\partial x_{11}} & \frac{\partial L}{\partial x12} \\ \frac{\partial L}{\partial x_{21}} & \frac{\partial L}{\partial x_{22}} \end{pmatrix} = \\ &= \begin{pmatrix} \frac{\partial L}{\partial z_{11}} \cdot w_{11} + \frac{\partial L}{\partial z_{12}} \cdot w_{12} + \frac{\partial L}{\partial z_{13}} \cdot w_{13} & \frac{\partial L}{\partial z_{11}} \cdot w_{21} + \frac{\partial L}{\partial z_{12}} \cdot w_{22} + \frac{\partial L}{\partial z_{13}} \cdot w_{23} \\ \frac{\partial L}{\partial z_{21}} \cdot w_{11} + \frac{\partial L}{\partial z_{22}} \cdot w_{12} + \frac{\partial L}{\partial z_{23}} \cdot w_{13} & \frac{\partial L}{\partial z_{21}} \cdot w_{21} + \frac{\partial L}{\partial z_{22}} \cdot w_{22} + \frac{\partial L}{\partial z_{23}} \cdot w_{23} \end{pmatrix} = \\ &= \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix} \cdot \begin{pmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{pmatrix} = \frac{\partial L}{\partial Z} W^T \end{aligned}$$

Нам повезло! Наша хитрость увенчалась успехом, и нам удалось записать нашу формулу в виде произведения двух матриц без вычисления четырёхмерных якобианов.

Провернём ровно такой же фокус с поиском производной $\frac{\partial L}{\partial W}$.

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \quad \Rightarrow \quad \frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix}.$$

По аналогии с предыдущей производной

$$\frac{\partial L}{\partial w_{kl}} = \sum_{i=1}^{n} \sum_{j=1}^{d} \frac{\partial L}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial w_{kl}} = \langle \frac{\partial L}{\partial Z}, \frac{\partial Z}{\partial w_{kl}} \rangle.$$

По матрице

$$Z = XW = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} & x_{11}w_{12} + x_{12}w_{22} & x_{11}w_{13} + x_{12}w_{23} \\ x_{21}w_{11} + x_{22}w_{21} & x_{21}w_{12} + x_{22}w_{22} & x_{21}w_{13} + x_{22}w_{23} \end{pmatrix}$$

мы можем найти все требуемые производные

$$\frac{\partial Z}{\partial w_{11}} = \begin{pmatrix} x_{11} & 0 & 0 \\ x_{21} & 0 & 0 \end{pmatrix} \quad \frac{\partial Z}{\partial w_{12}} = \begin{pmatrix} 0 & x_{11} & 0 \\ 0 & x_{21} & 0 \end{pmatrix} \quad \frac{\partial Z}{\partial w_{13}} = \begin{pmatrix} 0 & 0 & x_{11} \\ 0 & 0 & x_{21} \end{pmatrix}$$
$$\frac{\partial Z}{\partial w_{21}} = \begin{pmatrix} x_{12} & 0 & 0 \\ x_{22} & 0 & 0 \end{pmatrix} \quad \frac{\partial Z}{\partial w_{22}} = \begin{pmatrix} 0 & x_{12} & 0 \\ 0 & x_{22} & 0 \end{pmatrix} \quad \frac{\partial Z}{\partial w_{23}} = \begin{pmatrix} 0 & 0 & x_{12} \\ 0 & 0 & x_{22} \end{pmatrix}.$$

Чтобы найти $\frac{\partial L}{\partial w_{kl}}$ нам надо посчитать между матрицами $\frac{\partial L}{\partial Z}$ и $\frac{\partial Z}{\partial w_{kl}}$ скалярное произведение. Например,

$$\frac{\partial L}{\partial w_{21}} = \left\langle \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix}, \begin{pmatrix} x_{12} & 0 & 0 \\ x_{22} & 0 & 0 \end{pmatrix} \right\rangle = \frac{\partial L}{\partial z_{11}} \cdot x_{12} + \frac{\partial L}{\partial z_{11}} \cdot x_{22}.$$

Получается, что всю матрицу $\frac{\partial L}{\partial W}$ целиком можно найти как

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix} =$$
$$= \begin{pmatrix} \frac{\partial L}{\partial z_{11}} \cdot x_{11} + \frac{\partial L}{\partial z_{21}} \cdot x_{21} & \frac{\partial L}{\partial z_{12}} \cdot x_{11} + \frac{\partial L}{\partial z_{22}} \cdot x_{21} & \frac{\partial L}{\partial z_{13}} \cdot x_{11} + \frac{\partial L}{\partial z_{23}} \cdot x_{21} \\ \frac{\partial L}{\partial z_{11}} \cdot x_{12} + \frac{\partial L}{\partial z_{21}} \cdot x_{22} & \frac{\partial L}{\partial z_{12}} \cdot x_{12} + \frac{\partial L}{\partial z_{22}} \cdot x_{22} & \frac{\partial L}{\partial z_{13}} \cdot x_{12} + \frac{\partial L}{\partial z_{23}} \cdot x_{22} \end{pmatrix} =$$
$$= \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} & \frac{\partial L}{\partial z_{13}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} & \frac{\partial L}{\partial z_{23}} \end{pmatrix} = X^T \frac{\partial L}{\partial Z}.$$

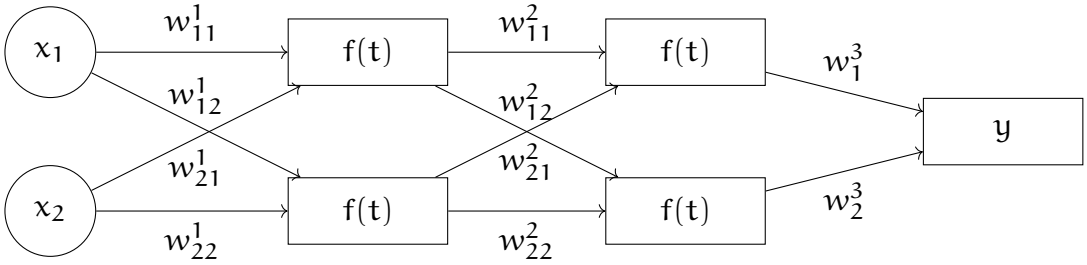
Таким образом, для линейного слоя, мы всегда можем посчитать производные как

$$\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Z} \qquad \frac{\partial L}{\partial X} = \frac{\partial L}{\partial Z} W^T.$$

Дальше под $\frac{\partial Z}{\partial W}$ будем всегда подразумевать X^T, а под $\frac{\partial Z}{\partial X}$ будем иметь в виду W^T.

