Распишем аналитически, эту модель. Сигмоиду, которая является последним преобразованием этой модели, подробно обсудили выше, потому сразу начнем с записи сумматорной функции выходного слоя z^2.

$$z^2 = \omega_{11}^2 f(z_1^1) + \omega_{21}^2 f(z_2^1) + b^2,$$

где z_i^j – значение сумматорной функции *i-го* нейрона *j-го* слоя;

b^l – свободный член *l-го* слоя.

Используя определение ReLU функции (1.12) можно последнюю формулу разложить на четыре случая:

$$z^2 = \begin{cases} b^2, & z_1^1 < 0 \text{ и } z_2^1 < 0; \\ \omega_{11}^2 z_1^1 + b^2, & z_1^1 \geq 0 \text{ и } z_2^1 < 0; \\ \omega_{21}^2 z_2^1 + b^2, & z_1^1 < 0 \text{ и } z_2^1 \geq 0; \\ \omega_{11}^2 z_1^1 + \omega_{21}^2 z_2^1 + b^2, & z_1^1 \geq 0 \text{ и } z_2^1 \geq 0. \end{cases} \quad (1.13)$$

Для решеная задачи классификации будет хорошо если каждая область, по штриховке на рисунке, получит свой вид решающего правила – получиться так, что, например, область синих точек отделенная обеими дискриминирующими линиями будет получать вероятности принадлежности ко второму классу по своему отдельному уравнению которое сформируется так чтобы дать наименьшие вероятности. Аналогично, но менее однозначно, вероятности будут формироваться для областей с единственной штриховкой. Для области без штриховки, очевидно, правило должно выстроиться так, чтобы давать наибольшие вероятности.

Зная законы по которым формировалась выборка из примера, можно согласовать веса первого слоя так, чтобы каждому случаю из формулы (1.13) соответствовала своя область рисунка. Это можно сделать несколькими способами, но пусть незаштрихованной области соответствует первый, области с наклоненной влево штриховкой второй, вправо – третий и области с двумя штриховками остается четвертый.

Покажем, как провести данное согласование для такого рисунка, хотя в общем это может выглядеть по другому – все зависит от того как проведены дискриминирующие линии.

На легенде рисунка обозначены уравнения дискриминирующих линий, начнем с первого. В данном случае области с наклоненной влево штриховкой соответствует неравенство:

$$c_1 x_1 + c_2 x_2 + c > 0.$$

Потому присвоив весам нейронной сети $\omega_{11}^1 := c_1$, $\omega_{12}^1 := c_2$ и свободному члену первого нейрона первого слоя $b_1^1 := c$, получим, что второй случай

формулы (1.13) действительно соответствует области с наклоненной влево штриховкой.

Области с наклоненной вправо штриховкой соответствует неравенство:

$$d_1 x_1 + d_2 x_2 + d < 0.$$

Потому присвоив весам нейронной сети $\omega_{12}^1 \coloneqq -d_1,\ \omega_{22}^1 \coloneqq -d_2$ и свободному члену второго нейрона первого слоя $b_2^1 \coloneqq -d$, получим, что третий случай формулы (1.13) действительно соответствует области с наклоненной вправо штриховкой.

При описанном выше распределении весов области с двумя штриховками автоматически будет соответствовать последний случай формулы (1.13). Так получиться, что первый слой идеально разделит область наблюденных данных на четыре части, что обеспечит такую ситуацию, что сигнал в выходной нейрон будут подавать только наблюдения обозначенные синим.

Выше было обозначено, что в рамках этого примера мы будем моделью оценивать вероятность того, что конкретное наблюдение принадлежит к второму классу обозначенному красным. Для того, чтобы оценки вероятностей области без штриховки были выше нежели любые другие, можно весам второго слоя присвоить любые отрицательные значения – положительный сигнал выходящий из первого слоя в результате попадания в модель любого «синего» наблюдения будет помножаться на отрицательное число и занижать сумматорную функцию z^2 и, как следствие, занижать оценки вероятностей первого класса, чего мы и добивались. В данном случае свободный член сумматорной функции выходного нейрона может принимать любое значение – все равно механизм описанный выше будет занижать оценки вероятностей для любого наблюдения из первого класса относительно наблюдений из второго класса, что обеспечит нам стопроцентную точность классификации.

