Нейронные сети. Общая структура (особый класс функций для оптимизации). Backpropagation как вычислительный подход

Жорникова Полина, 622 гр.

Содержание

1 Особый класс функций для задачи аппроксимации

Пусть X — множество объектов, Y — множество ответов; объекты описываются p числовыми признаками $f_j : X \to \mathbb{R}$, $j = 1, \ldots, p$. Вектор $(x^1, \ldots, x^p) \in \mathbb{R}^p$, где $x^j := f_j(x)$, называется *признаковым описанием* объекта x.

Рассмотрим стандартную задачу построения предсказывающей модели:

$$Q(a, X^n) = \frac{1}{n} \sum_{j=1}^{n} \mathscr{L}(a, x_j, y_j) \to \min_w, \tag{1}$$

где алгоритм a задаётся следующим образом:

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^{p} w_j f_j(x) - w_0 \right), \tag{2}$$

где $w = (w_1, \ldots, w_p) \in \mathbb{R}^p$ — вектор параметров; $\sigma : \mathbb{R} \to \mathbb{R}$ — функция, называющаяся *функцией активации*, w_0 — порог активации.

Если, например, $\sigma(z) = \text{sign}(z)$, то $a(x, w)$ — просто линейный классификатор. Уравнение $\langle w, x \rangle = 0$ задаёт гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^p$. Если вектор x находится по одну сторону гиперплоскости с её направляющим вектором w, то объект x относится к классу $+1$, иначе — к классу -1.

Схема для функции (2) представлена на рис. 1.

Рассматриваемый класс функций (2) имеет ряд преимуществ, речь о которых пойдет ниже.

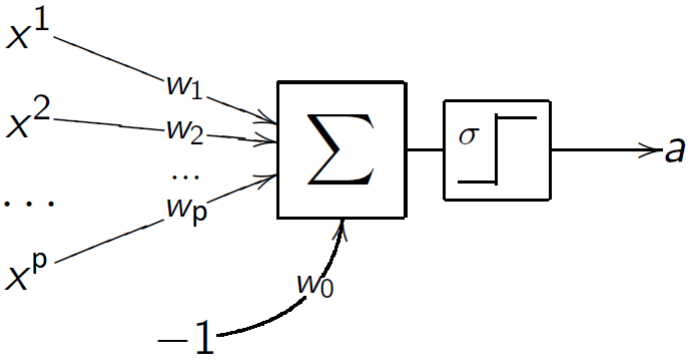


Рис. 1: Схема для особого класса функций

2 Модель нейрона МакКаллока-Питтса

Класс функций (2) имеет биологическую интерпретацию и является простейшей математической моделью нервной клетки — *нейрона*, рис. 2, предложенной МакКаллоком и Питтсом в 1943 году.

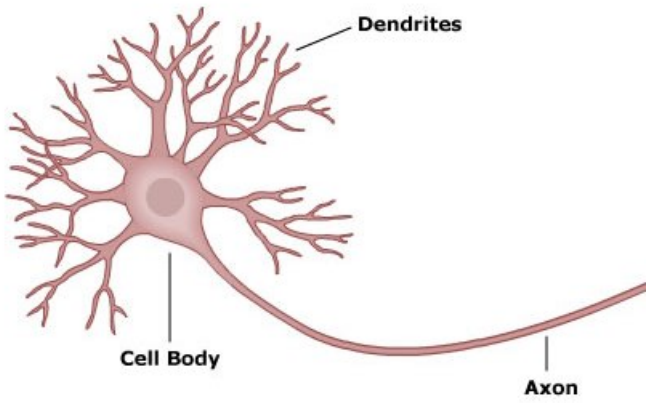


Рис. 2: Нервная клетка

Нейрон имеет множество разветвлённых отростков — *дендритов*, и одно длинное тонкое волокно — *аксон*, на конце которого находятся *синапсы*, примыкающие к дендритам других нервных клеток. Нервная клетка может находиться в двух состояниях: обычном и возбуждённом. Клетка возбуждается, когда в ней накапливается достаточное количество положительных зарядов. В возбуждённом состоянии клетка генерирует электрический импульс, который проходит по аксону до синапсов. Синапс при приходе импульса выделяет вещество, способствующее проникновению положительных зарядов внутрь соседней клетки, примыкающей к данному синапсу. Синапсы имеют разную способность концентрировать это вещество, причём некоторые даже препятствуют его выделению — они называются тормозящими (отсюда в формуле (2) и возникает w_0).

