Семинары по нейронным сетям

Евгений Соколов
sokolov.evg@gmail.com

29 апреля 2015 г.

1 Искусственные нейронные сети

Искусственная нейронная сеть — это общее название для целого класса моделей. Как правило, они представляют собой комбинацию нелинейных преобразований входных данных и могут восстанавливать сложные нелинейные зависимости. В последнее время все большую популярность приобретает *глубинное обучение (deep learning)*, которое заключается в обучении нейросетей с очень большим числом параметров. С помощью глубинных нейросетей успешно решаются различные задачи, связанные с распознаванием речи, компьютерным зрением, обработкой текстов и т.д.

§1.1 Метод обратного распространения ошибки

Мы будем вести речь об одном из самых распространенных типов нейросетей — многослойных нейронных сетях. Будем считать, что объекты принадлежат пространству $\mathbb{R}^d$, а ответы — пространству $\mathbb{Y}^m$. Как следует из названия, многослойная нейросеть состоит из L слоев. Входной слой нейросети состоит из d нейронов $v_1^0, \dots, v_d^0$, каждый из которых принимает значение, соответствующее одному из признаков объекта: $v_i^0(x) = x_i$. Последний, L-й слой, называется выходным, а слои с 1-го по $(L-1)$-й — скрытыми. Выходной слой состоит из m нейронов (столько же, сколько элементов в векторе ответов $y \in \mathbb{Y}$, а i-й скрытый слой состоит из n_i нейронов. Каждый нейрон суммирует с некоторыми весами выходы всех нейронов предыдущего слоя, а затем применяет к сумме функцию активации:

$$v_j^i = \sigma_i \left(\sum_{k=1}^{n_{i-1}} w_{kj}^i v_k^{i-1}(x) \right), \qquad i = 1, \dots, L; \quad j = 1, \dots, n_i.$$

Вообще говоря, каждый нейрон v_j^i может иметь собственную функцию активации σ_{ij}. Все дальнейшие выкладки могут быть легко обобщены на этот случай.

Чтобы задать нейросеть, нужно настроить ее веса $\{w_{kj}^i\}$. Будем делать это, оптимизируя среднеквадратичную ошибку:

$$Q(\mathbb{X}; w) = \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{m} \left(v_j^L(x_i) - y_{ij} \right)^2 \to \min. \tag{1.1}$$

Рассмотрим одно слагаемое функционала, соответствующее ошибке на одном объекте:

$$Q(x;w) = \frac{1}{2}\sum_{j=1}^{m}\left(v_j^L(x) - y_j\right)^2.$$

Для настройки весов нам понадобятся производные функционала по весам $\partial Q/\partial w_{kj}^i$. Попытка вычислить их «в лоб» приведет к крайне трудоемким выкладкам; более того, полученные выражения будут трудными для вычисления. Однако производные могут быть вычислены эффективно, если воспользоваться методом *обратного распространения ошибки*. Опишем его.

Найдем производную функционала по выходам последнего слоя $v_j^L(x)$:

$$\frac{\partial Q}{\partial v_j^L} = \frac{\partial}{\partial v_j^L}\sum_{s=1}^{m}\left(v_s^L(x) - y_s\right)^2 = v_j^L(x) - y_j = \varepsilon_j^L.$$

Теперь, пользуясь формулой дифференцирования сложной функции, мы можем найти производные по весам связей между предпоследним и последним слоями:

$$\frac{\partial Q}{\partial w_{kj}^L} = \frac{\partial Q}{\partial v_j^L}\frac{\partial v_j^L}{\partial w_{kj}^L} = \varepsilon_j^L \sigma_L'\left(\sum_{s=1}^{n_{L-1}} w_{sj}^L v_s^{L-1}(x)\right) v_k^{L-1}(x).$$

Перейдем к предпоследнему слою. Найдем производные функционала по его выходам:

$$\frac{\partial Q}{\partial v_j^{L-1}} = \sum_{t=1}^{m}\frac{\partial Q}{\partial v_t^L}\frac{\partial v_t^L}{\partial v_j^{L-1}} = \sum_{t=1}^{m}\varepsilon_t^L \sigma_L'\left(\sum_{s=1}^{n_{L-1}} w_{st}^L v_s^{L-1}(x)\right) w_{jt}^L = \varepsilon_j^{L-1}.$$

Найдем производные по весам связей между $(L-2)$-м и $(L-1)$-м слоями:

$$\frac{\partial Q}{\partial w_{kj}^{L-1}} = \frac{\partial Q}{\partial v_j^{L-1}}\frac{\partial v_j^{L-1}}{\partial w_{kj}^{L-1}} = \varepsilon_j^{L-1}\sigma_{L-1}'\left(\sum_{s=1}^{n_{L-2}} w_{sj}^{L-1} v_s^{L-2}(x)\right) v_k^{L-2}(x).$$