Все это мы вели к тому, что можно моделью нейронной сети архитектуры, представленной на рисунке 1.10, добиться идеальной классификации рассматриваемой задачи – ограничения логистической регрессии преодолены.

Однако, в любой реальной задаче классификации принципы разделения на классы, конечно, неизвестны, потому веса и свободные члены каждого слоя оцениваются статистически. На рисунке 1.9 справа показана работа реально обученной только на статистических данных нейронной сети. Для каждого наблюдения мы вычислили сумматорные функции нейронов первого слоя и обозначили на рисунке формой и цветом как они распределились по знаку названной функции. Присмотревшись к такой диаграмме рассеяния мы заметили, что без ошибок не обошлось – они выделены на рисунке черными кружками.

При построении моделей опирающихся на искусственные нейронные сети, кроме множества возможностей для идентификации нейронной сети, существуют возможности настройки алгоритма обучения, что может вести вообще говоря, к различным моделям. Модель для поставленной задачи, точно

можно привести к идеальной классификации, но этот пример нам даже на руку, можно показать роль, которую в финальном решении играют веса выходного слоя.

Как зависят предсказываемые вероятности от конкретной точки в системе координат представлено на рисунке А.2 – сигмоида состоящая из двух участков под разными углами. И несмотря на то, что ряд пограничных точек приводят к формированию сигнала в выходной слой (хотя по заложенным закономерностям недолжны), веса и свободный член выходного слоя придали им большую сумматорную функцию выходного слоя, нежели для наблюдений «синего» класса. Это привело к тому что все точки относящиеся ко второму классу на графике А.2 выше, нежели точки первого класса – от сюда, правильно выбрав высоту p', можно добиться 100% точности классификации.

Этот подраздел работы вводит понятие искусственной нейронной сети, с краткими наиболее общими идеями лежащими в основе этой группы моделей. Нейронная сеть представляет собой последовательность преобразований входных сигналов, каждое из которых в своей сумматорной функции сочетает выходные сигналы предыдущего слоя и к этому сочетанию применяет активационную функцию результат которой отправляется в сумматорные функции следующего слоя или, для выходного слоя, формирует предсказание модели. Далее вводиться архитектура нейронной сети в которой скрытые слои представляют собой ReLU функции а выходной слой содержит сигмоидальный нейрон. На примере подробно раскрыты механизмы почему эта модель работает, как она связана и развивает идеи модели логистической регрессии.

В этом разделе основное внимание было уделено тому как названная модель применяется для известной закономерности, но на практике закономерности неизвестны, а имеются лишь наблюденные данные и идентификацию и оценку параметров надо производить в этих условиях. В следующем подразделе описаны идеи, как производиться оценка параметров, а более детальная идентификация модели в контексте нейронных сетей лучше всего раскрывается на практике, и будет представлена в третьей главе.

1.4 Целевые функции и алгоритм обратного распространения ошибки

Процесс оценки коэффициентов в контексте нейронных сетей принято называть обучением модели.

Ключевым пунктом является выдвижение некоторого правила которое оценивает насколько модель соответствует наблюденным данным – целевую функцию. В общем, требуется ввести функцию:

$$F\left(Y, \hat{Y}\right),$$

где Y – реально наблюденный вектор предсказываемого явления;

$\hat{Y}$ – вектор текущих предсказаний.

Очевидно, что вектор предсказаний формируется данными, проходящими через модель т.е. правомерна запись:

$$\hat{Y} = \hat{Y}(X, W),$$

где X – реальное множество наблюдений за факторами для которых предполагается влияние на Y;

W – множество коэффициентов которые на разных этапах оказывают влияние на отклик вычисляемый моделью.