Нервную клетку можно рассматривать как устройство, которое на каждом такте своей работы принимает заряды величиной $x^j = f_j(x)$ от p входов — синапсов, примыкающих к её дендритам. Поступающие заряды складываются с весами w_j. Если вес w_j положительный, то j-й синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный заряд превышает порог активации w_0, то нейрон воз-

буждается и выдаёт на выходе $+1$, иначе выдаётся -1, когда $\sigma(z) = \text{sign}(z)$ — именно этот вариант имеется ввиду в модели МакКаллока-Питтса, но мы рассматриваем более общий случай. Поэтому x^j в (1) называются числовыми входами, σ — функцией активации, а w_0 порогом активации.

По аналогии с человеческим мозгом появилась гипотеза, что, соединив большое число функций (2), возможно создать универсальную машину, способную обучаться решению любых задач. На основе этой идеи, высказанной ещё в 50-е годы и названной принципом коннективизма, строятся *искусственные нейронные сети*.

На самом деле механизмы функционирования нервных клеток гораздо сложнее описанных выше. В нейрофизиологии известны десятки различных типов нейронов, и многие из них функционируют иначе. Однако в теории искусственных нейронных сетей не ставится задача максимально точного воспроизведения функций биологических нейронов. Цель в том, чтобы подсмотреть некоторые принципы в живой природе и использовать их для построения обучаемых устройств.

3 Способность нейронов аппроксимировать

Приняв модель нейрона (2) мы можем использовать её для решения различных задач машинного обучения.

Например, для задачи регрессии, когда $Y = \mathbb{R}$, $a(x_j, w) = \sigma(\langle w, x_j \rangle)$,

$$Q(w; X^n) = \sum_{j=1}^{n} \mathscr{L}\left(\langle w, x_j \rangle, y_j\right) = \sum_{j=1}^{n} \left(\sigma(\langle w, x_j \rangle) - y_j\right)^2 \to \min_w .$$

При $\sigma(z) = z$ получаем многомерную линейную регрессию.

Или для задачи линейной классификации, когда $Y = \{\pm 1\}$, $a(x_j, w) = \text{sign}\langle w, x_j \rangle$,

$$Q(w; X^n) = \sum_{j=1}^{n} \mathscr{L}\left(\langle w, x_j \rangle, y_j\right) = \sum_{j=1}^{n} [y_j \langle w, x_j \rangle < 0] \to \min_w .$$

Встает вопрос, на сколько богатый класс функций мы можем реализовать с помощью нейрона. Отдельно взятый нейрон вида (2) позволяет реализовать линейный классификатор или линейную регрессию. Или, например, простейшие логические функции И и ИЛИ, НЕ от бинарных переменных x^1 и x^2:

$$x^1 \wedge x^2 = \left[x^1 + x^2 - \frac{3}{2} > 0\right];$$
$$x^1 \vee x^2 = \left[x^1 + x^2 - \frac{1}{2} > 0\right];$$
$$\neg x^1 = \left[-x^1 - \frac{1}{2} > 0\right].$$

Однако при решении практических задач линейность оказывается чрезмерно сильным ограничением. Следующий контрпример иллюстрирует невозможность нейронной реализации простой логической функции $x^1 \oplus x^2 = [x^1 \neq x^2]$ — исключающего ИЛИ (XOR). Её принципиально невозможно реализовать одним нейроном с двумя входами x^1 и x^2, поскольку множества нулей и единиц этой функции линейно неразделимы. Возможны два пути решения этой проблемы, рис. 4.

