Тятя! Тятя! Нейросети заменили продавца!

Ульянкин Ппилиф

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

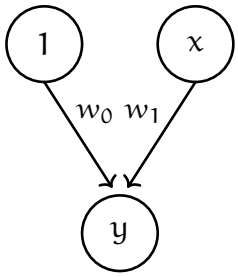
1 Всего лишь функция

Ты всего лишь машина, только имитация жизни. Робот сочинит симфонию? Робот превратит кусок холста в шедевр искусства?

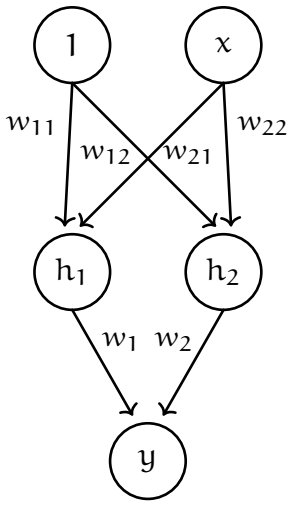
Из фильма «Я, робот» (2004)

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:

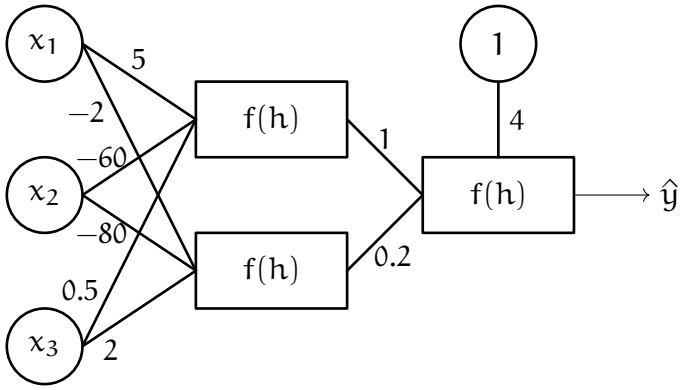
$$y = w_1 \cdot f(w_{11} + w_{21} \cdot x) + w_2 \cdot f(w_{12} + w_{22} \cdot x).$$

~~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:

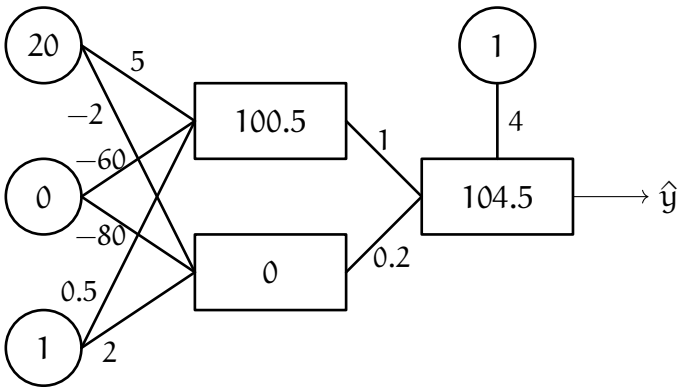
$$h_2 = \max(0, -2 \cdot 20 + (-80) \cdot 0 + 2 \cdot 1) = \max(0, -38) = 0$$

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:

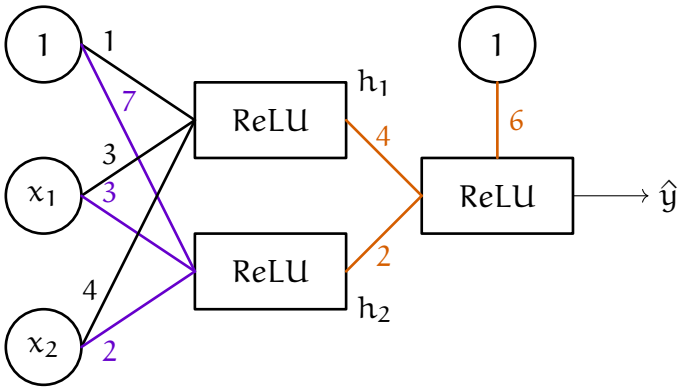
Начнём рисовать картинку с конца. На выход выплёвывается либо 0, либо комбинация из двух входов:

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

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:

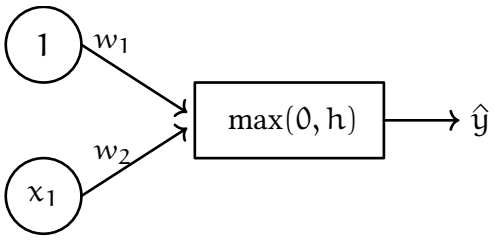


Упражнение 4 (армия регрессий)

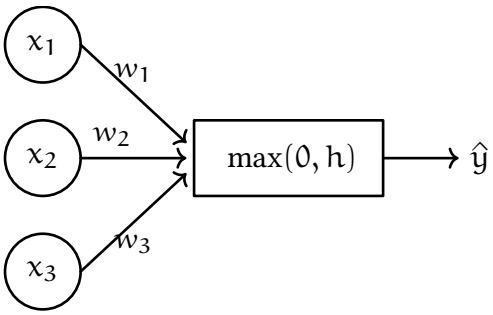
Парни очень любят Машу,[7] а Маша с недавних пор любит собирать персептроны и думать по вечерам об их весах и функциях активации. Сегодня она решила разобрать свои залежи из персептронов и как следует упорядочить их.

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



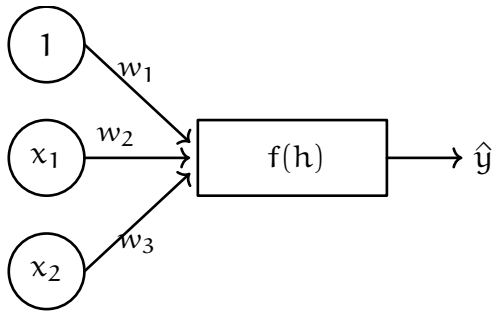
б. В тумбочке, среди носков, Маша нашла персептрон, с картинки 2, Маша хочет подобрать такие веса w_i, чтобы персептрон превращал x из таблички в соответствующие y:

x_1	x_2	x_3	y
1	1	2	0.5
1	−1	1	0

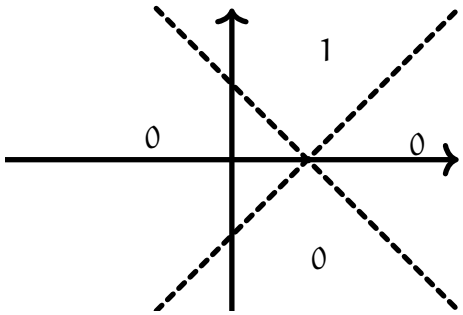
[7] когда у тебя есть лёрнинг, они так и лезут

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:

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

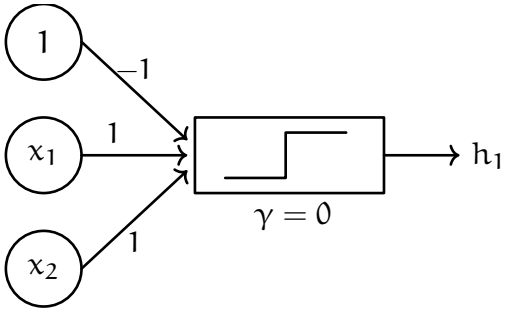
Тут решений может быть довольно много. Одно из них — это занулить w_1 и w_3 в первом уравнении, а w_2 поставить 0.5. Тогда во втором уравнении мы сразу же будем оказываться в отрицательной области и ReLU заботливо будет отдавать нам 0.

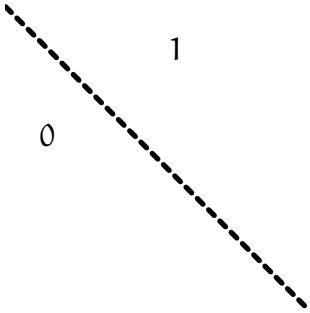
в. Единичная ступенька выглядит как

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

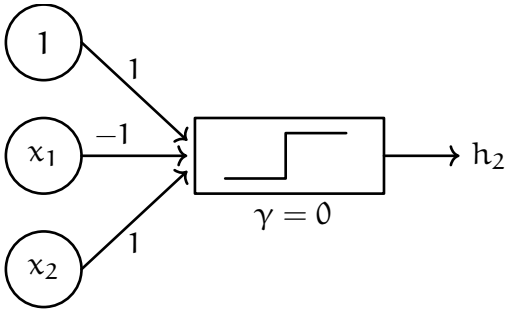
Один нейрон — это одна линия, проведённая на плоскости. Эта линия отделяет один

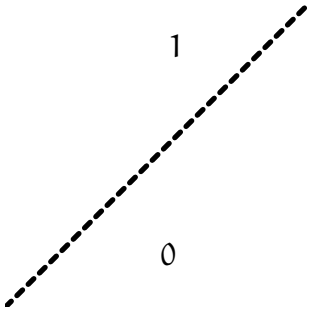
класс от другого. Например, линию $x_1 + x_2 - 1 = 0$ мог бы описать нейрон





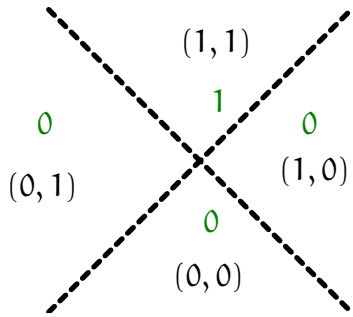
Порог γ для кусочной функции в каком-то смысле дублирует константу. Они взаимосвязаны. Будем всегда брать его нулевым. Видим, что если мы получили комбинацию x_1, x_2 и 1, большую, чем ноль, мы оказались справа от прямой. Если хочется поменять метки 0 и 1 местами, можно умножить все коэффициенты на -1. **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.