Упражнение 23 (Backpropagation в матричном виде)

У Маши есть нейросеть с картинки ниже, где w_{ij}^k — веса для k слоя, $f(t)$ — какая-то функция активации. Маша хочет научиться делать для такой нейронной сетки градиентный спуск.



а. Запишите Машину нейросеть, как сложную функцию. Сначала в виде нескольких уравнений, а затем в матричном виде.

б. Выпишите все производные в том виде, в котором их было бы удобно использовать для алгоритма обратного распространения ошибки, а затем, сформулируйте сам алгоритм. Нарисуйте под него удобную схемку.

Решение:

Договоримся до следующих обозначений. Буквами h_{ij}^k будем обозначать выход k—го слоя для j—го нейрона для i—го наблюдения до применения функции активации. Буквами o_{ij}^k будем обозначать всё то же самое после применения функции активации. Например, для первого слоя:

$$h_{i1}^1 = w_{11}^1 \cdot x_{i1} + w_{21}^1 \cdot x_{i2}$$
$$o_{i1}^1 = f(h_{i1}^1).$$

Делай раз. Для начала перепишем сетку в виде нескольких уравнений. Для первого слоя мы находим

$$o_{i1}^1 = f(w_{11}^1 \cdot x_{i1} + w_{21}^1 \cdot x_{i2})$$
$$o_{i2}^1 = f(w_{12}^1 \cdot x_{i1} + w_{22}^1 \cdot x_{i2}).$$

Для второго работают аналогичные уравнения, но значения x заменяются на соответствующие o. На выходе мы предсказываем y, как взвешенную суммы выходов со второго слоя

$$\hat{y}_i = w_1^3 \cdot o_{i1}^2 + w_2^3 \cdot o_{i2}^2.$$

Подставим вместо o^2_{1i} и o^2_{2i} результат вычисления предыдущих слоёв

$$\hat{y}_i = w^3_1 \cdot f(w^1_{12} \cdot o^1_{i1} + w^1_{22} \cdot o^1_{i2}) + w^3_2 \cdot f(w^1_{12} \cdot o^1_{i1} + w^1_{22} \cdot o^1_{i2}).$$

Подставим результат вычисления первого слоя

$$\begin{aligned}\hat{y}_i = w^3_1 \cdot f(w^1_{12} \cdot f(w^1_{11} \cdot x_{i1} + w^1_{21} \cdot x_{i2}) + w^1_{22} \cdot f(w^1_{12} \cdot x_{i1} + w^1_{22} \cdot x_{i2}))+ \\ + w^3_2 \cdot f(w^1_{12} \cdot f(w^1_{11} \cdot x_{i1} + w^1_{21} \cdot x_{i2}) + w^1_{22} \cdot f(w^1_{12} \cdot x_{i1} + w^1_{22} \cdot x_{i2})).\end{aligned}$$

Мы записали нашу нейросеть в виде сложной функции. Выглядит ужасно.

Давайте перепишем всё то же самое более компактно, в матричном виде. Начнём с первого слоя. На самом деле, чтобы найти строчку (h^1_{i1}, h^1_{i2}) мы делаем матричное умножение. Строчку (x_{i1}, x_{i2}) мы умножаем на матрицу весов W_1

$$\begin{pmatrix} h^1_{i1} & h^1_{i2} \end{pmatrix} = \begin{pmatrix} x_{i1} & x_{i2} \end{pmatrix} \cdot \begin{pmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{pmatrix}.$$

Чтобы получить h_1 мы умножаем строчку из переменных на первый столбец, чтобы получить h_2, на второй столбец. Получается, что в терминах матриц каждый нейрон нашей сети — это столбец. Если мы добавим ещё один столбец из весов в матрицу, это будет эквивалентно добавлению в сетку третьего нейрона, так как на выходе мы будем получать ещё и h_3. Если у нас появится дополнительный вход x_3, в матрицу нам нужно будет добавить ещё одну строчку из весов.

Запишем первый слой в матричном виде. На вход идёт матрица из наблюдений $X_{[n\times 2]}$, она умножается на матрицу весов $W_{[2\times 2]}$, получается матрица $H_{[n\times 2]}$. Ко всем элементам этой матрицы мы применяем функцию активации f. Делаем это поэлементно

$$O_1 = f(H_1) = f(X \cdot W_1).$$

Остальные слои записываются по аналогии. Получается, что наша нейросеть в матричном виде выглядит как

$$\hat{y} = f(f(X \cdot W_1) \cdot W_2) \cdot W_3.$$

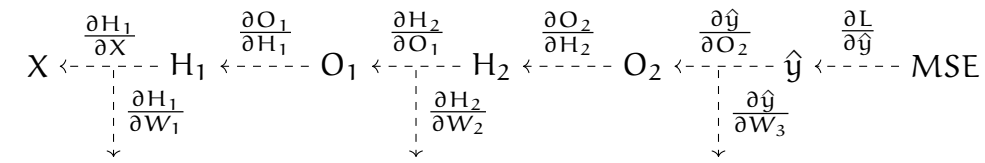
Здесь $\hat{y}$ — вектор столбец размера $[n \times 1]$, а матрицы весов обладают размерностями $[2 \times 2]$, $[2 \times 2]$ и $[2 \times 1]$ соответственно.

Делай два. Выпишем алгоритм обратного распространения ошибки в виде красивой схемки. Сначала мы делаем прямой проход по нейросети (forward pass):

$$\underset{[n\times 2]}{X} \xrightarrow[[2\times 2]]{W_1} \underset{[n\times 2]}{H_1} \xrightarrow{f} \underset{[n\times 2]}{O_1} \xrightarrow[[2\times 2]]{W_2} \underset{[n\times 2]}{H_2} \xrightarrow{f} \underset{[n\times 2]}{O_2} \xrightarrow[[2\times 1]]{W_3} \underset{[n\times 1]}{\hat{y}} \longrightarrow L(y, \hat{y})$$

Под всеми матрицами подписаны размерности. Взятие функции активации — поэлементная операция, она никак не меняет размер матрицы. Это будет важно при взятии производных. В ходе прямого прохода мы запоминаем все промежуточные результаты. Они нам пригодятся для поиска производных при обратном проходе. Например, $\frac{\partial H_2}{\partial W_2} = O_1^T$. Получается, в какой-то момент нам надо будет переиспользовать результаты вычислений, полученных при прямом проходе.

Наша нейросеть — граф вычислений. Давайте запишем для каждого ребра в рамках этого графа производную.