Первый путь — пополнить состав признаков, подавая на вход нейрона нелинейные преобразования исходных признаков. В частности, если разрешить образовывать

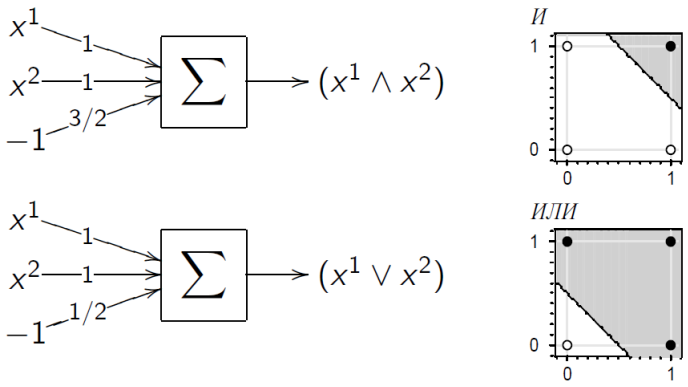


Рис. 3: Схема нейронов для И и ИЛИ

всевозможные произведения исходных признаков, то нейрон будет строить уже не линейную, а полиномиальную разделяющую поверхность. В случае исключающего ИЛИ достаточно добавить только один вход x^1x^2, чтобы в расширенном пространстве множества нулей и единиц оказались линейно разделимыми:

$$x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right].$$

Проблема заключается в том, что подбор нужных нелинейных преобразований является нетривиальной задачей, которая для общего случая до сих пор остаётся нерешённой.

Второй путь — построить композицию из нескольких нейронов. Например, с помощью композиции функций И, ИЛИ, НЕ:

$$x^1 \oplus x^2 = \left[\neg\left(x^1 \wedge x^2 - x^1 \vee x^2\right) > 0\right].$$

Схемы для двух вариантов представлены на рис. 4

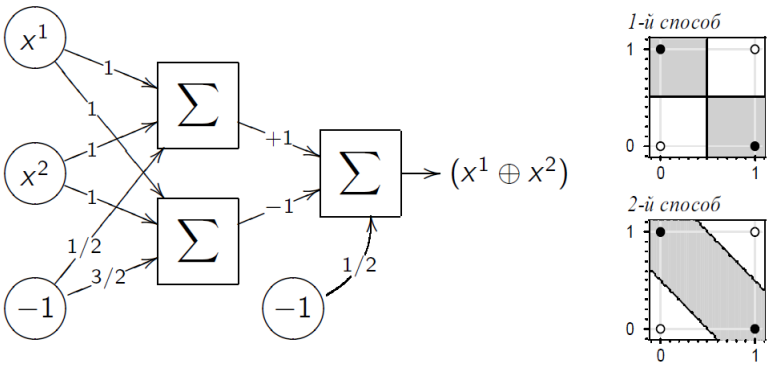


Рис. 4: Схемы нейронов для двух реализаций XOR

Дальнейшее обобщение этой идеи приводит к построению многослойных нейронных сетей, состоящих из огромного количества связанных нейронов и напоминающих естественные нейронные сети. Пример такой композиции показан на рис. 5. Значения всех p признаков одновременно подаются на вход всех H нейронов первого слоя. Затем их выходные значения подаются на вход всех M нейронов следующего слоя. В данном случае этот слой является выходным — такая сеть называется двухслойной. В общем случае сеть может содержать произвольное число слоёв. Все слои, за исключением последнего, называются *скрытыми* (hidden layers).