Рассмотрим, наконец, произвольный i-й слой, и найдем производные по весам связей между $(i-1)$-м и i-м слоями. Будем считать, что мы уже вычислили все производные, связанные со слоями с $(i+1)$-го до L-го. Найдем сначала производные функционала по выходам i-го слоя:

$$\frac{\partial Q}{\partial v_j^i} = \sum_{t=1}^{n_{i+1}}\frac{\partial Q}{\partial v_t^{i+1}}\frac{\partial v_t^{i+1}}{\partial v_j^i} = \sum_{t=1}^{n_{i+1}}\varepsilon_t^{i+1}\sigma_{i+1}'\left(\sum_{s=1}^{n_i} w_{st}^{i+1} v_s^i(x)\right) w_{jt}^i = \varepsilon_j^i.$$

Теперь мы можем вычислить производные по весам:

$$\frac{\partial Q}{\partial w_{kj}^i} = \frac{\partial Q}{\partial v_j^i}\frac{\partial v_j^i}{\partial w_{kj}^i} = \varepsilon_j^i \sigma_i'\left(\sum_{s=1}^{n_{i-1}} w_{sj}^i v_s^{i-1}(x)\right) v_k^{i-1}(x).$$

Итак, мы показали, что производные по всем весам многослойной нейросети могут быть вычислены последовательно, от последнего слоя к первому.

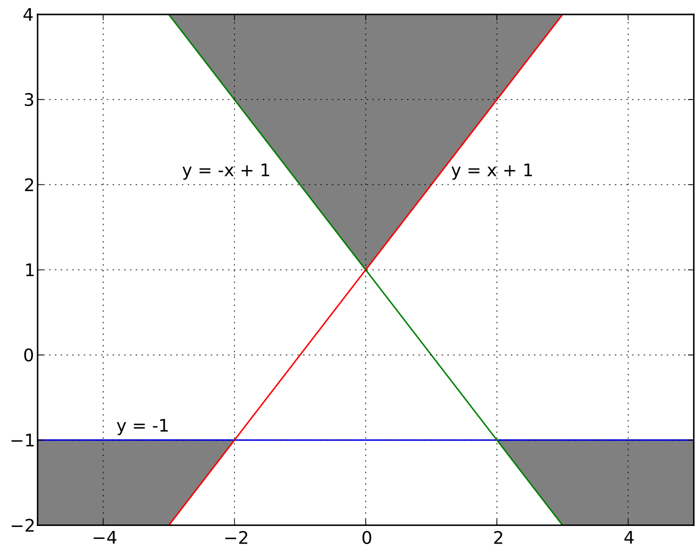

Рис. 1.

§1.2 Представление функций и разделяющих поверхностей

Задача 1.1. *Рассмотрим три прямые на плоскости:*

$$y_1 = -1,$$
$$y_2 = -x + 1,$$
$$y_3 = x + 1.$$

Постройте двухслойную нейронную сеть, которая будет выдавать ответ «−1» в областях, закрашенных на рис. 1, и ответ «+1» во всем остальном пространстве.

Решение. Выберем функцию активации $\sigma(x) = \operatorname{sign} x$. С помощью нейронов первого слоя реализуем разбиения на полуплоскости, которые производятся тремя прямыми из условия:

$$v_1^1 = \operatorname{sign}\left(y + 1\right),$$
$$v_2^1 = \operatorname{sign}\left(-x - y + 1\right),$$
$$v_3^1 = \operatorname{sign}\left(x - y + 1\right).$$

Заметим, что нейросеть должна выдавать ответ «−1» либо при $v_2^1 = v_3^1 = -1$, либо при $v_1^1 = -1, v_2^1 = -1, v_3^1 = +1$, либо при $v_1^1 = -1, v_2^1 = +1, v_3^1 = -1$.

Нетрудно убедиться, что требуемое разбиение будет достигаться, если задать нейрон второго слоя как

$$v_1^2 = \operatorname{sign}\left(3v_1^1 + 2v_2^1 + 2v_3^1\right).$$

■

Задача 1.2. *Реализуйте следующую булеву функцию с помощью одного нейрона:*

$$f(x) = x_1 \& \bar{x}_2 \& x_3.$$

Решение. Чтобы данная функция выдала единицу, необходимо, чтобы переменные приняли значения $x_1 = 1, x_2 = 0, x_3 = 1$. Легко видеть, что это равносильно выполнению равенства

$$x_1 + (1 - x_2) + x_3 = 3.$$

Значит, функцию можно реализовать с помощью следующего нейрона (с функцией активации $\sigma(x) = [x > 0]$:

$$v(x) = [x_1 - x_2 + x_3 - 1.5 > 0]\,.$$

■

Задача 1.3. *Реализуйте следующую булеву функцию с помощью двухслойной нейросети:*

$$f(x) = x_2 \& (x_1 \vee \bar{x}_3) \vee x_1 \& x_3.$$

Решение. Преобразуем сначала данную функцию, представив ее в виде ДНФ. Для этого раскроем скобки:

$$f(x) = x_1 \& x_2 \vee x_2 \& \bar{x}_3 \vee x_1 \& x_3.$$

С помощью первого слоя нейросети реализуем все конъюнкции. Мы уже умеем это делать:

$$a \& b = [a + b - 1.5 > 0].$$

Значит

$$v_1^1(x) = [x_1 + x_2 - 1.5 > 0];$$
$$v_2^1(x) = [x_2 - x_3 - 0.5 > 0];$$
$$v_3^1(x) = [x_1 + x_3 - 1.5 > 0].$$