Осталось только аккуратно записать обратный ход алгоритма. Заведём для накопленного значения производной переменную d. На первом шаге нам надо найти $\frac{\partial L}{\partial W_3}$. Не будем забывать, как выглядят производные для линейного слоя, полученные в предыдущей задаче. Сделаем это в два хода

$$d = \frac{\partial L}{\partial \hat{y}}$$
$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_3} = O_2^T \cdot d.$$

Матрица O_2^T будет размера $[2 \times n]$, вектор d будет размера $[n \times 1]$, размер производной будет $[2 \times 1]$, что совпадает с размером матрицы W_3.

Для поиска производной $\frac{\partial L}{\partial W_2}$ переиспользуем значение, которое накопилось в d

$$d = \frac{\partial L}{\partial H_2} = d \cdot \frac{\partial \hat{y}}{\partial O_2} \cdot \frac{\partial O_2}{\partial H_2} = d \cdot W_3^T * f'(H_2)$$
$$\frac{\partial L}{\partial W_2} = O_1^T \cdot d.$$

Размер d был $[n \times 1]$, после персчёта стал $[n \times 1] \cdot [1 \times 2] * [n \times 2] = [n \times 2]$. Поэлементное умножение на производную функции активации не повлияло на размер матрицы. Размер производной $\frac{\partial L}{\partial W_2}$ оказывается $[2 \times 2]$, что совпадает с размером матрицы W_2.

Осталась заключительная производная $\frac{\partial L}{\partial W_1}$. Нам надо найти

$$d = \frac{\partial L}{\partial H_1} = \frac{\partial L}{\partial H_2} \cdot \frac{\partial H_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial H_1} = d \cdot W_2^T * f'(H_1)$$
$$\frac{\partial L}{\partial W_1} = X^T \cdot d.$$

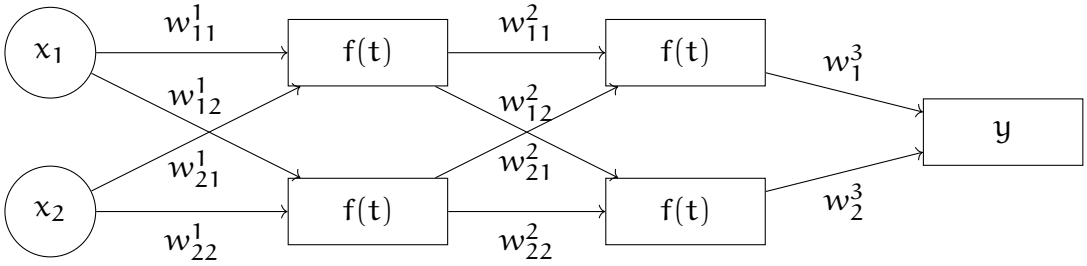
Если бы сетка была глубже, мы продолжили бы переиспользовать d. Каждую производную мы посчитали один раз. Такой алгоритм обучения линеен по числу параметров.

Упражнение 24 (Backpropagation своими руками)

У Маши есть нейросеть с картинки ниже. Она использует функцию потерь

$$L(W_1, W_2, W_3) = \frac{1}{2} \cdot (\hat{y} - y)^2.$$

В качестве функции активации Маша выбрала сигмоиду $\sigma(t) = \frac{e^t}{1+e^t}$.



Выпишите для Машиной нейросетки алгоритм обратного распространения ошибки в общем виде. Пусть Маша инициализировала веса нейронной сети нулями. У неё есть два наблюдения

№	x_1	x_2	y
1	1	1	1
2	5	2	0

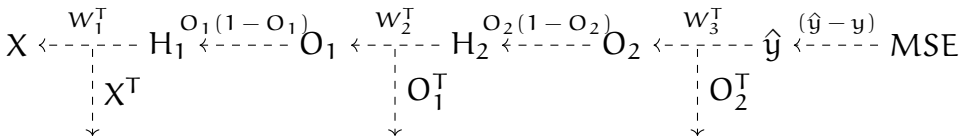
Сделайте руками два шага алгоритма обратного распространения ошибки. Пусть скорость обучения $\eta = 1$. Стохастический градиентный спуск решил, что сначала для шага будет использоваться второе наблюдение, а затем первое. Объясните, почему инициализировать веса нулями — плохая идея. Почему делать инициализацию весов любой другой константой — плохая идея?

Решение:

Для начала запишем алгоритм в общем виде. Для этого нам надо взять схему из предыдущей задачи и записать там все производные. Для сигмоиды $\sigma'(t) = \sigma(t) \cdot (1 - \sigma(t))$. Прямой проход по нейронной сети (forward pass):

$$X \xrightarrow{W_1} H_1 \xrightarrow{\sigma} O_1 \xrightarrow{W_2} H_2 \xrightarrow{\sigma} O_2 \xrightarrow{W_3} \hat{y} \longrightarrow MSE$$

Обратный проход по нейронной сети (backward pass):



По аналогии с предыдущей задачей выпишем формулы для обратного распространения ошибки:

$$d = (\hat{y} - y) \qquad d = d \cdot W_3^T * O_2 * (1 - O_2) \qquad d = d \cdot W_2^T * O_1 * (1 - O_1)$$
$$\frac{\partial MSE}{\partial W_3} = O_2^T \cdot d \qquad \frac{\partial MSE}{\partial W_2} = O_1^T \cdot d \qquad \frac{\partial MSE}{\partial W_1} = X^T \cdot d$$

Когда мы аккуратно подставим все числа, можно будет сделать шаг SGD

$$W_3^t = W_3^{t-1} - \eta \cdot \frac{\partial MSE}{\partial W_3} \qquad W_2^t = W_2^{t-1} - \eta \cdot \frac{\partial MSE}{\partial W_2} \qquad W_1^t = W_1^{t-1} - \eta \cdot \frac{\partial MSE}{\partial W_1}$$

Сделаем шаг SGD для второго наблюдения. Делаем прямое распространение для второго наблюдения, напомним, что матрицы весов инициализированы нулями:

$$(5, 2) \xrightarrow{W_1} (0, 0) \xrightarrow{\sigma} (0.5, 0.5) \xrightarrow{W_2} (0, 0) \xrightarrow{\sigma} (0.5, 0.5) \xrightarrow{W_3} 0 \longrightarrow 0.5 \cdot (0 - 1)^2$$

Делаем обратный проход.