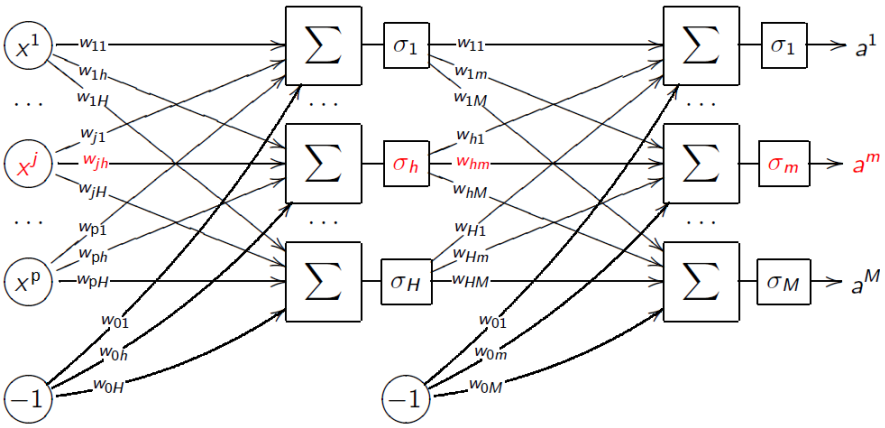


Рис. 5: Пример многослойной нейросети

Следующая теорема отвечает на вопрос, можно ли любую непрерывную функцию представить нейросетью.

Теорема 1 (Цыбенко, 1989)**.** Пусть $\sigma(x)$ — непостоянная, ограниченная и монотонно возрастающая непрерывная функция I_{p_0} — p_0-мерный единичный гиперкуб; $C(I_{p_0})$ — множество непрерывных функций на I_{p_0}. Тогда для любой $f \in C(I_{p_0})$ и $\varepsilon > 0$ существуют $p_1 \in \mathbb{Z}$ и α_i, b_i, $w_{ij} \in \mathbb{R}$, $i = 1, \ldots, p_1$, $j = 1, \ldots, p_0$, такие что функция

$$F(x^1, \ldots, x^{p_0}) = \sum_{i=1}^{p_1} \alpha_i \, \sigma \left(\sum_{j=1}^{p_0} w_{ij} x^j + b_i \right),$$

аппроксимирует функцию f с точностью ε:

$$|F(x^1, \ldots, x^{p_0}) - f(x^1, \ldots, x^{p_0})| < \varepsilon$$

для любого $x = (x^1, \ldots, x^{p_0}) \in I_{p_0}$.

Нейросеть, выход которой, соответствует F из теоремы: сеть с p_0 входными узлами, одним скрытым слоем с p_1 узлами и функцией активации $\sigma(z)$. Из теоремы следует, что

- любую непрерывную функцию можно приблизить нейросетью с любой заранее заданной точностью;
- для этой нейросети нужна одна нелинейная функция активации и один скрытый слой.

В качестве функции активации чаще всего используется одна из следующих функций:

- логистическая (сигмоидная) функция: $\sigma(z) = \frac{1}{1+e^{-a\,z}}$, $a \in \mathbb{R}$;
- гиперболический тангенс: $\sigma(z) = \frac{e^{a\,z} - e^{-a\,z}}{e^{a\,z} + e^{-a\,z}}$, $a \in \mathbb{R}$;
- rectifier: $f(z) = \max(0, z) \approx \ln(1 + e^z)$.

Последняя функция чаще всего используются для глубоких нейронных сетей с большим количеством слоёв.

4 Многослойные нейронные сети. Метод обратного распространения ошибок

Многослойная нейронная сеть, представленная на рис. 5, если число слоёв и нейронов достаточно большое, имеет очень много параметров, число которых равно $H(p+M+1)+M$; $w \equiv (w_{jh}, w_{hm}) = (\{w_{jh}\}_{j,h=0}^{p,H}, \{w_{hm}\}_{h,m=0}^{H,M}) \in \mathbb{R}^{H(p+M+1)+M}$ — вектор параметров. Тем не менее для настройки нейронных сетей тоже используют градиентные методы, в частности, стохастический градиентный спуск.

В середине 80-х одновременно несколькими исследователями был предложен эффективный способ вычисления градиента, при котором каждый градиентный шаг выполняется за число операций, лишь немногим большее, чем при обычном вычислении сети на одном объекте. Это кажется удивительным — потому что количество операций, необходимых для вычисления градиента, обычно возрастает пропорционально размерности, то есть числу весовых коэффициентов. Здесь этого удаётся избежать благодаря аналитическому дифференцированию суперпозиции с сохранением необходимых промежуточных величин. Метод получил название обратного распространения ошибок (error backpropagation).

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, рис. 5. Такая сеть называется *полносвязной*. Для большей общности положим $X = \mathbb{R}^p$, $Y = \mathbb{R}^M$. Для простоты рассматриваем двухслойную сеть.