С помощью нейрона второго слоя нужно реализовать дизъюнкцию трех переменных. Это тоже легко сделать:

$$v_1^2(x) = [v_1^1 + v_2^1 + v_3^1 - 0.5 > 0].$$

■

§1.3 Обучение глубоких сетей

Классический алгоритм обратного распространения ошибки хорошо работает на двухслойных и трехслойных нейронных сетях, но при дальнейшем увеличении глубины начинает испытывать проблемы. Одна из причин — так называемое затухание градиентов. По мере распространения ошибки от выходного слоя к входному на каждом слое происходит домножение текущего результата на производную функции активации. Производная у традиционной сигмоидной функции активации меньше единицы на всей области определения, поэтому после нескольких слоев ошибка станет близкой к нулю. Если же, наоборот, функция активации имеет неограниченную производную (как, например, гиперболический тангенс), то может произойти взрывное увеличение ошибки по мере распространения, что приведет к неустойчивости процедуры обучения.

В последние годы были достигнуты существенные успехи в области методов настройки глубоких нейросетей. В данном разделе мы рассмотрим некоторые наиболее интересные приемы.

ReLU. Известно, что нейронные сети способны приблизить сколь угодно сложную функцию, если в них достаточно слоев и функция активации является нелинейной. Функции активации вроде сигмоидной или тангенциальной являются нелинейными, но приводят к проблемам с затуханием или увеличением градиентов. Однако можно использовать и гораздо более простой вариант — *выпрямленную линейную функцию активации* (rectified linear unit, ReLU):

$$\sigma(x) = \max(0, x).$$

Ее производная равна либо единице, либо нулю, и поэтому не может произойти разрастания или затухания градиентов. Более того, использование данной функции приводит к прореживанию весов.

Dropout. Глубокие нейронные сети сильно подвержены переобучению из-за большого числа параметров. Одним из способов борьбы с ним является dropout-регуляризация. Обучение нейронной сети обычно производят стохастическим градиентным спуском, случайно выбирая по одному объекту из выборки. Dropout-регуляризация заключается в том, что при выборе очередного объекта изменяется структура сети: каждая вершина выбрасывается с некоторой вероятностью p. По такой прореженной сети делается обратное распространение ошибки, для оставшихся весов делается градиентный шаг, после чего все выброшенные вершины возвращаются в нейросеть. Таким образом, на каждом шаге стохастического градиента мы настраиваем одну из возможных 2^N архитектур сети, где под архитектурой мы понимаем структуру связей между вершинами, а через N обозначаем суммарное число вершин. При применении нейросети вершины уже не выбрасываются, но выход каждой вершины домножается на $(1 - p)$ — благодаря этому на выходе вершины мы будем получать матожидание ее ответа по всем 2^N архитектурам. Таким образом, обученную с помощью dropout-регуляризации нейросеть можно рассматривать как результат усреднения 2^N сетей.

Extreme learning machines. Было замечено, что если настраивать только веса выходного слоя нейросети, а все остальные веса выставлять случайно, то при достаточном увеличении числа вершин и слоев может быть достигнуто качество, сравнимое с полностью настраиваемыми весами. Двуслойные нейронные сети, основанные на этой идее, носят название *extreme learning machines* (ELM) [1]. Рассмотрим функцию активации

$$\sigma(x; a, b) = g\left(\frac{\|x - a\|}{b}\right),$$

параметрами которой являются вектор $a \in \mathbb{R}^d$ и число $b \in \mathbb{R}$. Функция g должна быть ограниченной, неконстантной и кусочно-непрерывной. Саму нейросеть с n скрытыми нейронами при этом определим как

$$a_n(x) = \sum_{i=1}^{n} w_i \sigma(x; a_i, b_i).$$

Можно показать, что последовательность нейросетей $\{a_n\}$ со случайно генерируемыми параметрами a_i и b_i в пределе будет аппроксимировать любую непрерывную функцию при правильной настройке выходных весов w_i. Таким образом, процесс настройки такой нейросети выглядит следующим образом: добавляем новый скрытый нейрон со случайными параметрами a_i и b_i, настраиваем w_i, фиксируем его, добавляем еще один нейрон, и так далее до сходимости. Процедура обучения крайне быстрая, и при этом результирующий алгоритм может иметь очень хорошее качество. Также ELM могут выступать в качестве базовых алгоритмов в бустинге.

Список литературы

[1] *Huang G., Chen L.* (2008). Enhanced random search based incremental extreme learning machine. // Neurocomputing, Vol. 71, 16–18, Pp. 3460–3468.