Шаг 1:

$$d = (\hat{y} - y) = -1$$
$$\frac{\partial MSE}{\partial W_3} = O_2^T \cdot d = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \cdot (-1) = \begin{pmatrix} -0.5 \\ -0.5 \end{pmatrix}$$

Шаг 2:

$$d = d \cdot W_3^T * O_2 * (1 - O_2) = -1 \cdot (0, 0) * (0.5, 0.5) * (0.5, 0.5) = (0, 0)$$
$$\frac{\partial MSE}{\partial W_2} = O_1^T \cdot d = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \cdot (0, 0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Шаг 3:

$$d = d \cdot W_2^T * O_1 * (1 - O_1) = (0, 0) \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} * (0.5, 0.5) * (0.5, 0.5) = (0, 0)$$
$$\frac{\partial MSE}{\partial W_1} = X^T \cdot d = \begin{pmatrix} 5 \\ 2 \end{pmatrix} \cdot (0, 0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Делаем шаг градиентного спуска

$$W_1^1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$
$$W_2^1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$
$$W_3^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1 \cdot \begin{pmatrix} -0.5 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

Сделаем шаг SGD для первого наблюдения. Делаем прямое распространение для второго наблюдения, напомним, что матрицы весов инициализированы нулями:

$$(1,1) \xrightarrow{W_1} (0,0) \xrightarrow{\sigma} (0.5,0.5) \xrightarrow{W_2} (0,0) \xrightarrow{\sigma} (0.5,0.5) \xrightarrow{W_3} 0.5 \longrightarrow 0.5 \cdot (0.5-0)^2$$

Делаем обратный проход.

Шаг 1:

$$d = (\hat{y} - y) = 0.5$$
$$\frac{\partial MSE}{\partial W_3} = O_2^T \cdot d = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \cdot (0.5) = \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix}$$

Шаг 2:

$$d = d \cdot W_3^T * O_2 * (1 - O_2) = 0.5 \cdot (0.5, 0.5) * (0.5, 0.5) * (0.5, 0.5) = (1/16, 1/16)$$
$$\frac{\partial MSE}{\partial W_2} = O_1^T \cdot d = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \cdot (1/16, 1/16) = \begin{pmatrix} 1/32 & 1/32 \\ 1/32 & 1/32 \end{pmatrix}$$

Шаг 3:

$$d = d \cdot W_2^T * O_1 * (1 - O_1) = (1/16, 1/16) \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} * (0.5, 0.5) * (0.5, 0.5) = (0, 0)$$
$$\frac{\partial MSE}{\partial W_1} = X^T \cdot d = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot (0, 0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

На этой задаче видно, как сигмоида способствует затуханию градиента. Её производная по абсолютной величине всегда принимает значения меньше 1. Из-з этого значение d от слоя к слою становится всё меньше и меньше. Чем ближе к началу нашей сети мы находимся, тем на меньшую величину шагают веса. Если сетка оказывается очень глубокой, такой эффект ломает её обучение. Его обычно называют **параличом нейронной сети.** Именно из-за этого сигмоиду обычно не используют в глубоких архитектурах.

Делаем шаг градиентного спуска

$$W_3^2 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix}$$
$$W_2^2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 1 \cdot \begin{pmatrix} 1/32 & 1/32 \\ 1/32 & 1/32 \end{pmatrix} = \begin{pmatrix} -1/32 & -1/32 \\ -1/32 & -1/32 \end{pmatrix}$$
$$W_1^1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Из-за того, что мы инициализировали веса нулями, слои поначалу учатся по-очереди. Пока мы не сдвинем веса более поздних слоёв, веса более ранних слоёв не сдвинутся. Это замедляет обучение. Обратите внимание, что все веса меняются на одну и ту же величину в одном и том же направлении. При инициализации любой другой константой этот эффект сохраниться. Нам хочется, чтобы после обучения нейроны внутри сетки были максимально разнообразными. Для этого веса лучше инициализировать случайно. В будущем мы обсудим грамотные способы инициализации, которые не портят обучение.

Упражнение 25 (Незаметный backpropagation)

Маша собрала нейросеть:

$$y = \max\left(0; X \cdot \begin{pmatrix} 1 & -1 \\ 0.5 & 0 \end{pmatrix}\right) \cdot \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

Теперь Маша внимательно смотрит на неё.

а) Первый слой нашей нейросетки — линейный. По какой формуле делается forward pass? Сделайте его для матрицы

$$X = \begin{pmatrix} 1 & 2 \\ -1 & 2 \end{pmatrix}.$$

б) Найдите для первого слоя производную выхода по входу. При обратном движении по нейросетке, в первый слой пришёл накопленный градиент

$$d = \begin{pmatrix} -0.5 & 0 \\ 0 & 0 \end{pmatrix}.$$

Каким будет новое накопленное значение градиента, которое выплюнет из себя линейный слой? По какой формуле делается backward pass?

в) Второй слой нейросетки — функция активации, ReLU. По какой формуле делается forward

pass? Сделайте его для матрицы

$$H_1 = \begin{pmatrix} 2 & -0.5 \\ 0 & 1 \end{pmatrix}.$$

г) Найдите для второго слоя производную выхода по входу. При обратном движении по нейросетке во второй слой пришёл накопленный градиент

$$d = \begin{pmatrix} -0.5 & -1 \\ 0 & 0 \end{pmatrix}.$$

Каким будет новое накопленное значение градиента, которое выплюнет из себя ReLU? По какой формуле делается backward pass?

д) Третий слой нейросетки — линейный. По какой формуле делается forward pass? Сделайте его для матрицы

$$O_1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$

е) Найдите для третьего слоя производную выхода по входу. При обратном движении по нейросетке, в третий слой пришёл накопленный градиент $d = (-1, 0)^T$. Каким будет новое накопленное значение градиента, которое выплюнет из себя линейный слой?

ж) Мы решаем задачу Регрессии. В качестве функции ошибки мы используем

$$MSE = \frac{1}{2n} \sum (\hat{y}_i - y_i)^2.$$

Пусть для рассматриваемых наблюдений реальные значения $y_1 = 2, y_2 = 1$. Найдите значение MSE.

з) Чему равна производная MSE по прогнозу? Каким будет накопленное значение градиента, которое MSE выплюнет из себя в предыдущий слой нейросетки?

и) Пусть скорость обучения $\gamma = 1$. Сделайте для весов нейросети шаг градиентного спуска.

к) Посидела Маша, посидела, и поняла, что неправильно она всё делает. В реальности перед ней не задача регрессии, а задача классификации. Маша применила к выходу из нейросетки сигмоиду. Как будет для неё выглядеть forward pass?

л) В качестве функции потерь Маша использует logloss . Как для этой функции потерь выглядит forward pass? Сделайте его.

м) Найдите для logloss производную прогнозов по входу в сигмоиду. Как будет выглядеть backward pass, если $y_1 = 0, y_2 = 1$? Как поменяется оставшаяся часть алгоритма обратного распространения ошибки?