Введём следующие обозначения. Пусть выходной слой состоит из M нейронов с функциями активации σ_m и выходами a^m, $m = 1, \ldots, M$. Перед ним находится скрытый слой из H нейронов с функциями активации σ_m и выходами u^h, $h = 1, \ldots, H$.

Веса связей между h-м нейроном скрытого слоя и m-м нейроном выходного слоя будем обозначать через w_{hm}. Перед этим слоем может находиться либо входной слой признаков, либо ещё один скрытый слой с выходами v_j, $j = 1, \ldots, J$ и весами w_{jh}. В общем случае число слоёв может быть произвольным. Если сеть двухслойная, то v_j есть просто j-й признак: $v_j(x) = f_j(x) = x_j$, и $J = p$, w — вектор всех весов сети.

Выходные значения сети на объекте x_i вычисляются как суперпозиция:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \quad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{p} w_{jh} f_j(x_i)\right). \tag{3}$$

Зафиксируем объект x_i и запишем функционал среднеквадратичной ошибки (для других функций потерь выкладки могут быть проделаны аналогично):

$$\mathscr{L}_i(w) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2.$$

В дальнейшем нам понадобятся частные производные Q по выходам нейронов. Выпишем их сначала для выходного слоя:

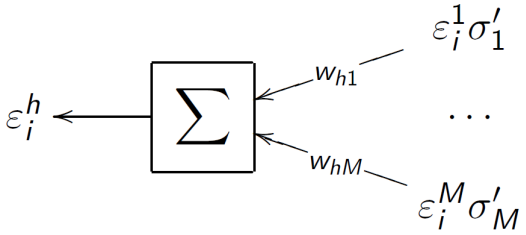
$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

Оказывается, частная производная Q по a^m равна величине ошибки ε_i^m на объекте x_i. Теперь выпишем частные производные по выходам скрытого слоя:

$$\frac{\partial \mathscr{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M}(a^m(x_i) - y_i^m)\sigma_m' w_{hm} = \sum_{m=1}^{M}\varepsilon_i^m \sigma_m' w_{hm} = \varepsilon_i^h.$$

Эту величину, по аналогии с ε_i^m, будем называть *ошибкой сети на скрытом слое* и обозначать через ε_i^h. Через $\sigma_m^{'}$ обозначена производная функции активации, вычисленная при том же значении аргумента, что и в (3). Если используется сигмоидная функция активации, то для эффективного вычисления производной можно воспользоваться формулой: $\sigma_m^{'} = \sigma_m(1-\sigma_m) = a^m(x_i)(1-a^m(x_i))$.

Заметим, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд», подав на выходы нейронов скрытого слоя значения $\varepsilon_i^m\sigma_m^{'}$, а результат ε_i^h получив на выходе. При этом входной вектор скалярно умножается на вектор весов w_{hm}, находящихся справа от нейрона, а не слева, как при прямом вычислении (отсюда и название алгоритма — *обратное распространение ошибки*):



Имея частные производные $\mathscr{L}_i(w)$ по a^m и u^h, легко выписать градиент $\mathscr{L}_i(w)$ по весам w:

$$\frac{\partial\mathscr{L}_i(w)}{\partial w_{hm}} = \frac{\partial\mathscr{L}_i(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m\sigma_m^{'}u^h(x_i), \quad h=0,\ldots,H, \quad m=1,\ldots,M;$$
$$\frac{\partial\mathscr{L}_i(w)}{\partial w_{jh}} = \frac{\partial\mathscr{L}_i(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h\sigma_h^{'}f_j(x_i), \quad j=0,\ldots,p, \quad h=1,\ldots,H;$$

и так далее для каждого слоя. Если слоёв больше двух, то остальные частные производные вычисляются аналогично — обратным ходом по слоям сети справа налево.

Теперь мы обладаем всем необходимым, чтобы полностью выписать алгоритм обратного распространения ошибки.