Итак, окончательно, обобщенная целевая функция примет вид:

$$J(Y, X, W).$$

Её значение должно быть тем больше, чем сильнее отличаются наблюденные и предсказанные значения. Нам выгодно, чтобы полученные предсказания были максимально похожи на наблюденные, потому нам тем лучше, чем меньше эта функция.

В контексте рассматриваемого вопроса можно сказать, что множества X и Y неизменны (хотя некоторые продвинутые обучающие алгоритмы могут использовать на разных стадиях обучения разные подмножества этих множеств).

Приходим к тому, что необходимо подобрать такие коэффициенты W, чтобы целевая функция J была минимальна, или более формально:

$$W^* = argmax_W[J(Y, X, W)].$$

По выводам предыдущего подраздела очевидно, что в случае нейронной сети $\hat{Y}(X, W)$ – нелинейная функция, да и все широко применяемые целевые функции также не линейны, потому очевидно, что перед нами стоит задача нелинейной оптимизации.

Для решения таких задач широко распространена группа методов основанных на градиентном спуске. Основная идея этих методов заключается в том, чтобы постепенно от некоторой выбранной начальной точки двигаться в направлении наискорейшего убывания J.

В методах градиентного спуска используется свойство антиградиента функции, которое утверждает, что функция в каждой точке убывает быстрее всего в направлении ее антиградиента, который определяется так:

$$-\nabla J = -\frac{dJ}{dW} = -\left(\frac{dJ}{d\omega_1}, \frac{dJ}{d\omega_2}, \dots, \frac{dJ}{d\omega_n}\right),$$

где ω_i – некоторый отдельный коэффициент модели $i = \overline{1, n}$.

Возвращаясь к искусственным нейронным сетям, приходим к тому, что единственная сложность реализации этого алгоритма заключается в том, чтобы получить частные производные по всем весам. К решению этой проблемы

призван метод обратного распространения ошибки. Будем его рассматривать как надстройку над методами градиентного спуска.

В процессе разъяснения принципа метода обратного распространения ошибки нам пригодится рисунок 1.11.

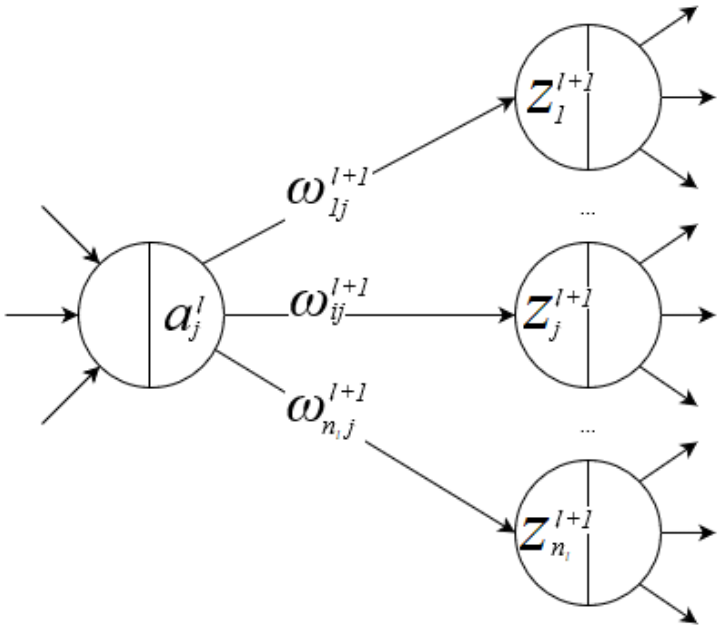

Рисунок 1.11 – Связь соседних слоев
Примечание – Источник: собственная разработка.