Решение:

Весь путь по нейросети от начала к концу, то есть forward pass будет выглядеть следующим

образом:

$$H_1 = X \cdot W_1 = \begin{pmatrix} 1 & 2 \\ -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 \\ 0.5 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -0.5 \\ 0 & 1 \end{pmatrix}$$

$$O_1 = ReLU(H_1) = \begin{pmatrix} \max(0,2) & \max(0,-0.5) \\ \max(0,0) & \max(0,1) \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\hat{y} = O_1 \cdot W_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$MSE = \frac{1}{4} \cdot ((1-2)^2 + (1-1)^2) = 0.25$$

Все необходимые для обратного прохода производные выглядят как

$$\frac{\partial MSE}{\partial \hat{y}} = \begin{pmatrix} \hat{y}_1 - y_1 \\ \hat{y}_2 - y_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\frac{\partial \hat{y}}{\partial O_1} = W_2^T = (0.5, 1) \qquad \frac{\partial \hat{y}}{\partial W_2} = O_1^T = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\frac{\partial O_1}{\partial H_1} = [H_{ij} > 0] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\frac{\partial H_1}{\partial X} = W_1^T = \begin{pmatrix} 1 & 0.5 \\ -1 & 0 \end{pmatrix} \qquad \frac{\partial H_1}{\partial W_1} = X^T = \begin{pmatrix} 1 & -1 \\ 2 & 2 \end{pmatrix}$$

Когда мы считаем производную MSE, мы ищем её по каждому прогнозу. В случае производной для ReLU запись $[H_{ij} > 0]$ означает, что на месте ij стоит 1, если элемент больше нуля и ноль иначе. Делаем шаг обратного распространения ошибки

$$d = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\frac{\partial MSE}{\partial W_2} = O_1^T \cdot d = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

$$d = d \cdot W_2^T * [H_{ij} > 0] = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \cdot (0.5, 1) * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -0.5 & -1 \\ 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -0.5 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\frac{\partial MSE}{\partial W_1} = X^T \cdot d = \begin{pmatrix} 1 & -1 \\ 2 & 2 \end{pmatrix} \cdot \begin{pmatrix} -0.5 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -0.5 & 0 \\ -1 & 0 \end{pmatrix}$$

Делаем шаг градиентного спуска

$$W_1 = \begin{pmatrix} 1 & -1 \\ 0.5 & 0 \end{pmatrix} - \gamma \cdot \begin{pmatrix} -0.5 & -1 \\ 0 & 0 \end{pmatrix}$$
$$W_2 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} - \gamma \cdot \begin{pmatrix} -2 \\ 0 \end{pmatrix}.$$

Меняем MSE на logloss и добавляем сигмоиду. Производная для сигмоиды выглядит как

$$\text{logloss} = y_i \cdot \ln \hat{p}_i + (1 - y_i) \cdot (1 - \hat{p}_i)$$
$$\frac{\partial \, \text{logloss}}{\partial \hat{p}_i} = \frac{y_i}{\hat{p}_i} - \frac{1 - y_i}{1 - \hat{p}_i}.$$

Так как в бинарной классификации y_i принимает значения $\{0, 1\}$, производная равна либо первому либо второму слагаемому. Получаем вычисления

$$O_2 = \sigma(\hat{y}) = \begin{pmatrix} 0.73 \\ 0.73 \end{pmatrix} \qquad \frac{\partial O_2}{\partial \hat{y}} = O_2 \cdot (1 - O_2) \approx \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix}$$
$$\frac{\partial \, \text{logloss}}{\partial \hat{p}} \approx \begin{pmatrix} 3.7 \\ 1.4 \end{pmatrix} \qquad \frac{\partial \, \text{logloss}}{\partial \hat{y}} \approx \begin{pmatrix} 3.7 \\ 1.4 \end{pmatrix} * \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 0.28 \end{pmatrix}$$

Дальше алгоритм делается ровно также, только в качестве стартового d используется $\text{logloss}'_{\hat{y}}$, а не $\text{MSE}'_{\hat{y}}$.

Упражнение 26 (Нестеров и backprop)

К Маше приехал её папа и загрузил её интересным вопросом. В алгоритме обратного распространения ошибки мы можем делать шаг как минимум двумя способами:

а. Зафиксировали все w_{t-1}, нашли все градиенты, сделали сразу по всем весам шаг градиентного спуска.

б. Нашли градиенты для последнего слоя и сделали шаг для его весов, получили w_t^k. Для поиска градиентов предпоследнего слоя используем веса w_t^k, а не w_{t-1}^k. Все остальные слои обновляем по аналогии.

Как думаете, какой из способов будет приводить к более быстрой сходимости и почему[15]?

Решение:

С одной стороны идея чем-то похожа на градиентный спуск с поправкой Нестерова. Возмож-

[15] Я придумал эту задачу и не смог найти статью, где делали бы что-то похожее. Если вы видели такую, пришлите мне её плиз.

но, сходимость ускорится. С другой стороны, градиенты оказываются смещёнными. Если сеть глубокая, в её начале смещение может быть очень большим. Из-за этого сходимость может сломаться.

Листочек 5: всего лишь кубики LEGO

Цитата про лего

автор цитаты

Функции активации

Желание - Ржавый - Семнадцать - Рассвет - Печь - Девять - Добросердечный - Возвращение на Родину - Один - Грузовой вагон.

Код активации Зимнего Солдата

Упражнение 27 (про сигмоиду)

Любую "sобразную функцию называют сигмоидой. Наиболее сильно прославилась под таким названием функция $f(t) = \frac{e^t}{1+e^t}$. Слава о ней добралась до Маши и теперь она хочет немного поисследовать её свойства.

а. Выпишите формулы для forward pass и backward pass через слой с сигмоидой.

б. Какое максимальное значение принимает производная сигмоиды? Объясните как это способствует затуханию градиента и параличу нейронной сети?

Упражнение 28 (про тангенс)

Функция $f(t) = \tanh(t) = \frac{2}{1+e^{-2t}} - 1$ называется гиперболическим тангенсом.

а. Что происходит при $t \to +\infty$? А при $t \to -\infty$?

б. Как связаны между собой $f(t)$ и $f'(t)$?

в. Выпишите формулы для forward pass и backward pass через слой с тангенсом.

г. Правда ли, что тангенс способствует затуханию градиента и параличу нейронной сети? Какое максимальное значение принимает производная тангенса?

д.

пункт про то, почему часто функцию юзают в RNN

Упражнение 29 (про ReLU)

Функция $f(t) = \mathrm{ReLU}(t) = \max(t, 0)$ называется ReLU.

а.