Вход: $X^n = (x_i, y_i)_{i=1}^n \subset \mathbb{R}^p \times \mathbb{R}^M$; параметры H, λ, η.
Выход: веса w_{jh}, w_{hm}.
1: инициализировать веса w_{jh}, w_{hm};
2: **повторять**
3: случайно выбрать элемент x_i из выборки X^n;
4: прямой ход:
$u_i^h := \sigma_h\left(\sum_{j=0}^p w_{jh}x_i^j\right),\ h=1,\ldots,H;$
$a_i^m := \sigma_m\left(\sum_{h=0}^H w_{hm}u_i^h\right),\ \varepsilon_i^m := a_i^m - y_{im},\ m=1,\ldots,M;$
$\mathscr{L}_i := \sum_{m=1}^M(\varepsilon_i^m)^2$, вычисление производных $\sigma_m^{'}, \sigma_h^{'}$;
5: обратный ход:
$\varepsilon_i^h := \sum_{m=1}^M \varepsilon_i^m\sigma^{'}w_{hm},\ h=1,\ldots,H;$
6: градиентный шаг:
$w_{hm} := w_{hm} - \eta\varepsilon_i^m\sigma_m^{'}u_i^h,\ h=0,\ldots,H,\ m=1,\ldots,M;$
$w_{jh} := w_{jh} - \eta\varepsilon_i^h\sigma_h^{'}x_i^j,\ j=0,\ldots,p,\ h=1,\ldots,H;$
7: $Q := (1-\lambda)Q + \lambda\mathscr{L}_i;$
8: **пока:** Q не сойдется.

Опишем достоинства метода обратного распространения ошибки.

- Достаточно высокая эффективность. В случае двухслойной сети прямой ход, обратный ход и вычисления градиента требуют порядка $O(Hp+HM)$ операций.

- Через каждый нейрон проходит информация только о связных с ним нейронах. Поэтому backpropagation легко реализуется на вычислительных устройствах с параллельной архитектурой.
- Высокая степень общности. Алгоритм легко записать для произвольного числа слоёв, произвольной размерности выходов и выходов, произвольной функции потерь и произвольных функций активации, возможно, различных у разных нейронов. Кроме того, backpropagation можно применять совместно с различными градиентными методами оптимизации: методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

Недостатки метода обратного распространения (есть все те же, что и у Стохастического градиента).

- Возможна медленная сходимость.
- Застревание в локальных минимумах.
- Проблема «паралича» сети (горизонтальные асимптоты σ).
- Проблема переобучения.
- Сложно подбирать эвристики.

5 Эвристики в методе обратного распространения ошибки для улучшения сходимости

Применимы все те же эвристики, что и в обычном методе стохастического градиента.

- Инициализация весов.
- Порядок предъявления объектов.
- Оптимизация величины градиентного шага.
- Регуляризация (сокращение весов).

И появляются новые эвристики для улучшения сходимости, специфичные для нейросети.

Выбор начального приближения Для предотвращения паралича веса должны инициализироваться небольшими по модулю значениями. Часто веса выбирают случайными значениями из отрезка $[-1/2k, 1/2k]$, где k — число нейронов в том слое, из которого выходит данная связь.

Существует и более тонкий способ формирования начального приближения. Идея заключается в том, чтобы сначала настроить нейроны первого слоя по-отдельности, как H однослойных нейронных сетей. Затем по-отдельности настраиваются нейроны второго слоя, которым на вход подаётся вектор выходных значений первого слоя. Чтобы сеть не получилась вырожденной, нейроны первого слоя должны быть существенно различными. Ещё лучше, если они будут хоть как-то приближать целевую зависимость, тогда второму слою останется только усреднить результаты первого слоя, сгладив ошибки некоторых нейронов. Добиться этого совсем несложно, если

обучать нейроны первого слоя на различных случайных подвыборках, либо подавать им на вход различные случайные подмножества признаков. Отметим, что при формировании начального приближения не требуется особая точность настройки, поэтому отдельные нейроны можно обучать простейшими градиентными методами.