И некоторые обозначения. Матрица весов *(l+1)-го* слоя

$$W^{l+1} = (\omega_{ij}^{l+1})_{n_{l+1} \times n_l}.$$

Вектор из весов помножаемых на *j-ю* активацию *l-го* слоя, это-же столбец *j* матрицы весов *(l+1)-го* слоя

$$(W^{l+1})_j. \tag{1.14}$$

И так называемая, ошибка *j-го* нейрона *l-го* слоя, определяемая так:

$$\delta_j^l = \frac{\partial J}{\partial z_j^l}. \tag{1.15}$$

Выясним взаимосвязь ошибок *l-го* и *(l+1)-го* слоев. Для того придадим некоторое малое приращение Δ активации a_j^l и посмотрим какое приращение при этом получат сумматорные функции *(l+1)-го* слоя. В силу линейности сумматорных функции приращение для z_j^{l+1} составит $\Delta\omega_{ij}^{l+1}, j = \overline{1,n}$.

Пользуясь свойством, что при достаточно малом приращении аргументов соответствующее приращение функции близко к скалярному произведению градиента и вектора приращений аргументов получаем:

$$\Delta J\left(z_1^{l+1}, \dots, z_j^{l+1}, \dots, z_{n_{l+1}}^{l+1}\right) = \\ = \left(\frac{\partial J}{\partial z_1^{l+1}}, \dots, \frac{\partial J}{\partial z_j^{l+1}}, \dots, \frac{\partial J}{\partial z_{n_{l+1}}^{l+1}}\right)\left(\Delta\omega_1^{l+1}, \dots, \Delta\omega_j^{l+1}, \dots, \Delta\omega_{n_{l+1}}^{l+1}\right).$$

Или, используя свойство ассоциативности скалярного произведения, обозначения (1.14) и (1.15):

$$\Delta J(z^{l+1}) = \Delta\delta^{l+1}(W^{l+1})_j,$$

где z^{l+1} – вектор из значений сумматорной функции *(l+1)-го* слоя;

δ^{l+1} – вектор ошибок *(l+1)-го* слоя.

Разделив обе части уравнения на Δ, устремив его к нулю и вспомнив, что это приращение активационной функции a_j^l запишем:

$$\frac{\partial J}{\partial a_j^l} = \delta^{l+1}(W^{l+1})_j.$$

Далее вспоминая, что a_j^l, функция от z_j^l и, используя правило дифференцирования сложной функции:

$$\frac{\partial J}{\partial z_j^l} = \frac{\partial J}{\partial a_j^l}\frac{da_j^l}{dz_j^l} = \delta^{l+1}(W^{l+1})_j\frac{da_j^l}{dz_j^l}.$$

Опять обратившись к (1.15) и переходя к матричной форме записи, получим:

$$\delta^l = (W^{l+1})^T\delta^{l+1} \odot \frac{da_j^l}{dz_j^l}. \qquad (1.16)$$

Формула (1.16) самый важный вывод метода обратного распространения ошибки, собственно она отражает название метода – каждая предыдущая ошибка итеративно вычисляется из следующей, конечно, кроме ошибки выходного слоя, которую можно записать так:

$$\delta_j^L = \frac{\partial J}{\partial z_j^L} = \frac{\partial J}{\partial a_j^L}\frac{da_j^L}{dz_j^L}.$$

Или переходя к матричной записи:

$$\delta^L = \frac{dJ}{da^L} \odot \frac{da_j^L}{dz_j^L}. \tag{1.17}$$

Заметим также, что основной целью было названо выделение частных производных целевой функции по весам и свободным членам. Учитывая (1.15) и то что z_j^l функция от весов и свободных членов, не составляет труда очередной раз использовать правило дифференцирования сложной функции и получить:

$$\begin{aligned} \frac{\partial J}{\partial b_j^l} &= \frac{\partial J}{\partial z_j^l}\frac{\partial z_j^l}{b_j^l} = \delta_j^l; \\ \frac{\partial J}{\partial \omega_{ji}^l} &= \frac{\partial J}{\partial z_j^l}\frac{\partial z_j^l}{\partial \omega_{ji}^l} = a^{l-1}\delta_j^l. \end{aligned} \tag{1.18}$$

К сожалению, ни в одном из источников мы не нашли причины, по которой, используется именно такой подход, а не просто аналитически выводятся производные по весам и смещениям. Но можно предположить, что такой подход однозначно проще подлежит программированию – достаточно знать лишь производные активационных функций по их аргументам, и вне зависимости от слоя в котором находиться нейрон, зная его активацию, для текущих параметров модели можно получить производную по весам и смещениям используя одни и те-же рекуррентные формулы (1.18).