Задача про ReLU и сигмоиду (Николенко)

Задача про паралич сигмоиды и ReLU

Parametric Rectifier (PReLU) Выписать уравнения для бэкпропа по параметру α

Упражнение 30 (softplus)

Про то почему её не используют

Что-то про Mish и Swish

Упражнение 31 (температура генерации)

Иногда в функцию softmax добавляют дополнительный параметр T, который называют температурой. Тогда она приобретает вид

$$f(z) = \frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{K} e^{\frac{z_k}{T}}}$$

Обычно это делается, когда с помощью нейросетки нужно сгенерировать какой-нибудь новый объект. Пусть у нас есть три класса. Наша нейросеть выдала на последнем слое числа $1, 2, 5$.

а. Какое итоговое распределение вероятностей мы получим, если $T = 10$?

б. А если $T = 1$?

в. А если $T = 0.1$?

г. Какое распределение получится при $T \to 0$?

д. А при $T \to \infty$?

е. Предположим, что объектов на порядок больше. Например, это реплики, которые Алиса может сказать вам в ответ на какую-то фразу. Понятное дело, что вашей фразе будет релевантно какое-то подмножество ответов. Какое значение температуры сэмплирования T смогут сделать реплики Алисы непредсказуемыми? А какие сделают их однотипными?

Регуляризация

Цитата про регуляризацию

Автор цитаты

Упражнение 32 (Маша и покемоны)

Маша измерила вес трёх покемонов, $y_1 = 6$, $y_2 = 6$, $y_3 = 10$. Она хочет спрогнозировать

вес следующего покемона. Модель для веса покемонов у Маши очень простая, $y_i = \beta + \varepsilon_i$, поэтому прогнозирует Маша по формуле $\hat{y}_i = \hat{\beta}$.

Для оценки параметра β Маша использует следующую целевую функцию:

$$\sum (y_i - \hat{\beta})^2 + \lambda \cdot \hat{\beta}^2$$

а) Найдите оптимальное $\hat{\beta}$ при $\lambda = 0$.

б) Найдите оптимальное $\hat{\beta}$ при произвольном λ. Правда ли, что чем больше λ, тем меньше β?

в) Подберите оптимальное λ с помощью кросс-валидации leave one out («выкинь одного»). При такой валидации на первом шаге мы оцениваем модель на всей выборке без первого наблюдения, а на первом тестируем её. На втором шаге мы оцениваем модель на всей выборке без второго наблюдения, а на втором тестируем её. И так далее n раз. Каждое наблюдение является отдельным фолдом.

г) Найдите оптимальное $\hat{\beta}$ при λ_{CV}.

Упражнение 33 (а вот и моя остановочка)

Сделать задачу по связи ранней остановки и регуляризатора. Как в книжке про диплернинг

Упражнение 34 (дропаут)

Маша собирается обучить нейронную сеть для решения задачи регрессии, На вход в неё идёт 12 переменных, в сетке есть 3 скрытых слоя. В пером слое 300 нейронов, во втором 200, в третьем 100.

a) Сколько параметров предстоит оценить Маше? Сколько наблюдений вы бы на её месте использовали?

b) Пусть в каждом слое была отключена половина нейронов. Сколько коэффициентов необходимо оценить?

c) Предположим, что Маша решила после первого слоя добавить в свою сетку Dropout с вероятностью p. Какова вероятность того, что отключится весь слой?

d) Маша добавила Dropout с вероятностью p. после каждого слоя. Какова вероятность того, что один из слоёв отключится и сетка не сможет учиться?

e) Пусть случайная величина N — это число включённых нейронов. Найдите её математическое ожидание и дисперсию. Если Маша хочет проредить сетку на четверть, какое значение p она должна поставить?

f) Пусть случайная величина P — это число параметров в нейросети, которое необходимо оценить. Найдите её математическое ожидание и дисперсию. Почему найденное вами

математическое ожидание выглядит очень логично? Что оно вам напоминает? Обратите внимание на то, что смерть одного из параметров легко может привести к смерти другого.

Добавить вопросиков про дропконнект

Бэкпроп через дропаут

Нормализация по батчам

Чашка хорошего чая восстановит мою нормальность.

Артур из «Автостопом по галактике»

Бэкпроп через батчнорм, смысл батчнорма, нарисовать граф через него, про то что это уродливая процедура

родить задачу из статьи dropout vs batchnorm

Инициализация

цитата об этом

автор

Упражнение 35 (инициализация весов)

а. Маша использует для активации симметричную функцию. Для инициализации весов она хочет использовать распределение

$$w_i \sim U\left[-\frac{1}{\sqrt{n_{in}}}; \frac{1}{\sqrt{n_{in}}}\right].$$

Покажите, что это будет приводить к затуханию дисперсии при переходе от одного слоя к другому.

б. Какими нужно взять параметры равномерного распределения, чтобы дисперсия не затухала?

в. Маша хочет инициализировать веса из нормального распределения. Какими нужно взять параметры, чтобы дисперсия не затухала?

г. Несимметричный случай

Упражнение 36 (ReLU и инициализация весов)

Внутри нейрона в качестве функции активации используется ReLU. На вход идёт 10 признаков. В качестве инициализации для весов используется нормальное распределение, $N(0, 1)$. С какой вероятностью нейрон будет выдавать на выход нулевое наблюдение, если

Предположения на входы? Какое распределение и с какими параметрами надо использовать, чтобы этого не произошло? Сюда же про инициализацию Хе.

Упражнение 37 (сложная задача про инициализацию)

Так чтобы плотность частного надо было искать и все офигели (Воронцов)

Стрельба по ногам

цитата об этом

автор

Упражнение 38 (Проблемы с архитектурой)

Миша принёс Маше несколько разных архитектур. Они выглядят довольно странно. Помогите Маше разобраться, что именно Миша сделал неправильно.

Упражнение 39 (Ещё один backpropagation)

У Маши есть трёхслойная нейросеть:

$$y = f(f(X \cdot W_3) \cdot W_2) \cdot W_1$$

а. Маща использует в качестве функции активации $f(t) = \mathrm{ReLU}(t) = \max(0; t)$, а в качестве функции потерь $L(W_1, W_2) = \frac{1}{2} \cdot (y - \hat{y})^2$.

б.

в.

г.

Для всех пунктов запишите уравнения для прямого и обратного проходов по сетке. Выпишите для всех весов уравнения, по которым будет делаться шаг градиентного спуска.

Листочек 6: свёрточные сети

> Урра! Отлично сработано, ребятки. Давайте завтра не придем? Возьмем отгул на денек? Вы пробовали шаурму? В двух кварталах отсюда делают какую-то шаурму. Не знаю, что это, но мне хочется.
>
> *Тони Старк (Мстители, 2012)*

Упражнение 40 (Свёртка своими руками)

У Маши есть картинка и свёртка, которую она хочет применить к этой картинке.