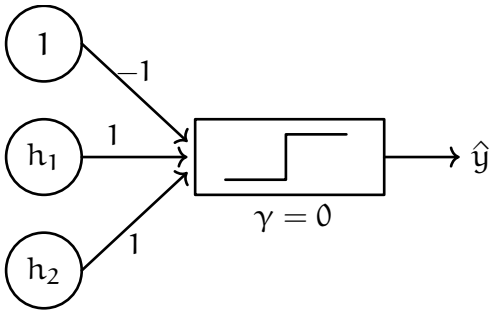




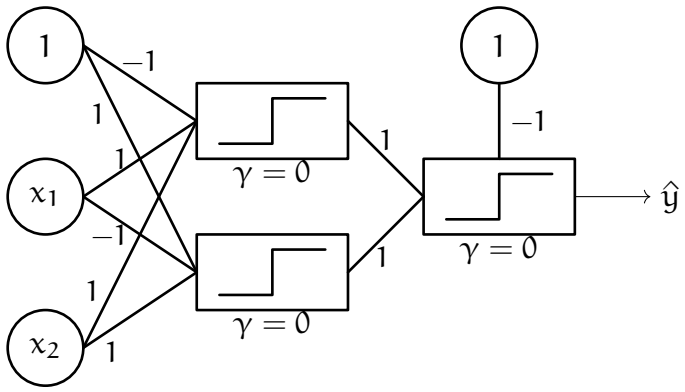
Итак, первый персептрон выбрал нам позицию относительно первой прямой, второй относительно второй. Остаётся только объединить эти результаты. Нейрон для скрепки должен реализовать для нас логическую функцию, которую задаёт табличка ниже. Там же нарисованы примеры весов, которые могли бы объединить выхлоп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

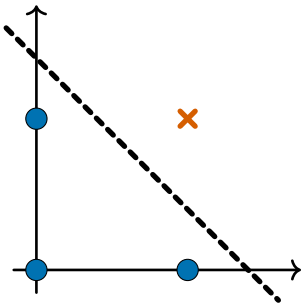
x_1	x_2	x_1 XoR x_2
1	1	0
1	0	1
0	1	1
0	0	0

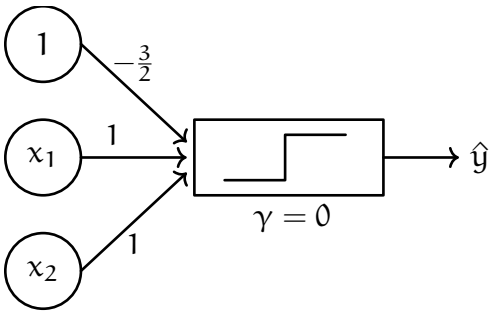
Первые два столбика идут на вход, третий получается на выходе. Первая операция — логическое "и вторая — "или". Операция из третьей таблицы называется "исключающим или (XoR). Если внимательно приглядеться, то можно заметить, что XoR — это то же самое что и $[x_1 \neq x_2]$[8].

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.

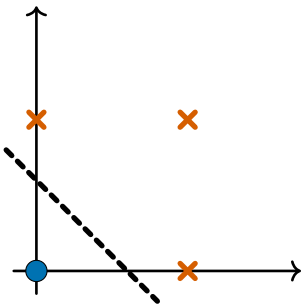


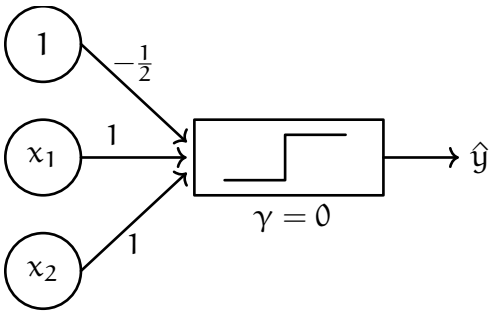


Если нарисовать все наши четыре точки на плоскости, становится ясно, что мы хотим отделить точку $(1, 1)$ от всех остальных. Сделать это можно большим числом способов. Например, в нейроне выше задана линия $x_2 = 1.5 - x_1$. Подойдёт и любая другая линия, отделяющая $(1, 1)$ от остальных точек. Пропустим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

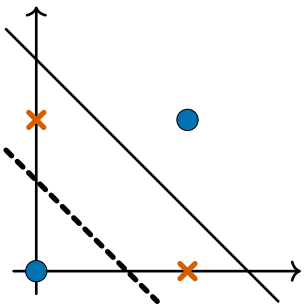
$$[-1.5 + 1 + 1 > 0] = [0.5 > 0] = 1$$
$$[-1.5 + 0 + 0 > 0] = [-1.5 > 0] = 0$$
$$[-1.5 + 0 + 1 > 0] = [-0.5 > 0] = 0$$
$$[-1.5 + 1 + 0 > 0] = [-0.5 > 0] = 0$$

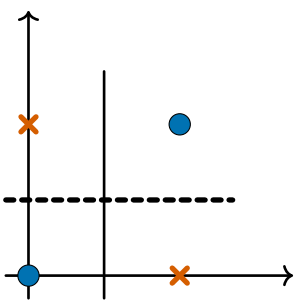
С объединением та же ситуация, только на этот раз линия должна пройти чуть ниже. Подойдёт $x_2 = 0.5 - x_1$.





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)

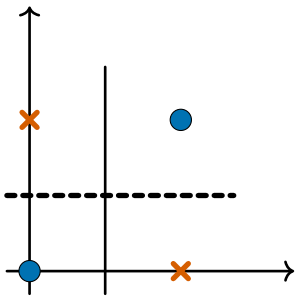
Маша заметила, что на XoR ушло очень много персептронов. Она поняла, что первые два персептрона пытаются сварить для третьего нелинейные признаки, которых нейросетке не хватает. Она решила самостоятельно добавить персептрону вход $x_3 = x_1 \cdot x_2$ и реализовать XoR одним персептроном. Можно ли это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

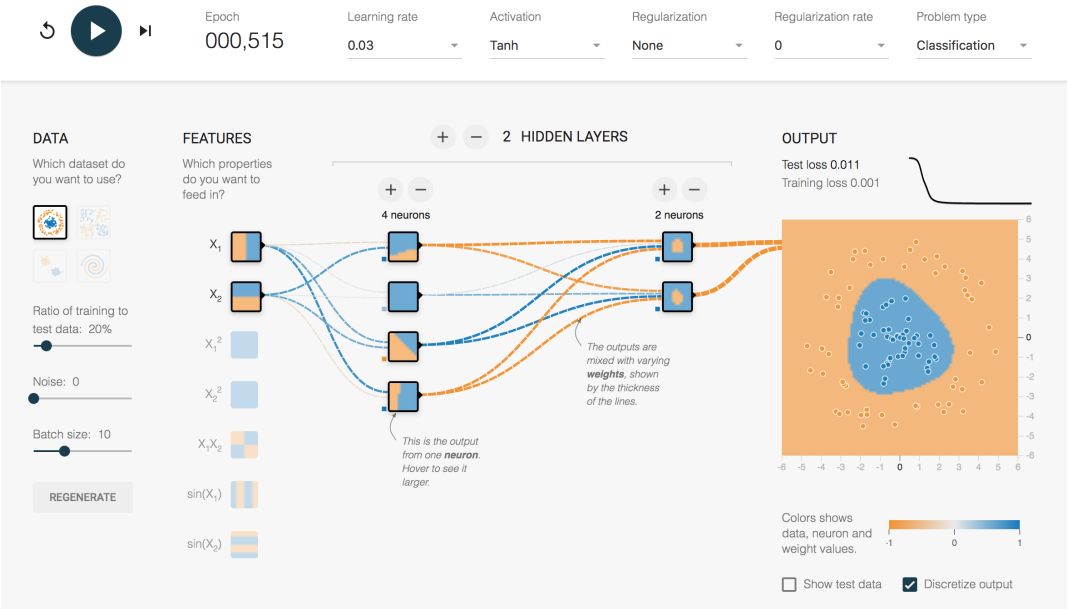
$$\begin{aligned}
&x_1 + x_2 - 2x_1x_2 - 0.5 = 0 \\
&2x_1 + 2x_2 - 4x_1x_2 - 1 = 0 \\
&2x_1(1 - 2x_2) + 2x_2 - 1 = 0 \\
&(1 - 2x_2) \cdot (2x_1 - 1) = 0
\end{aligned}$$

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.

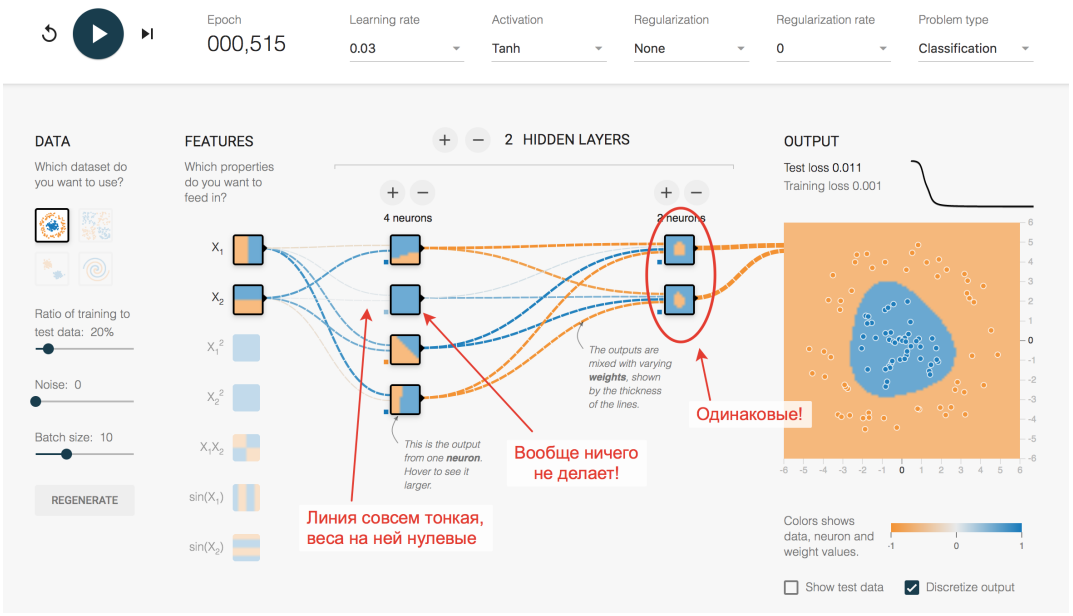


Маша заметила, что с её архитектурой что-то не так. Какие у неё проблемы?