Заметим также, что рассуждения, касающиеся этого метода, мы строили в отрыве от наблюденных данных *X, Y*. Все выше сказанное будет справедливо для каждого конкретного примера по отдельности, а должны быть учтены все примеры участвующие в обучении. Благо все целевые функции построены на том, что суммируют ошибки отдельных примеров, а производная суммы, как известно, сумма производных потому результаты полученные из (1.18) надо просто просуммировать для каждого отельного примера и будет получена действительно производная целевой функции.

И так, теперь, на основе метода обратного распространения ошибки сформируем одноименный алгоритм оценки параметров нейронной сети:

1. Выполнить прямое распространение активации для каждого примера (подставить *X* в модель), сохраняя промежуточные активации;
2. Использовать формулу (1.17) для вычисления ошибок выходного слоя;
3. Произвести обратное распространение ошибки используя формулу (1.16) – будут получены ошибки всех нейронов сети;
4. Используем формулы (1.18) просуммировав их значения для каждого примера – будут получены градиенты по весам и смещениям;
5. Обновление параметров вдоль антиградиента.

Описанные пункты повторяются до тех пор пока не будут выполнены условия остановки алгоритма выбранной вариации градиентного спуска. [5 с. 243]

В целом, эта глава была посвящено методам классификации. Изначально была раскрыта подстановка задачи – требуется на основе наблюденных данных сформировать правило, которое получив некоторую информацию об объекте сможет отнести его к нужному классу. В приложении к кредитному риску: на основе данных предоставляемых клиентом в заявке на получение кредита, требуется принять решение о выдаче или удержании займа.

Описанная задача не нова и для ее решения широко используются, уже ставшие классическими, методы: логистическая регрессия и дискриминантный анализ. В начале главы описаны предпосылки использования логистической регрессии – невозможность решения поставленной задачи обычными регрессионными моделями объясняется, тем что эта группа моделей предсказывает число, в то время как нам требуется получить предсказание класса. Логистическая регрессия обходит это ограничение благодаря свойствам логит функции, которые позволяют предсказанное значение интерпретировать как вероятность отнесения к некоторому классу. В подразделе 1.2 подробно описывается процесс перехода к модели именно такой формы.

В том же разделе мы указали и на ограничения модели логистической регрессии – несмотря на очевидную нелинейность используемой формулы модель логистической регрессии остается линейным классификатором.

Для преодоления этого ограничения мы использовали концепцию нейронной сети, рассмотренной как развитие модели логистической регрессии через усложнение формы функции от показателей лежащей под логит функцией. По сути вместо линейной функции под логит функцию была положена кусочно-линейная функция. В подразделе 1.3 мы постарались раскрыть как этот механизм приводит к улучшению классифицирующих свойств модели в некоторых случаях.

Завершается глава исчерпывающим описанием математики используемой для формирования алгоритма оценки коэффициентов нейронной сети – метода обратного распространения ошибки. В целом, это тот же численный метод оптимизации как градиентный спуск, но частные производные целевой функции вычисляются особым способом, очевидно, особенно удобным для программирования.

На практике, кроме оценки коэффициентов, для модели требуется дополнительно еще целый комплекс мер связанных подготовкой данных. Кроме того в идентификация модели нейронной сети это отдельное «искусство» неразрывно связано с методом подбора решения – хитрости и уловки используемые соответствующими специалистами лучше всего раскрывать на практике. Этим вопросам и посвящены следующие разделы этой работы.