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
3	1	2	2	3
3	1	2	2	3

Картинка Маши

0	1	2
2	2	0
0	1	2

Свёртка Маши

а. Сделайте свёртку картинки без сдвигов и дополнений. К тому, что получилось применить max pooling и average pooling размера 2×2.

б. Примените к исходной картинке свёртку с параметром сдвига (stride) равным 2.

в. Примените к исходной картинке свёртку с параметром сдвига (stride) равным 3.

г. Примените к исходной картинке свёртку с дополнением нулями (zero padding) и параметром сдвига (stride) равным 0.

Упражнение 41 (Ядра)

У Маши есть куча ядер для свёрток. Догадайтесь какое из них что делает:

0	0	0
0	1	0
0	0	0

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-1	-1
-1	8	-1
-1	-1	-1

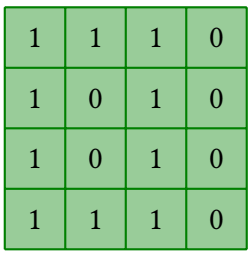
-1	-1	-1
-1	8	-1
-1	-1	-1

Упражнение 42 (Крестики и нолики)

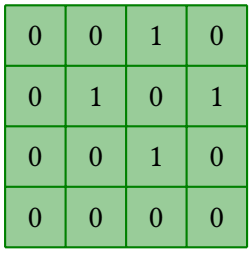
Маша хочет научить компьютер играть в крестики и нолики. На первом шаге ей надо научить алгоритм распознавать есть ли крестик на картинке. Под ноликом понимается любая фигура с дырой в середине. Под крестиком понимается любая фигура из двух пересекающихся линий.

Алгоритм должен быть устроен следующим образом. На первом шаге одна или несколько свёрток проходят по картинке. На втором шаге по результатам свёрток принимается решение. Например, берётся максимальное получившееся число и сравнивается с каким-то порогом. Классификатор крестиков и ноликов должен работать безупречно. Помогите Маше придумать такой классификатор.

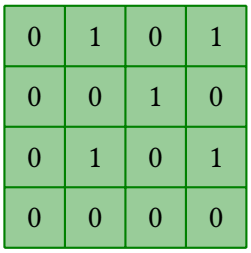
а. В мире Маши на картинках могут быть нарисованы либо крестики либо нолики. Все картинки, подающиеся на вход алгоритма могут быть только размера 4×4. Примеры крестиков и ноликов нарисованы ниже.



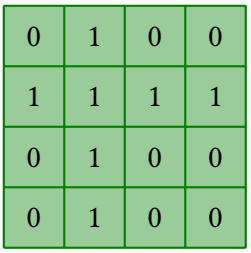
нолик



нолик



крестик



крестик

б. Предположим, что теперь кроме крестиков и ноликов в нашем мире существуют ещё и другие любые картинки. Как можно модернизировать ваш алгоритм, чтобы он по-прежнему стабильно работал с безупречным качеством?

Упражнение 43 (Свёрточный и полносвязный)

Маше рассказали, что свёрточный слой — это полносвязный слой с некоторыми ограничениями. Она хочет разобраться, что это за ограничения. На вход в слой идёт чёрно-белое изображение размера 4×4. Каждый пиксель изображения — отдельная переменная.

а. Нарисуйте с помощью кругляшей и стрелочек полносвязным слой, который обрабатывает картинку. Подпишите все веса. Нарисуйте свёрточный слой в таком же формате. На картинке часть связей исчезнет, а часть весов станет одинаковой.

б. Запишите свёрточный слой с помощью перемножения матриц в виде $H = X \cdot W$. Как выглядит матрица W? Как через свёрточный слой можно сделать шаг обратного распространения ошибки?

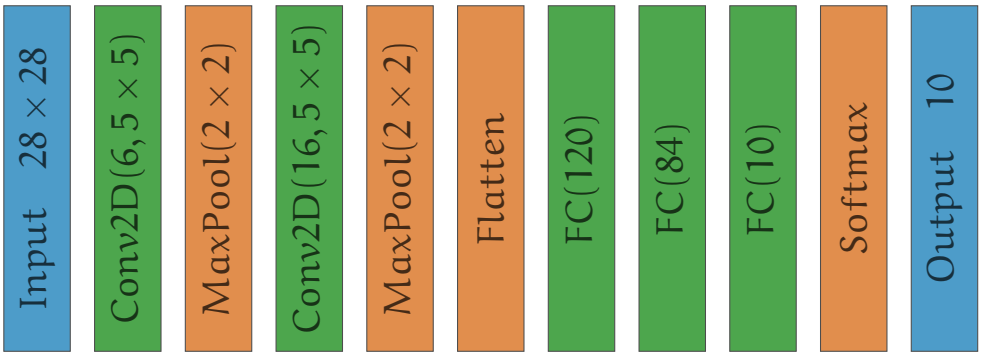
Упражнение 44 (Число параметров)

На вход в нейронную сетку идёт изображение рукописной цифры размера 28×28.

а. Маша вытягивает изображение в длинный вектор и использует полносвязную сетку для классификации изображений. В сетке идёт один полносвязный слой из 1000 нейронов. После идёт слой, который осуществляет классификацию изображения на 10 классов.

Сколько параметров нужно оценить?

б. Маша вместо полносвязной сетки использует свёрточную. На первом шаге делается 6 свёрток размера 5×5. На втором шаге делается max-pooling размера 2×2. На третьем 16 свёрток размера 5×5. На четвёртом max-pooling размера 2×2. На пятом картинка вытягивается в длинный вектор. Далее идут три полносвязных слоя размеров $120, 84, 10$. В конце делается softmax. После каждой свёртки и полносвязного слоя, кроме последнего, в качестве функции активации используется ReLU.



Сколько параметров необходимо будет оценить в такой модели? Какого размера будут выходы из каждого слоя?

в. Маша использует архитектуру из пункта б, но все свёртки делает с дополнением нулями (zero padding). Как изменится число оцениваемых параметров? Какого размера будут выходы из каждого слоя?

г. Маша использует архитектуру из пункта б, но все свёртки делает с дополнением нулями (zero padding) и параметром сдвига (stride) равным 2. ак изменится число оцениваемых параметров? Какого размера будут выходы из каждого слоя?

Упражнение 45 (Поле обзора)

Маша хочет найти котика размера 512×512 пикселей. Для этого она использует свёртки размера 5×5 без дополнения нулями (zero padding) После каждого свёрточного слоя Маша делает max-pooling.

а. Через сколько слоёв поле восприятия Машиной нейросетки впервые охватит котейку?

б. Маша хочет поменять max-pooling на свёртки со сдвигом (stride) так, чтобы котейка находился за такое же число слоёв. Какой размер сдвига ей надо выбрать?