Решение:

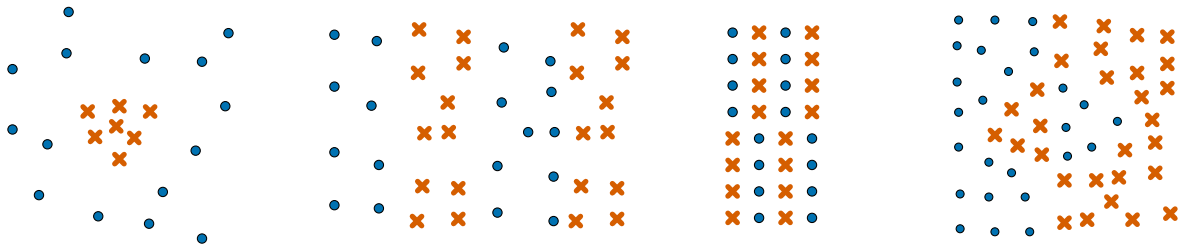
Нейросетка Маши оказалась избыточной. Во-первых, можно увидеть, что на первом слое есть нейрон, который вообще ничего не делает. Связи, которые идут к нему от входов настолько тусклые (коэффициенты при них равны нулю), что их даже не видно на картинке. От этого нейрона смело можно избавиться и сделать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

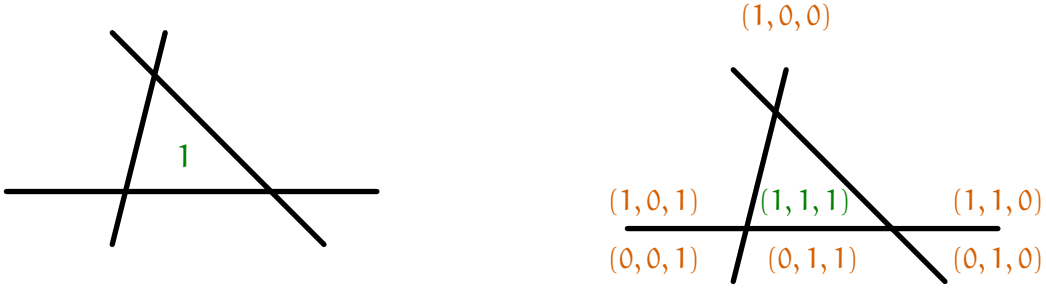
Шестилетняя сестрёнка ворвалась в квартиру Маши и разрисовала ей все обои:



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

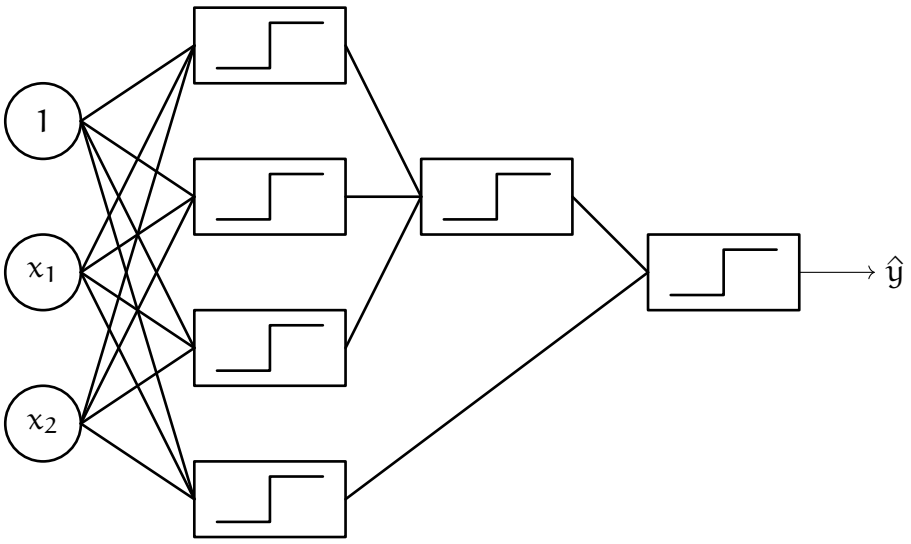
$$\begin{cases} f(1 \cdot w_1 + 1 \cdot w_2 + 1 \cdot w_3) = 1 \\ f(1 \cdot w_1 + 0 \cdot w_2 + 0 \cdot w_3) = 0 \\ f(1 \cdot w_1 + 1 \cdot w_2 + 01 \cdot w_3) = 0 \\ f(0 \cdot w_1 + 0 \cdot w_2 + 0 \cdot w_3) = 0 \end{cases}.$$

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.

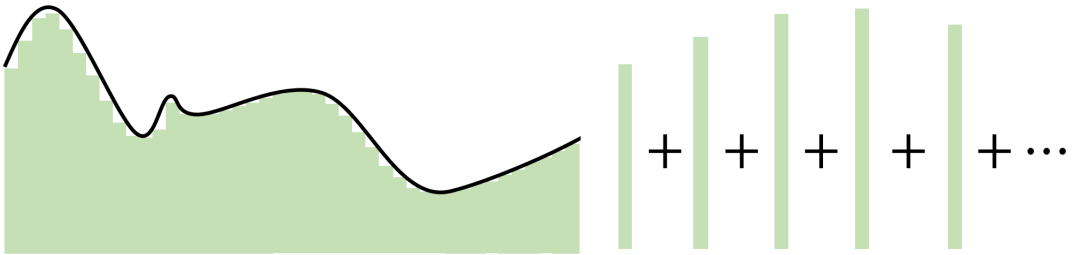
Упражнение 9 (универсальный регрессор)

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:

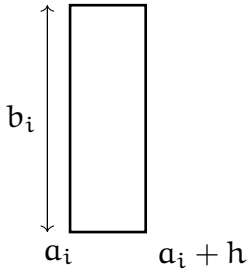
Не нужно воспринимать эту задачку как строгое доказательство. Скорее, это показательство. Мы хотим приблизить функцию $f(x)$ с какой-то точностью. Будем делать это с помощью кусочно-линейных ступенек. Чем выше точность, тем больше будем рисовать ступенек.



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 $a_{i+1} = a_i + h$.



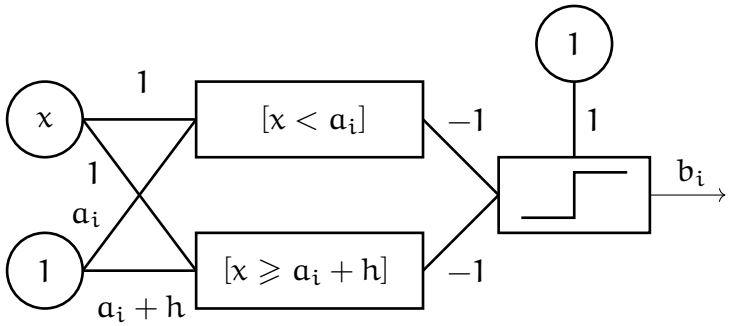
Если x, для которого мы ищем $f(x)$ попадает в полуинтервал, на котором задана наша ступенька, мы будем приближать $f(x)$ этой ступенькой. Ступенька состоит из двух линий.

[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

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

2 Что выплёвывает нейросеть

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

Пацанский паблик категории Б

Упражнение 1 (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.

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 2 (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 3 (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 результат: $1, -2, 0$ 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 4 (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.

3 Пятьдесят оттенков градиентного спуска

Повторять до сходимости — это как жарить до готовности

Неизвестный студент Вышки

Упражнение 1 (50 оттенков спуска)

Маша Нестерова, хозяйка машин лёрнинга[10], 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 два шага градиентного спуска. В качестве стартовой точки используйте $w_0 = 0$. В качестве скорости обучения возьмите $\eta = 0.1$.

в. Сделайте два 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 $\beta_1 = \beta_2 = 0.9, \eta = 0.1$

Упражнение 2 (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:

[10] Лёрнинг ей папа подарил

Сначала нам надо найти $\text{logloss}'_{\beta}$. 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:

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

4 Алгоритм обратного распространения ошибки

Что происходит, когда мы суём пальцы в розетку? Нас бьёт током! Мы делаем ошибку, и она распространяется по нашему телу.

Твоя мама

Упражнение 1 (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

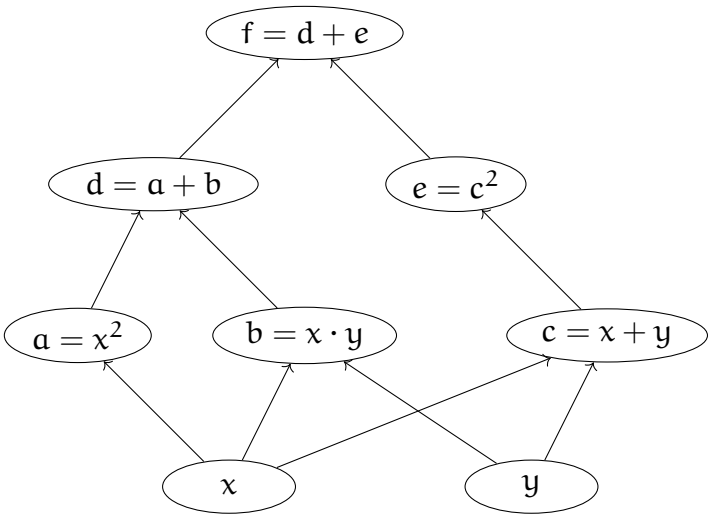
$$f(x, y) = x^2 + xy + (x + y)^2.$$

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.

граф вычислений. В вершинах графа она будет записывать результаты вычислений. Каждое ребро будет обозначать элементарную операцию: плюс или умножить[11].