Выбор градиентного метода оптимизации Градиентные методы первого порядка сходятся довольно медленно, и потому редко применяются на практике. Ньютоновские методы второго порядка также непрактичны, но по другой причине — они требуют вычисления матрицы вторых производных функционала $Q(w)$, имеющей слишком большой размер. Существуют следующие рекомендации.

1. Если обучающая выборка имеет большой объём (порядка нескольких сотен объектов и более), или если решается задача классификации, то можно использовать метод стохастического градиента с адаптивным шагом, где на каждом шаге ищем η_*:
$$\mathscr{L}_i(w - \nabla\mathscr{L}_i(w)) \to \min_{\eta}.$$
Это одномерная задача оптимизации, для неё можно использовать простейшие методы, нам не обязательно находить точный минимум.

2. Диагональный метод Левенберга–Марквардта. В метод Ньютона-Рафсона (второго порядка):
$$w := w - \eta(\mathscr{L}_i''(w))^{-1}\mathscr{L}_i'(w),$$
где $\mathscr{L}_i''(w) = \left(\frac{\partial^2\mathscr{L}_i(w)}{\partial w_{jh}\partial w_{j'h'}}\right)$ — гессиан размера $(H(p+M+1)+M)^2$.

   Метод Левенберга–Марквардта (и сама эвристика) состоит в том, что считаем, что гессиан диагонален:
$$w_{jh} := w_{jh} - \eta\left(\frac{\partial^2\mathscr{L}_i(w)}{\partial w_{jh}^2} + \mu\right)^{-1}\frac{\partial\mathscr{L}_i(w)}{\partial w_{jh}},$$
   где η — темп обучения, μ — параметр, предотвращающий обнуление элемента.

   Отношение η/μ есть темп обучения на ровных участках функционала $Q(w)$, где вторая производная обращается в нуль. Диагональный элемент матрицы вторых производных вычисляется с помощью специального варианта backpropagation, аналогично как с первой производной.

6 Построение нейросети

Выбор структуры сети, то есть числа слоёв, числа нейронов и числа связей для каждого нейрона, является одной из наиболее сложных проблемой. Существуют различные стратегии поиска оптимальной структуры сети: постепенное наращивание, построение заведомо слишком сложной сети с последующим упрощением, поочерёдное наращивание и упрощение. Проблема выбора структуры тесно связана с проблемами недообучения и переобучения. Слишком простые сети не способны адекватно моделировать целевые зависимости в реальных задачах. Слишком сложные сети имеют избыточное число свободных параметров, которые в процессе обучения настраиваются не только на восстановление целевой зависимости, но и на воспроизведение шума.

Выбор числа слоёв Если в конкретной задаче гипотеза о линейной разделимости классов выглядит правдоподобно, то можно ограничиться однослойной нейросетью. Двухслойные сети позволяют представлять извилистые нелинейные границы, и в большинстве случаев этого хватает. Трёхслойными сетями имеет смысл пользоваться для представления сложных многосвязных областей. Чем больше слоёв, тем более богатый класс функций реализует сеть, но тем хуже сходятся градиентные методы, и тем труднее её обучить.

Выбор числа нейронов в скрытом слое H Производят различными способами, но ни один из них не является лучшим.

1. Визуальный способ. Если граница классов (или кривая регрессии) слишком сглажена, значит, сеть переупрощена, и необходимо увеличивать число нейронов в скрытом слое. Если граница классов (или кривая регрессии) испытывает слишком резкие колебания, на тестовых данных наблюдаются большие выбросы, веса сети принимают большие по модулю значения, то сеть переусложнена, и скрытый слой следует сократить. Недостаток этого способа в том, что он подходит только для задач с низкой размерностью пространства (небольшим числом признаков).

2. Оптимизация H по внешнему критерию, например, по критерию скользящего контроля или средней ошибки на независимой контрольной выборке $Q(X^k)$. Зависимость внешних критериев от параметра сложности, каким является H, обычно имеет характерный оптимум. Недостаток этого способа в том, что он очень трудоемок, приходится много раз заново строить сеть при различных значениях параметра H, а в случае скользящего контроля — ещё и при различных разбиениях выборки на обучающую и контрольную части.