в. Пусть s — величина сдвига, k — размер свёртки, m — размер пулинга, n — номер слоя. Выпишите формулу, по которой можно найти размер поля видимости (receptive field).

Упражнение 46 (Снова число параметров)

Маша собирает разные архитектуры. Помогите ей оценить число параметров для каждой из них.

а. У Маши есть свёрточный слой. На вход в свёрточный слой идёт изображение с C_{in} каналами размера $W \times H$. Маша использует C_{out} фильтров размера $W_k \cdot H_k$. Сколько параметров ей предстоит оценить?

б.

в.

Сюда вариант с подсчётом числа параметров в сепарабельной свёртке

Упражнение 47 (Скользящее среднее)

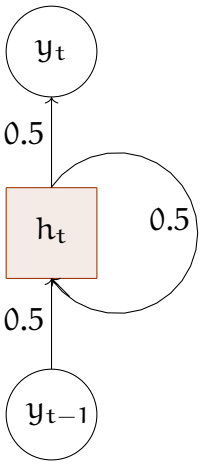
Скользящее среднее — это свёртка, которая работает для вектора. Опишите как именно она работает. Какой физический смысл стоит за размером такой свёртки и дополнением нулями?

Листочек 7: рекуррентные сети

> А сегодня в завтрашний день, не все могут смотреть. Вернее смотреть могут не только лишь все, не каждый может это делать.
>
> *Рекуррентная сеть глубины один*

Упражнение 48 (Туда и обратно)

Маша хочет сделать шаг обратного распространения ошибки через рекуррентную ячейку для последовательности $y_0 = 0, y_1 = 1, y_2 = -1, y_3 = 2$. Скрытое состояние инициализировано как $h_0 = 0$. Все веса инициализированы как 0.5. Во всех уравнениях, описывающих ячейку нет констант. В качестве функций активаций Маша использует ReLU. В качестве функции потерь Маша использует MSE.



а. Сделайте прямой шаг через ячейку. Для каждого элемента последовательности постройте прогноз. Посчитайте значение ошибки.

б. Сделайте обратный шаг распространения ошибки. Посчитайте для каждого из весов градиенты и обновите значения весов.

Упражнение 49 (Число параметров)

У Маши есть очень длинный временной ряд. Она хочет обучить несколько нейросетей предсказывать его дальнейшее значение. В своих моделях Маша нигде не использует константы.

а. Маша выделяет окно длины 100. Оно движется по последовательности. Для каждого окна Маша предсказывает следующее значение в ряду. В сетку подаются наблюдения с 1—го по 100—е. Прогнозируется 101—ое наблюдение. Затем на вход подаются наблюдения со 2—го по 100—е. Прогнозируется 102—ое наблюдение. И так далее до конца последовательности.

   На первом слое используется 20 нейронов. На втором слое используется один нейрон. Сколько параметров нужно оценить?

б. Маша использует одну простую RNN-ячейку. Сколько параметров ей необходимо оценить?

в. Маша хочет предсказывать значение y_t по трём последовательностям y_{t-1}, y_{t-2} и y_{t-3}. На первом слое сети Маша использует два рекуррентных нейрона. На втором слое она использует один рекуррентный нейрон. Матрица какого размера идёт на вход в первый слой? Матрица какого размера передаётся во второй слой? Какое число параметров необходимо оценить Маше?

г. Мы находимся в условиях прошлого пункта, но используетм LSTM-ячейки с забыванием. Сколько параметров надо оценить?

д. Мы находимся в условиях прошлого пугкта, но используем GRU-ячейки. Сколько параметров надо оценить?

Упражнение 50 (Из картинки в формулу)

У Маши есть два рекуррентных нейрона. Помогите ей изобразить их в виде вычислительных графов.

Однонаправленный:

$$\begin{aligned} h_t &= f_h(b_h + W \cdot h_{t-1} + V \cdot x_t) \\ y_t &= f_y(b_y + U \cdot h_t) \end{aligned}$$

Двунаправленный:

$$\begin{aligned} h_t &= f_h(b_h + W \cdot h_{t-1} + V \cdot x_t) \\ s_t &= f_s(b_s + W' \cdot s_{t+1} + V' \cdot x_t) \\ o_t &= b_y + U \cdot h_t + U' \cdot s_t \\ y_t &= f_y(o_t) \end{aligned}$$

Упражнение 51 (Замочные скважины)

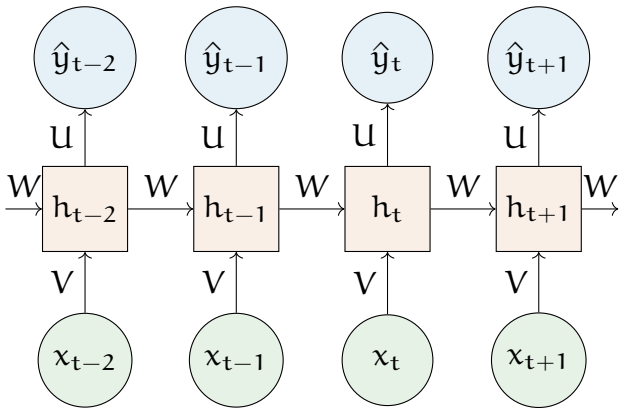
В 2000 году Шмидхубер и Герс предложили модификацию LSTM с замочными скважинами. Она описывается следующей системой из уравнений

$$\begin{aligned} c'_t &= \phi_c(W_c x_t + V_c h_{t-1} + b_c) \\ i_t &= \phi_i(W_i x_t + V_i h_{t-1} + U_i c_{t-1} + b_i) \\ f_t &= \phi_f(W_f x_t + V_f h_{t-1} + U_f c_{t-1} + b_f) \\ o_t &= \phi_o(W_o x_t + V_o h_{t-1} + U_o c_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot c'_t \\ h_t &= o_t \odot \phi_h(c_t) \end{aligned}$$

Изобразите эту ячейку в виде вычислительного графа. Объясните, чем именно она отличается от базовой модификации LSTM. Какой в этом смысл?

Упражнение 52 (Лишние части)

Выпишите уравнения, описывающие LSTM-ячейку с забыванием и GRU-ячейку. Какие последовательности и веса нужно занулить, чтобы эти ячейки превратились в простую RNN-ячейку?


$\hat{y}_{t-2}$
$\hat{y}_{t-1}$
$\hat{y}_t$
$\hat{y}_{t+1}$
U
W
h_{t-2}
h_{t-1}
h_t
h_{t+1}
V
x_{t-2}
x_{t-1}
x_t
x_{t+1}

какие-нибудь упражнения про w2v

упражнение про разные модные виды ячеек типа резнетов и тп

задачи про атеншны и тп