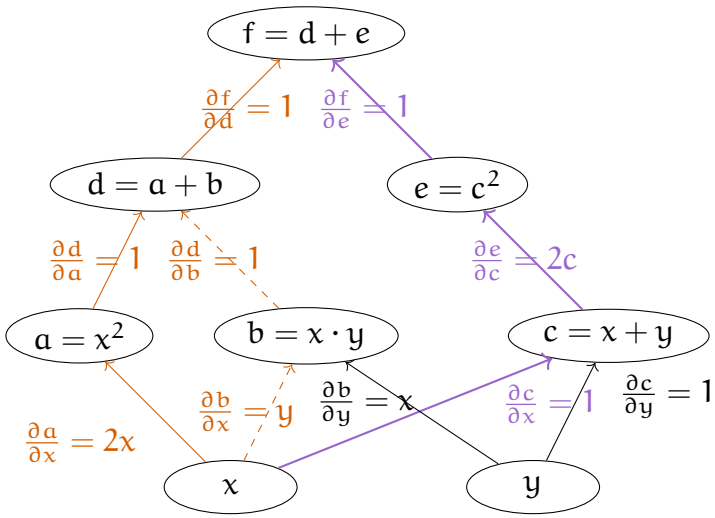
Решение:

Нарисуем граф вычислений.



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

[11] 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

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

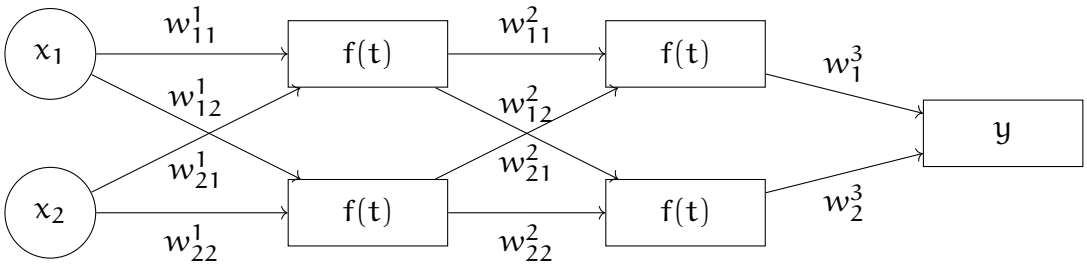
По аналогии найдём производную по траекториям из y в f:

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

В тексте ниже огромное число ошибок, я их со временем исправлю!

Упражнение 2 (придумываем backpropagation)

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.



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

б. Предположим, что Маша решает задачу регрессии. Она прогоняет через нейросетку одно наблюдение. На выходе она вычисляет знчение функции потерь $L(W_1, W_2, W_3) = \frac{1}{2} \cdot (y - \hat{y})^2$ — функция потерь, где W_k — веса k—го слоя. 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 часть 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:

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_{1i}^2 и o_{2i}^2 результат вычисления предыдущих слоёв

$$\hat{y}_i = w_1^3 \cdot f(w_{12}^1 \cdot o_{i1}^1 + w_{22}^1 \cdot o_{i2}^1) + w_2^3 \cdot f(w_{12}^1 \cdot o_{i1}^1 + w_{22}^1 \cdot o_{i2}^1).$$

Подставим результат вычисления первого слоя

$$\hat{y}_i = w_1^3 \cdot f(w_{12}^1 \cdot f(w_{11}^1 \cdot x_{i1} + w_{21}^1 \cdot x_{i2}) + w_{22}^1 \cdot f(w_{12}^1 \cdot x_{i1} + w_{22}^1 \cdot x_{i2})) + \\ + w_2^3 \cdot f(w_{12}^1 \cdot f(w_{11}^1 \cdot x_{i1} + w_{21}^1 \cdot x_{i2}) + w_{22}^1 \cdot f(w_{12}^1 \cdot x_{i1} + w_{22}^1 \cdot x_{i2})).$$

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_{i1}^1, h_{i2}^1) мы делаем матричное умножение. Строчку

(x_{i1}, x_{i2}) мы умножаем на матрицу весов W_1

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

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, выписываем функцию потерь и аккуратно берём все производные

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

Бляяяя

Здесь y это скаляр, а x это строчка. Всего нам надо взять три матричные производные. Функция $h = x \cdot A$ бьёт из векторов $[1 \times 2]$ в вектора $[1 \times 2]$. Значит

С транспонированием около матриц надо просто смириться. Мы используем правило взятия матричных производных, которое состоит в том, что для производная $\frac{\partial f}{\partial X} = A^T$. Обычно мы работаем с векторами-столбцами и это нужно, чтобы с размерностями матриц не возникало проблем. Если хочется разобраться подробнее, прочитайте необязательную часть про матричное дифференцирование.

Просто делаем это по правилу взятия производной сложной функции. Как в школе.

$$\underset{[2\times1]}{\frac{\partial L}{\partial W_3}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_3} = \underset{[1\times1]}{(y - \hat{y})} \cdot f(f(x \cdot W_1) \cdot W_2)$$
$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_2} = (y - \hat{y}) \cdot W_3 \cdot f'(f(x \cdot W_1) \cdot W_2) \cdot f(x \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_1} = (y - \hat{y}) \cdot W_3 \cdot f'(f(x \cdot W_1) \cdot W_2) \cdot W_2 \cdot f'(x \cdot W_1) \cdot X$$

Проследим за размером всех производных.

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

$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_3} = \boxed{(y - \hat{y})} \cdot f(f(x \cdot W_1) \cdot W_2)$$
$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_2} = \boxed{(y - \hat{y})} \cdot \boxed{W_3 \cdot f'(f(x \cdot W_1) \cdot W_2)} \cdot f(x \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_1} = \boxed{(y - \hat{y})} \cdot \boxed{W_3 \cdot f'(f(x \cdot W_1) \cdot W_2)} \cdot W_2 \cdot f'(x \cdot W_1) \cdot X$$

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

$$W_3^t = W_3^{t-1} - \eta \cdot \frac{\partial L}{\partial W_3}(W_3^{t-1})$$
$$W_2^t = W_2^{t-1} - \eta \cdot \frac{\partial L}{\partial W_2}(W_2^{t-1})$$
$$W_1^t = W_1^{t-1} - \eta \cdot \frac{\partial L}{\partial W_1}(W_1^{t-1}).$$

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.

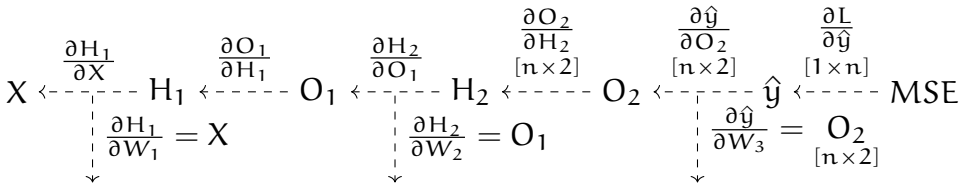
Делай три, выпишем алгоритм обратного распространения ошибки в виде красивой схемки. Сначала мы делаем прямой проход по нейросети (forward pass):

$$\underset{[n\times2]}{X} \xrightarrow{\substack{W_1 \\ [2\times2]}} \underset{[n\times2]}{H_1} \xrightarrow{f} \underset{[n\times2]}{O_1} \xrightarrow{\substack{W_2 \\ [2\times2]}} \underset{[n\times2]}{H_2} \xrightarrow{f} \underset{[n\times2]}{O_2} \xrightarrow{\substack{W_3 \\ [2\times1]}} \underset{[n\times1]}{\hat{y}} \longrightarrow L(y, \hat{y})$$

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, $\frac{\partial H_2}{\partial W_2} = O_2^T$. 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. Везде под производными подпишем размерности соотвествующих матриц



Осталось только аккуратно записать обратный ход алгоритма. Заведём для накопленного значения производной переменную d. На первом шаге нам надо найти $\frac{\partial L}{\partial W_3}$. Сделаем это в два хода

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

Вектор d будет размера $[n \times 1]$, матрица O_2^T будет размера

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

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_2} \cdot \frac{\partial O_2}{\partial H_2} \cdot \frac{\partial H_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial H_1} \cdot \frac{\partial H_1}{\partial W_1} = d \cdot \frac{\partial H_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial H_1} \cdot \frac{\partial H_1}{\partial W_1}.$$

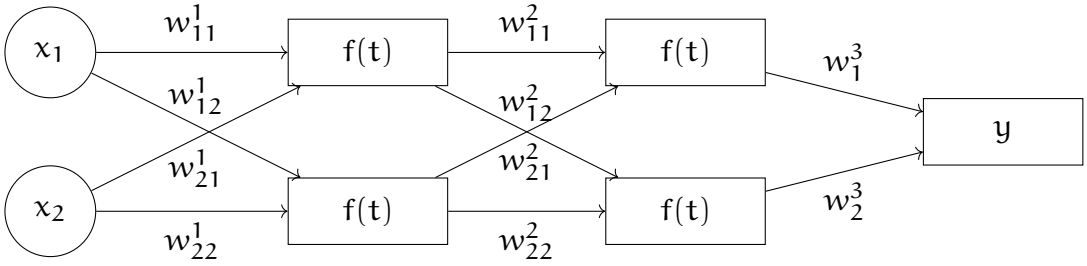
Снова делаем это в два шага

$$d = d \cdot \frac{\partial H_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial H_1}$$
$$\frac{\partial L}{\partial W_1} = d \cdot X.$$

Если бы нейросетка была бы глубже, мы смогли бы переиспользовать d на следующих слоях. Каждую производную благодаря такому постепенному подходу мы нашли ровно один раз. Это и есть алгоритм обратного распространения ошибки.