Динамическое добавление нейронов

1. Обучение при заведомо недостаточном числе нейронов $H \ll n$ до тех пор, пока ошибка не перестаёт убывать.

2. Добавление нового нейрона и инициализация его связей небольшими случайными весами или путем обучения

   - либо по случайной подвыборке $X' \subseteq X^n$;
   - либо по объектам с наибольшими значениями потерь;
   - либо по случайному подмножеству входов;
   - либо из различных случайных начальных приближений.

   Веса старых связей не меняются.

3. Снова итерации backpropagation.

Полезно наблюдать за внешним критерием. Например, прохождение $Q(X^k)$ через минимум — надежный критерий остановки.

Эмпирический опыт показывает: общее время обучения обычно лишь в 1.5–2 раза больше, чем если бы в сети сразу было нужное количество нейронов, при этом в первом случае происходит меньше переобучения.

Удаление избыточных связей Метод оптимального прореживания сети (optimal brain damage, OBD) удаляет те связи, к изменению которых функционал Q наименее чувствителен. Уменьшение числа весов снижает склонность сети к переобучению.

Метод OBD основан на предположении, что после стабилизации функционала ошибки Q вектор весов w находится в локальном минимуме, где функционал может быть аппроксимирован квадратичной формой:

$$Q(w+\delta) = Q(w) + \frac{1}{2}\delta^{\mathrm{T}} Q''(w)\delta + o(\|\delta\|^2),$$

где $Q''(w) = \frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}$ — гессиан размера $(H(p+M+1)+M)^2$.

Как и в диагональном методе Левенберга–Марквардта, предполагается, что диагональные элементы доминируют в гессиане, а остальными частными производными можно пренебречь, положив их равными нулю. Это предположение носит эвристический характер и вводится для того, чтобы избежать трудоёмкого вычисления всего гессиана.

Если гессиан диагонален, то

$$\delta^{\mathrm{T}} Q''(w)\delta = \sum_{j=0}^{p}\sum_{h=0}^{H}\delta_{jh}^2\frac{\partial^2 Q(w)}{\partial w_{jh}^2} + \sum_{h=0}^{H}\sum_{m=0}^{M}\delta_{hm}^2\frac{\partial^2 Q(w)}{\partial w_{hm}^2}.$$

Обнуление веса w_{jh} эквивалентно выполнению условия $w_{jh} + \delta_{jh} = 0$. Введём величину значимости (salience) связи, равную изменению функционала $Q(w)$ при обнулении веса: $S_{jh} = w_{jh}^2\frac{\partial^2 Q(w)}{\partial w_{jh}^2}$.

Эвристика OBD заключается в том, чтобы удалить из сети d синапсов, соответствующих наименьшим значениям S_{jh}. Здесь d — это ещё один параметр метода настройки. После удаления производится цикл итераций до следующей стабилизации функционала Q. При относительно небольших значениях d градиентный алгоритм довольно быстро находит новый локальный минимум Q. Процесс упрощения сети останавливается, когда внутренний критерий стабилизируется, либо когда заданный внешний критерий начинает возрастать.

Обнуление веса w_{jh} между входным и скрытым слоями означает, что h-й нейрон скрытого слоя не будет учитывать j-й признак. Тем самым происходит отбор информативных признаков для h-го нейрона скрытого слоя. Метод OBD легко приспособить и для настоящего отбора признаков. Вводится суммарная значимость признака $S_j = \sum_{h=1}^{H} = S_{jh}$, и из сети удаляется один или несколько признаков с наименьшим значением S_j.

Обнуление веса w_{hm} между скрытым и выходным слоями означает, что m-е выходное значение не зависит от h-го нейрона скрытого слоя. Если выход одномерный $(M = 1)$, то h-й нейрон можно удалить. В случае многомерного выхода для удаления нейронов скрытого слоя вычисляется суммарная значимость $S_h = \sum_{m=1}^{M} = S_{hm}$.

Для данного метода есть такие же эмпирические результаты, как и для предыдущего: такая постепенно построенная сеть меньше переобучается, чем сеть, сразу построенная, с нужным количеством связей.