Упражнение 3 (Backpropagation без константы)

У Маши есть нейросеть с картинки ниже. Она использует функцию потерь $L(W_1, W_2, W_3) = \frac{1}{2} \cdot (y - \hat{y})^2$. 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

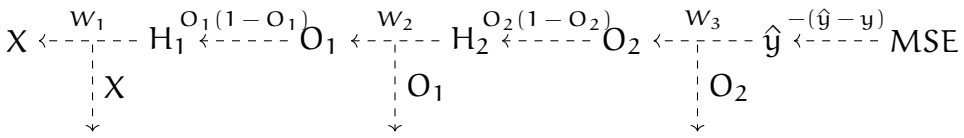
Сделайте руками два шага алгоритма обратного распространения ошибки. Пусть скорость обучения $\eta = 1$. Стохастический градиентный спуск решил, что сначала для шага будет использоваться второе наблюдение, а затем первое.

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:

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

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

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

Начинаем проворачивать алгоритм. Делаем прямое распространение для второго наблюдения, напомним, что матрицы весов инициализированы нулями:

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

Делаем обратный проход. **Шаг 1**:

$$d = -(\hat{y} - y) = -1$$
$$\frac{\partial MSE}{\partial W_3} = d \cdot O_2 = -1 \cdot (0.5, 0.5) = (-0.5, -0.5)$$

Шаг 2:

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

Шаг 3:

$$d = d \cdot W_2 \cdot O_1 \cdot (1 - O_1)$$
$$\frac{\partial MSE}{\partial W_1} = d \cdot X$$

5 Всего лишь кубики LEGO

5.1 Функции активации

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

Код активации Зимнего Солдата

Упражнение 1 (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 2 (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 3 (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 4 (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?

5.2 Регуляризация

Цитата про переобучение

Автор цитаты

Упражнение 5 (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 6 (а вот и моя остановочка)

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

Упражнение 7 (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

5.3 Нормализация по батчам

Чашка хорошего чая восстановит мою нормальность.

Артур из «Автостопом по галактике»

Бэкпроп через батчнорм, смысл батчнорма

родить задачу из статьи dropout vs batchnorm

5.4 Инициализация

цитата об этом

автор

Упражнение 8 (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 9 (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 Xe.

задача про инициализацию от Воронцова

5.5 Стрельба по ногам

Упражнение 10 (Проблемы с архитектурой)

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

а. Решается задача регрессии, предсказываются цены на недвижимость.

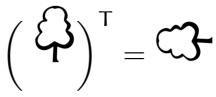
б. Решается задача классификация картинок на 10 классов. Исходный размер картинок 28×28.

в. Решается задача классификация картинок на 10 классов. Исходный размер картинок 100×100.

6 Свёрточные сетки

7 Рекуррентные сетки

8 Матричное дифФфФфФфференцирование



«Джек и бобовый стебель» (1890)

Эта часть виньетки необязательна для изучения. В ней мы подробно поговорим про матричные производные. С их помощью удобно заниматься оптимизацией, в том числе бэкпропом[12].

Упражнение 1

Найдите следующие производные:

а. $f(x) = x^2$, где x скаляр

б. $f(x) = a^T x$, где a и x векторы размера $1 \times n$

в. $f(x) = x^T A x$, где x вектор размера $1 \times n$, A матрица размера $n \times n$

г. $f(x) = \ln(x^T A x)$, где x вектор размера $1 \times n$, A матрица размера $n \times n$

д. $f(x) = a^T X A X a$, где x вектор размера $1 \times n$, A матрица размера $n \times n$

е. $f(x) = x x^T x$, где x вектор размера $1 \times n$

Решение:

Зачем нам нужно научиться искать матричные производные? **Машинное обучение — это сплошная оптимизация.** В нём мы постоянно вынуждены искать минимум какой-нибудь штрафной функции. Матричные производные довольно сильно в этом помогают[13].

Когда мы работаем с одномерными функциями, для поиска любых производных нам хватает небольшой таблицы со стандартными случаями и пары правил. Для случая матриц все эти правила можно обобщить, а таблицы дополнить специфическими функциями вроде определителя. **Удобнее всего оказывается работать в терминах «дифференциала» — с ним можно не задумываться о промежуточных размерностях, а просто применять стандартные правила.**

Мы будем работать в этом конспекте со скалярами, векторами и матрицами. Нас будет интересовать, что именно мы дифференцируем, по чему мы дифференцируем и что получается в итоге.

Строчными буквами мы будем обозначать векторы-столбцы и константы. Заглавными буквами мы будем обозначать матрицы. Производная столбца — это столбец. Производная по столбцу — это столбец.

---

[12] Часть задач взята из прототипа задачника по ML Бориса Демешева, часть из конспектов по ML Жени Соколова

[13] В тексте про матричные производные я опирался на этот конспект. Для того, чтобы познакомиться с темой более широко, имеет смысл прочитать его.

$$x = \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix} \qquad X = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \square & \ddots & \square \\ x_{n1} & \dots & x_{nn} \end{pmatrix}.$$

Мы рассмотрим постепенно много разных входов и выходов, и получим таблицу из канонических случаев. По строчкам будем откладывать то, откуда бьёт функция, то есть входы. По столбцам будем откладывать то, куда бьёт функция, то есть выходы. Для ситуаций обозначенных прочерками обобщения получить не выйдет.

	скаляр	вектор	матрица
скаляр	$f'(x)\,dx$	$\mathfrak{J} * dx$	—
вектор	$\nabla f^T\,dx$	$\mathfrak{J}\,dx$	—
матрица	$\mathrm{tr}(\nabla f^T\,dX)$	—	—

Символом $\nabla^T f$ обозначается градиент (вектор из производных). Символом $\mathfrak{J}$ обозначена матрица Якоби. Символом H мы будем обозначать матрицу Гессе из вторых производных.

a. Начнём с уже известного нам случая: функция бьёт из скаляров в скаляры

$$f(x) : R \to R.$$

Примером такой функции может быть $f(x) = x^2$. Мы знаем, что по таблице производных $f'(x) = 2x$. Также мы знаем, что **дифференциал** — это линейная часть приращения функции, а **производная** — это предел отношения приращения функции к приращению аргумента при приращении аргумента стремящемся к нулю.

Грубо говоря, дифференциал помогает представить приращение функции в линейном виде

$$df(x) = f'(x)\,dx.$$

Если мы находимся в какой-то точке x_0 и делаем из неё небольшое приращение dx, то наша функция изменится примерно на $df(x)$. **Оказывается, что именно в терминах дифференциалов удобно работать с матричными производными.**

Свойства матричных дифференциалов очень похожи на свойства обычных. Надо только не забыть, что мы работаем с матрицами.

$$d(XY) = dX\,Y + X\,dY, \quad dX\,Y \neq Y\,dX$$
$$d(\alpha X + \beta Y) = \alpha\,dX + \beta\,dY$$
$$d(X^T) = (dX)^T$$
$$dA = 0, \quad A - \text{матрица из констант}$$

Чтобы доказать все эти свойства достаточно просто аккуратно расписать их. Кроме этих правил нам понадобится пара трюков по работе со скалярами. Если s — скаляр размера 1×1, тогда $s^T = s$ и $\mathrm{tr}(s) = s$, где tr — операция взятия следа матрицы.

С помощью этих преобразований мы будем приводить дифференциалы к каноническому виду и вытаскивать из них производные.

б. Рассмотрим вторую ситуацию из таблицы, функция бьёт из векторов в скаляры. Это обычная функция от нескольких аргументов

$$f(x) : \mathbb{R}^n \to \mathbb{R}.$$

Такие производные мы брать также умеем. Если мы хотим найти производную функции $f(x_1, x_2, \dots, x_n)$, нам надо взять производную по каждому аргументу и записать их все в виде вектора. Такой вектор **называют градиентом**

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Если умножить градиент на вектор приращений, у нас получится дифференциал

$$df(x) = \nabla f^T dx = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n} \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \\ \dots \\ dx_n \end{pmatrix} = \frac{\partial f}{\partial x_1} \cdot dx_1 + \frac{\partial f}{\partial x_2} \cdot dx_2 + \dots + \frac{\partial f}{\partial x_n} \cdot dx_n.$$

При маленьком изменении x_i на dx_i функция будет при прочих равных меняться пропорционально соответствующей частной производной. Посмотрим на конкретный пример, **скалярное произведение**. Можно расписать умножение одного вектора на другой в виде привычной нам формулы

$$\underset{[1\times 1]}{f(x)} = \underset{[1\times n]}{a^T} \cdot \underset{[n\times 1]}{x} = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n.$$

Из неё чётко видно, что $\frac{\partial f}{\partial x_i} = a_i$. Увидев это мы можем выписать градиент функции

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} = a,$$

теперь можно записать дифференциал

$$df = a^T\,dx = \frac{\partial f}{\partial x_1} \cdot dx_1 + \frac{\partial f}{\partial x_2} \cdot dx_2 + \dots + \frac{\partial f}{\partial x_n} \cdot dx_n = a_1 \cdot dx_1 + a_2 \cdot dx_2 + \dots + a_n \cdot dx_n.$$

В то же самое время можно было бы просто воспользоваться правилами нахождения матричных дифференциалов

$$df = da^T x = a^T\,dx = \nabla f^T\,dx,$$

откуда $\nabla f = a$. Производная найдена. При таком подходе нам не надо анализировать каждую частную производную по отдельности. Мы находим одним умелым движением руки сразу же все производные. Давайте немного усложним задачу и увидим его мощь во всей красе.

в. Функция по-прежнему бьёт из векторов в скаляры. Попробуем перемножить все матрицы и расписать её в явном виде по аналогии со скалярным произведением

$$\underset{[1\times 1]}{f(x)} = \underset{[1\times n]}{x^T} \cdot \underset{[n\times n]}{A} \cdot \underset{[n\times 1]}{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j.$$

Если продолжить в том же духе, мы сможем найти все частные производные, а потом назад вернём их в матрицу. Единственное, что смущает — **мы делаем что-то неестественное.** Всё было записано в красивом компактном матричном виде, а мы это испортили. А что, если множителей будет больше? Тогда суммы станут совсем громоздкими, и мы легко запутаемся.

При этом, если воспользоваться тут правилами работы с матричными дифференциалами, мы легко получим красивый результат

$$df = dx^T Ax = d(x^T)Ax + x^T\,d(Ax) = d(x^T)Ax + x^T \underset{dA=0}{d(A)x} + x^T A\,d(x).$$

Заметим, что $d(x^T)Ax$ это скаляр. Мы перемножаем матрицы с размерностями $1\times n$, $n\times n$ и $n \times 1$. В результате получается размерность 1×1. Мы можем смело транспонировать скаляр, когда нам это надо. Эта операция никак не повлияет на результат

$$df = d(x^T)Ax + x^T A\,d(x) = x^T A^T\,dx + x^T A\,dx = x^T(A^T + A)\,dx.$$

Мы нашли матричный дифференциал и свели его к каноничной форме

$$df = \nabla^T f \, dx = x^T (A^T + A) \, dx$$

Получается, что искомая производная $\nabla f = (A + A^T)x$. Обратите внимание, что размерности не нарушены и мы получили столбец из производных, то есть искомый градиент нашей функции f.

По аналогии мы легко можем найти вторую производную. Для этого надо взять производную производной. Функция $g(x) = (A + A^T)x$ бьёт из векторов в вектора

$$f(X) : R^n \to R^m.$$

На самом деле с такой ситуацией мы также знакомились на математическом анализе. Если $n = 1$ то у нас есть m функций, каждая из которых применяется к x. На выходе получается вектор

$$\begin{pmatrix} f_1(x) \\ f_2(x) \\ \dots \\ f_m(x). \end{pmatrix}$$

Если мы хотим найти производную, нужно взять частную производную каждой функции по x и записать в виде вектора. Дифференциал также будет представлять из себя вектор, так как при приращении аргумента на какую-то величину изменяется каждая из функций

$$df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \dots \\ \frac{\partial f_m}{\partial x} \end{pmatrix} * \begin{pmatrix} dx \\ dx \\ \dots \\ dx \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x} dx \\ \frac{\partial f_2}{\partial x} dx \\ \dots \\ \frac{\partial f_m}{\partial x} dx \end{pmatrix}.$$

Под символом $*$ имеется в виду поэлементное умножение. Если $n > 1$, то аргументов на вход в такой вектор из функций идёт несколько, на выходе получается матрица

$$\begin{pmatrix} f_1(x_1) & f_1(x_2) & \dots & f_1(x_n) \\ f_2(x_1) & f_2(x_2) & \dots & f_2(x_n) \\ \dots & \dots & \ddots & \dots \\ f_m(x_1) & f_m(x_2) & \dots & f_m(x_n) \end{pmatrix}$$

Производной такой многомерной функции будет матрица из частных производных каждой функции по каждому аргументу

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \ddots & \dots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

Дифференциал снова будет представлять из себя вектор

$$df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \ddots & \dots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \cdot \begin{pmatrix} dx_1 \\ dx_2 \\ \dots \\ dx_n \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} dx_1 + \frac{\partial f_1}{\partial x_2} dx_2 + \dots + \frac{\partial f_1}{\partial x_n} dx_n \\ \frac{\partial f_2}{\partial x_1} dx_1 + \frac{\partial f_2}{\partial x_2} dx_2 + \dots + \frac{\partial f_2}{\partial x_n} dx_n \\ \dots \\ \frac{\partial f_m}{\partial x_1} dx_1 + \frac{\partial f_m}{\partial x_2} dx_2 + \dots + \frac{\partial f_m}{\partial x_n} dx_n \end{pmatrix}.$$

В обоих ситуациях мы можем записать дифференциал через матричное произведение. Вернёмся к поиску второй производной

$$dg(x) = (A + A^T)\, dx.$$

Выходит, что матрица из вторых производных для функции $f(x)$ выглядит как $A + A^T$. Обратите внимание, что для этой ситуации в каноническом виде нет транспонирования. Когда мы вытаскиваем из записи дифференциала производную, нам не надо его применять.

г. Когда мы хотим найти производную $f(x) = \ln(x^T A x)$, мы просто можем применить правила взятия производной от сложной функции $f(y) = \ln(y)$. К логарифму на вход идет скаляр, а значит его производная равна $\frac{1}{y}$. Выходит, что

$$df(x) = d\ln(y) = \frac{1}{y}\, dy = \frac{1}{y}\, d(x^T A x) = \frac{1}{x^T A x} \cdot x^T (A^T + A)\, dx.$$

В таких ситуация нужно быть осторожным и следить за тем, что на вход функции идёт скаляр. Если это не так, то ситуация усложняется и мы оказываемся в ситуации, где надо поднапрячься. Чуть ниже мы рассмотрим в качестве примера матричную экспоненту $\exp(X)$, где X — квадратная матрица.

д. Движемся к следующей ситуации. Функция бьёт из матриц в скаляры

$$f(X) : R^{n \times k} \to R.$$

В таком случае нам надо найти производную функции по каждому элементу матрицы, то есть дифференциал будет выглядеть как

$$df(X) = f'_{x_{11}}\, dx_{11} + f'_{x_{12}}\, dx_{12} + \dots + f'_{x_{nk}}\, dx_{nk}.$$

Его можно записать в компактном виде через след матрицы как

$$df(X) = \operatorname{tr}(\nabla f^T \, dX),$$

где

$$\nabla f = \begin{pmatrix} f'_{x_{11}} & \dots & f'_{x_{1k}} \\ \dots & \ddots & \dots \\ f'_{x_{n1}} & \dots & f'_{x_{nk}} \end{pmatrix}$$

Вполне естественен вопрос **а почему это можно записать именно так?** Давайте попробуем увидеть этот факт на каком-нибудь простом примере. Пусть у нас есть две матрицы

$$A_{[2\times 3]} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \qquad X_{[2\times 3]} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix}.$$

Посмотрим на то, как выглядит $\operatorname{tr}(A^T \, dX)$. Как это не странно, он совпадает с дифференциалом

$$\operatorname{tr}(A^T \, dX) = \operatorname{tr}\left(\begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{pmatrix} \begin{pmatrix} dx_{11} & dx_{12} & dx_{13} \\ dx_{21} & dx_{22} & dx_{23} \end{pmatrix} \right),$$

при произведении на выходе получаем матрицу размера 3×3

$$\begin{pmatrix} a_{11}\,dx_{11} + a_{21}\,dx_{21} & a_{11}\,dx_{12} + a_{21}\,dx_{22} & a_{11}\,dx_{13} + a_{21}\,dx_{23} \\ a_{12}\,dx_{11} + a_{22}\,dx_{21} & a_{12}\,dx_{12} + a_{22}\,dx_{22} & a_{12}\,dx_{13} + a_{22}\,dx_{23} \\ a_{13}\,dx_{11} + a_{23}\,dx_{21} & a_{13}\,dx_{12} + a_{23}\,dx_{22} & a_{13}\,dx_{13} + a_{23}\,dx_{23} \end{pmatrix}.$$

Дальше мы периодически будем пользоваться таким приёмом. Например, величину

$$\|X - A\|^2 = \sum_{i,j} (x_{ij} - a_{ij})^2$$

можно записать в матричном виде как

$$\operatorname{tr}((X - A)^T (X - A)).$$

Итак, найдём производную от $f(X) = a^T X A X a$. Нам нужно выписать дифференциал и привести его к каноническому виду

Обратим внимание на то, что мы бьём из матриц в скаляры. Дифференциал будет по своей размерности совпадать со скаляром. Производная будет размера матрицы

$$df(X) = d(a^T X A X a) = \underset{[1\times 1]}{a^T \, d(X) A X a} + \underset{[1\times 1]}{a^T X A \, d(X) a}.$$

Оба слагаемых, которые мы получаем после перехода к дифференциалу — скаляры. Мы хотим представить дифференциал в виде $\operatorname{tr}(\text{нечто}\, dX)$. След от скаляра это снова скаляр. Получается, что мы бесплатно можем навесить над правой частью наешго равенство знак следа и воспользоваться его свойствами

$$\begin{aligned} df(X) = d(a^T X A X a) &= \operatorname{tr}(a^T\, d(X) A X a) + \operatorname{tr}(a^T X A\, d(X) a) = \\ &= \operatorname{tr}(A X a a^T\, d(X)) + \operatorname{tr}(a a^T X A\, d(X)) = \\ &= \operatorname{tr}(A X a a^T\, d(X) + a a^T X A\, d(X)) = \operatorname{tr}((A X a a^T + a a^T X A)\, d(X)). \end{aligned}$$

Производная найдена, оказалось что это

$$\nabla f = (A X a a^T + a a^T X A)^T = a a^T X^T A^T + A^T X a a^T.$$

Как бы мы нашли это, всё по-честному перемножив, даже боюсь себе представлять.

е. Ещё один пример на ситуацию, когда функция бьёт из векторов в вектора

$$\underset{[n\times 1]}{f(x)} = \underset{[n\times 1]}{x}\ \underset{[1\times n]}{x^T}\ \underset{[n\times 1]}{x}.$$

В нём надо аккуратно вести себя со сложением матриц со скалярами. Берём дифференциал

$$df(x) = dx x^T x = dx x^T x + x\, dx^T x + x x^T\, dx.$$

В первом слагаемом пользуемся тем, что $x^T x$ скаляр и его можно вынести перед дифференциалом. Этот скаляр умножается на каждый элемент вектора. Дальше мы захотим вынести дифференциал за скобку, чтобы не испортить матричное сложение, подчеркнём факт этого перемножения на каждый элемент единичной матрицей. Во втором слагаемом пользуемся тем, что $dx^T x$ скаляр и транспонируем его

$$df(x) = \underset{[1\times 1]}{x^T x}\ \underset{[n\times n]}{I_n}\ \underset{[n\times 1]}{dx} + x x^T\, dx + x x^T\, dx = (x^T x I_n + 2 x x^T)\, dx.$$

Обратите внимание, что без единичной матрицы размерности у сложения поломаются. Получается, что наша производная выглядит как

$$\mathfrak{J} = x^T x I_n + 2 x x^T = \begin{pmatrix} \sum x_i^2 + 2x_1^2 & 2x_1x_2 & \dots & 2x_1x_n \\ 2x_1x_2 & \sum x_i^2 + 2x_2^2 & \dots & 2x_2x_n \\ \dots & \dots & \ddots & \dots \\ 2x_1x_n & 2x_nx_2 & \dots & \sum x_i^2 + 2x_n^2 \end{pmatrix}.$$

Как и ожидалось, на выходе получилась матрица.

ж. **В нашей таблице. осталось ещё несколько ситуаций, которые остались вне поля на-**

шего зрения. Давайте их обсудим более подробно. Например, давайте посмотрим на ситуацию когда отображение бьёт из матриц в вектора

$$f(X) : R^{n \times k} \to R^m.$$

Тогда X матрица, а $f(X)$ вектор. Нам надо найти производную каждого элемента из вектора $f(X)$ по каждому элементу из матрицы X. Получается, что $\frac{\partial f}{\partial X}$ — это трёхмерная структура. Обычно в таких ситуациях ограничиваются записью частных производных либо прибегают к более сложным, многомерным методикам. Мы такие ситуации опустим.

Упражнение 2

Давайте пополним таблицу дифференциалов несколькими новыми функциями, специфичными для матриц. Найдём матричные дифференциалы функций:

а. $f(X) = X^{-1}$, где матрица X размера $n \times n$

б. $f(X) = \det X$, где матрица X размера $n \times n$

в. $f(X) = \mathrm{tr}(X)$, где матрица X размера $n \times n$

г. Ещё больше матричных производных можно найти в книге The Matrix Cookbook[14]

Решение:

а. Найдём производную обратной матрицы. Тут логично вспомнить, что производная константы это ноль, обратная матрица определяется как $X^{-1} \cdot X = I_n$, а единичная матрица I_n это как раз константа. Берём дифференциал с обеих сторон нашего равенства

$$dX^{-1}X + X^{-1}\,dX = dI_n = 0,$$

отсюда получаем что

$$dX^{-1} = -X^{-1}\,dXX^{-1}.$$

б. Определитель — это функция, которая бьёт из матриц в скаляры. Его можно искать по-разному, один из способов — разложение по строке

$$\det X = \sum_{j=1}^{n} x_{ij}(-1)^{i+j}M_{ij},$$

где M_{ij} — дополнительный минор матрицы X. Берём дифференциал

$$d(\det X) = (-1)^{1+1}M_{11}\,dx_{11} + (-1)^{1+2}M_{12}\,dx_{12} + \ldots = \sum_{ij} A_{ji}\,dx_{ij} = \mathrm{tr}(A\,dX),$$

[14] https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

где $A_{ij} = (-1)^{i+j} M_{ij}$, то есть A — матрица алгебраических дополнений. Трюк со следом аналогичен пункту д из предыдущей задачи. Вспомним, что обратную матрицу можно получить отталкиваясь от алгебраических дополнений по формуле

$$X^{-1} = \frac{A}{\det X}.$$

Выразим матрицу A и подставим её а получившееся выше уравнение

$$d(\det X) = \operatorname{tr}(\det(X) X^{-1}\, dX),$$

выходит что искомая производная равна

$$\nabla f = (\det X \cdot X^{-1})^T = \det X \cdot X^{-T}.$$

в. По аналогии с определителем след бьёт из пространства матриц в пространство скаляров, получается

$$d(\operatorname{tr} X) = \operatorname{tr}(I_n\, dX).$$

Это логично, так как след представляет из себя сумму диагональных элементов.

Упражнение 3

Найдите следующие производные:

а. $f(X) = \operatorname{tr}(AXB)$, где матрица A размера $p \times m$, матрица B размера $n \times p$, матрица X размера $m \times n$.

б. $f(X) = \operatorname{tr}(AX^T X)$, где матрица A размера $n \times n$, матрица X размера $m \times n$.

в. $f(X) = \ln \det X$

г. $f(X) = \operatorname{tr}(AX^T X B X^{-T})$

д. $f(X) = \det(X^T A X)$

е. $f(x) = x^T A b$, где матрица A размера $n \times n$, вектора x и b размера $n \times 1$.

ж. $f(A) = x^T A b$.

Решение:

Проверить правильность своего решения можно в матричном калькуляторе[15]. Не забывайте, что $\operatorname{tr}(A) = \operatorname{tr}(A^T)$ и что под знаком следа можно циклически переставлять матрицы, если размерность не ломается.

Упражнение 4

[15] http://www.matrixcalculus.org/

Рассмотрим задачу линейной регресии

$$L(w) = (y - Xw)^T(y - Xw) \to \min_w.$$

а. Найдите $L(w)$, выведите формулу для оптимального w.

б. Как выглядит шаг градиентного спуска в матричном виде?

в. Найдите $d^2L(w)$. Убедитесь, что мы действительно в точке минимума.

Решение:

Ради интереса убедимся, что перед нами в качестве функции потерь используется именно MSE, в качестве x_i будем обозначать i—ую строчку матрицы X

$$(y - Xw)^T(y - Xw) = \begin{pmatrix} y_1 - x_1^T w & \dots & y_n - x_n^T w \end{pmatrix} \begin{pmatrix} y_1 - x_1^T w \\ \dots \\ y_n - x_n^T w \end{pmatrix} = \sum_{i=1}^{n} (y_i - x_i^T w)^2.$$

Найдём дифференциал для нашей функции потерь, держим в голове что производная берётся по вектору w

$$\begin{aligned} dL = d[(y - Xw)^T(y - Xw)] &= d[(y - Xw)^T](y - Xw) + (y - Xw)^T d[(y - Xw)] = \\ &= d[(-Xw)^T](y - Xw) - (y - Xw)^T X\, dw = \\ &= -dw^T X^T(y - Xw) - (y - Xw)^T X\, dw = -2(y - Xw)^T X\, dw. \end{aligned}$$

Тут мы воспользовались тем, что $dw^T X^T(y - Xw)$ это скаляр и его можно транспонировать. Производная найдена. Шаг градиентного спуска будет выглядеть как

$$w_t = w_{t-1} + \gamma \cdot 2X^T(y - Xw).$$

Здесь γ — это скорость обучения. Приравняем производную к нулю, чтобы найти минимум для w. Получается система уравнений

$$2X^T(y - Xw) = 0 \qquad X^T y = X^T X w \qquad w = (X^T X)^{-1} X^T y.$$

При решении системы мы сделали предположение, что матрица $X^T X$ обратима. Это так, если в матрице X нет линейно зависимых столбцов, а также наблюдений больше чем переменных. Найдём вторую производную

$$d[-2X^T(y - Xw)] = 2X^T X\,dw.$$

Выходит, что $H = 2X^T X$. Так как матрица $X^T X$ положительно определена, по критерию Сильвестра, мы находимся в точке минимума.

Матрица $X^T X$ положительно определена по определению. Если для любого вектора $v \neq 0$ квадратичная форма $v^T X^T X v > 0$, матрица $X^T X$ положительно определена. При перемножении Xv у нас получается вектор. Обозначим его как z, значит $v^T X^T X v = z^T z = \sum_{i=1}^{n} z_i^2 > 0$.

Выпишем в явном виде второй дифференциал

$$d^2L = dw^T 2X^T X\,dw.$$

Упражнение 5

В случае Ridge-регрессии минимизируется функция со штрафом:

$$L(w) = (y - xw)^T(y - xw) + \lambda w^2,$$

где λ — положительный параметр, штрафующий функцию за слишком большие значения w.

а. Найдите $dL(w)$, выведите формулу для оптимального w.

б. Как выглядит шаг градиентного спуска в матричном виде?

в. Найдите $d^2L(w)$. Убедитесь, что мы действительно в точке минимума.

Решение:

$$dL = 2(y - Xw)^T X\,dw + 2\lambda w^T\,dw$$
$$\nabla L(w) = 2X^T(Xw - y) + 2\lambda w$$
$$w = (X^T X + \lambda I)^{-1} X^T y$$
$$w_t = w_{t-1} - \gamma \cdot (2X^T(Xw_{t-1} - y) + 2\lambda w_{t-1})$$
$$d^2L = dw^T(2X^T X + 2\lambda)\,dw$$
$$H = 2X^T X + 2\lambda \text{ положительно определена}$$

Упражнение 6

Пусть x_i — вектор-столбец $k \times 1$, y_i — скаляр, равный $+1$ или -1, w — вектор-столбец размера $k \times 1$. Рассмотрим логистическую функцию потерь с l_2 регуляризацией

$$L(w) = \sum_{i=1}^{n} \ln(1 + \exp(-y_i x_i^T w)) + \lambda w^T w$$

а. Найдите dL;

б. Найдите вектор-столбец ∇L.

в. Как для этой функции потерь выглядит шаг градиентного спуска в матричном виде?

Решение:

Используем весь арсенал, который обсудили выше. Начнём с одного слагаемого. Обозначим его как y. Это скаляр, значит

$$d \ln y = \frac{1}{y} dy = \frac{1}{\ln(1 + \exp(-y_i x_i^T w))} \cdot -y_i \exp(-y_i x_i^T w) \cdot x_i^T dw.$$

Выписываем дифференциал

$$dL = \left(-\sum_{i=1}^{n} \frac{y_i \exp(-y_i x_i^T w)}{1 + \exp(-y_i x_i^T w)} \cdot x_i^T + 2\lambda w^T \right) dw.$$

Можно записать градиент с помощью сигмоиды $\sigma(z) = \frac{1}{1+\exp(-z)}$. Получится, что

$$\nabla L = \sum_{i=1}^{n} -y_i \sigma(-y_i x_i^T w) x_i + 2\lambda w.$$

Выходит, что шаг градиентного спуска можно записать как

$$w_t = w_{t-1} + \gamma \cdot \nabla L.$$

Упражнение 7

Упражняемся в матричном методе максимального правдоподобия. Допустим, что выборка размера n пришла к нам из многомерного нормального распределения с неизвестными вектором средних μ и ковариационной матрицей Σ. В этом задании нужно найти оценки максимального правдоподобия для $\hat{\mu}$ и $\hat{\Sigma}$. Обратите внимание, что выборкой здесь будет не $x_1, \dots, x_n$, а

$$\begin{pmatrix} x_{11}, \dots, x_{n1} \\ \dots \\ x_{n1}, \dots, x_{nm} \end{pmatrix}$$

Решение:

Плотность распределения для m—мерного вектора y будет выглядеть как

$$f(x \mid \mu, \Sigma) = \frac{1}{(\sqrt{2\pi})^m \cdot \sqrt{\det \Sigma}} \cdot \exp\left(-\frac{1}{2} \cdot (x-\mu)^T \Sigma^{-1} (x-\mu)\right).$$

В силу того, что все наблюдения независимы, функция правдоподобия для выборки объёма n примет вид:

$$L(x \mid \mu, \Sigma) = \frac{1}{(\sqrt{2\pi})^{m \cdot n} \cdot \sqrt{\det \Sigma}^n} \cdot \exp\left(-\frac{1}{2} \cdot \sum_{i=1}^{n} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu)\right).$$

Прологарифмировав правдоподобие, получим

$$\ln L(x \mid \mu, \Sigma) = -\frac{m \cdot n}{2} \ln 2\pi - \frac{n}{2} \ln \det \Sigma - \frac{1}{2} \sum_{i=1}^{n} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu)$$

Нам нужно найти максимум этой функции по μ и Σ. Начнём с μ. Аргумент Σ будем считать константой. Обозначим такую функцию за $f(\mu)$. Эта функция бьёт с множества векторов в множество скаляров. Значит дифференциал этой функции можно записать в виде:

$$df(\mu) = \nabla f^T d\mu.$$

Найдём этот дифференциал. Не будем забывать, что дифференциал от константы нулевой, а также что дифференциал суммы равен сумме дифференциалов

$$\begin{aligned} df(\mu) = -\frac{1}{2} \cdot d \sum_{i=1}^{n} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu) = -\frac{1}{2} \cdot \sum_{i=1}^{n} d[(x_i-\mu)^T \Sigma^{-1} (x_i-\mu)] = \\ = -\frac{1}{2} \cdot \sum_{i=1}^{n} d[(x_i-\mu)^T] \Sigma^{-1} (x_i-\mu) + (x_i-\mu)^T \Sigma^{-1} d[(x_i-\mu)] = \\ = \frac{1}{2} \cdot \sum_{i=1}^{n} d\mu^T \Sigma^{-1} (x_i-\mu) + (x_i-\mu)^T \Sigma^{-1} d\mu. \end{aligned}$$

Первое слагаемое под суммой имеет размерность $1 \times m \cdot m \times m \cdot m \times 1$. Это константа. Если мы протранспонируем константу, ничего не изменится. Обратим внимание, что матрица Σ симметричная и при транспонировании не меняется. Сделаем этот трюк

$$\frac{1}{2}\cdot\sum_{i=1}^{n} d\mu^T\Sigma^{-1}(x_i-\mu)+(x_i-\mu)^T\Sigma^{-1}d\mu=\frac{1}{2}\cdot\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}d\mu+(x_i-\mu)^T\Sigma^{-1}d\mu=$$

$$=\frac{1}{2}\cdot\sum_{i=1}^{n}[(x_i-\mu)^T\Sigma^{-1}+(x_i-\mu)^T\Sigma^{-1}]d\mu=\left[\cdot\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}\right]d\mu$$

Получается, что $f'(\mu)=\sum_{i=1}^{n}\Sigma^{-1}(x_i-\mu)$. Приравняв производную к нулю и домножив обе части уравнения слева на Σ, получим оптимальное значению μ:

$$\sum_{i=1}^{n}\Sigma^{-1}(x_i-\hat{\mu})=0$$
$$\sum_{i=1}^{n}(x_i-\hat{\mu})=0$$
$$\sum_{i=1}^{n}x_i=n\cdot\hat{\mu}\Rightarrow\hat{\mu}=\bar{x}.$$

Не будем забывать, что в записях выше x и μ были векторами-столбцами размерности $m\times 1$. В итоговом ответе они также являются векторами-столбцами такой размерности.

Займёмся оценкой для Σ. Аргумент μ будем считать константой. Обозначим такую функцию за $f(\Sigma)$

$$f(\Sigma)=-\frac{n}{2}\ln\det\Sigma-\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu).$$

Эта функция бьёт с множества матриц в множество скаляров. Значит дифференциал этой функции можно записать в виде:

$$df(\Sigma)=\operatorname{tr}(\nabla f^T dx).$$

Начнём с первого слагаемого. Для него нам понадобится вспомнить как выглядит дифференциал для определителя

$$-\frac{n}{2}\frac{1}{\det\Sigma}d[\det\Sigma]=-\frac{n}{2}\frac{1}{\det\Sigma}\operatorname{tr}(\det\Sigma\cdot\Sigma^{-T}d\Sigma)=-\operatorname{tr}(\frac{n}{2}\cdot\Sigma^{-1}d\Sigma).$$

Теперь поработаем со вторым слагаемым. В нём нас интересует дифференциал обратной матрицы

$$-\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T d[\Sigma^{-1}](x_i-\mu)=\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}\cdot d\Sigma\cdot\Sigma^{-1}(x_i-\mu).$$

Под знаком суммы размерность каждого слагаемого $1\times m\cdot m\times m\cdot m\times m\cdot m\times m\cdot m\times 1$. Это константа. Если мы возьмём от неё след, ничего не изменится. Взяв след, переставим внутри множители

$$\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}\cdot d\Sigma\cdot\Sigma^{-1}(x_i-\mu)=\frac{1}{2}\sum_{i=1}^{n}\operatorname{tr}(\Sigma^{-1}(x_i-\mu)\cdot(x_i-\mu)^T\Sigma^{-1}\cdot d\Sigma).$$

Сумма следов — след суммы. Объединяем наши слагаемые в месте. В первом множитель n подменяем на сумму

$$df(\Sigma)=\operatorname{tr}\left(\left[-\frac{1}{2}\sum_{i=1}^{n}\Sigma^{-1}+\Sigma^{-1}(x_i-\mu)\cdot(x_i-\mu)^T\Sigma^{-1}\right]d\Sigma\right)$$

Забираем себе из-под знака дифференциала производную. Под знаком суммы после транспонирования ничего не поменяется. Приравниваем производную к нулю, домножим справа каждое слагаемое на Σ. На четвёртой строчке домножим слева на Σ:

$$\begin{aligned}
&\frac{1}{2}\sum_{i=1}^{n}-\Sigma^{-1}+\Sigma^{-1}(x_i-\mu)\cdot(x_i-\mu)^T\Sigma^{-1}=0\\
&-n\cdot\Sigma^{-1}+\sum_{i=1}^{n}\Sigma^{-1}(x_i-\mu)\cdot(x_i-\mu)^T\Sigma^{-1}=0\\
&-n+\Sigma^{-1}\sum_{i=1}^{n}(x_i-\mu)\cdot(x_i-\mu)^T=0\\
&-n\Sigma+\sum_{i=1}^{n}(x_i-\mu)\cdot(x_i-\mu)^T=0\\
&\Sigma=\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)\cdot(x_i-\mu)^T
\end{aligned}$$

До оценок остался один шаг. Вспоминаем оценку для μ, подставляем её в уравнение и получаем, что

$$\hat{\Sigma}=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})\cdot(x_i-\bar{x})^T.$$

Не забываем, что x_i и $\bar{x}$ — вектора размерности $m\times 1$.

Упражнение 8

Найдите симметричную матрицу X наиболее близкую к матрице A по норме Фробениуса, $\sum_{i,j}(x_{ij} - a_{ij})^2$. Тут мы просто из каждого элемента вычитаем каждый и смотрим на сумму квадратов таких разностей. То есть решите задачку условной матричной минимизации

$$\begin{cases} \|X - A\|^2 \to \min_A \\ X^T = X \end{cases}$$

Hint: Надо будет выписать Лагранджиан. А ещё пригодится тот факт, что $\sum_{i,j}(x_{ij} - a_{ij})^2 = \|X - A\|^2 = \operatorname{tr}((X - A)^T(X - A))$.

Решение:

Выписываем лагранджиан

$$\mathcal{L} = \sum_{i,j}(x_{ij} - a_{ij})^2 + \sum_{ij}\lambda_{ij}(x_{ij} - x_{ji}) = \operatorname{tr}((X - A)^T(X - A)) + \operatorname{tr}(\Lambda^T(X - X^T)) = \\ = \operatorname{tr}(X^TX) - 2\operatorname{tr}(X^TA) + \operatorname{tr}(A^TA) + \operatorname{tr}(\Lambda^T(X - X^T))$$

Найдём все необходимые нам дифференциалы

$$\begin{aligned} &\mathrm{d}\operatorname{tr}(X^TX) = \operatorname{tr}(\mathrm{d}(X^TX)) = \operatorname{tr}(X^T\,\mathrm{d}X) + \operatorname{tr}(\mathrm{d}X^TX) = \operatorname{tr}(2X^T\,\mathrm{d}X) \\ &\mathrm{d}\operatorname{tr}(X^TA) = \operatorname{tr}(A^T\,\mathrm{d}X) \\ &\mathrm{d}\operatorname{tr}(\Lambda^TX) = \operatorname{tr}(\Lambda^T\,\mathrm{d}X) \\ &\mathrm{d}\operatorname{tr}(\Lambda^TX^T) = \operatorname{tr}(\Lambda\,\mathrm{d}X) \end{aligned}$$

Выписываем в яном виде производную по X

$$\frac{\partial\mathcal{L}}{\partial X} = 2X^T - 2A^T + \Lambda^T - \Lambda = 0$$

Нужно избавиться от Λ, давайте транспонируем уравнение

$$\frac{\partial\mathcal{L}}{\partial X} = 2X - 2A + \Lambda - \Lambda^T = 0,$$

а после прибавим его к исходному, тогда лишние части исчезнут

$$4X - 2A^T - 2A = 0 \qquad X = \frac{1}{2}(A + A^T).$$