Семинары по нейронным сетям

Евгений Соколов
sokolov.evg@gmail.com

1 марта 2014 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 $v_1^0, \ldots, v_d^0$, 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. Скрытый 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:

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

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.

§1.2 Градиентные методы оптимизации

Теперь, когда мы умеем вычислять частные производные функционала по параметрам нейросети, можно воспользоваться любым методом оптимизации первого порядка. В данном разделе мы рассмотрим некоторые градиентные методы и гарантии их сходимости, которые могут быть даны для выпуклых функций.

Пусть $Q(w)$ — функционал, представимый в виде суммы n функций:

$$Q(w) = \sum_{i=1}^{n} q_i(w).$$

В таком виде, например, может быть представлен квадратичный функционал для нейросети (1.1). Отдельные функции $q_i(w)$ будут соответствовать ошибкам на отдельных объектах.

Наиболее известным является метод *градиентного спуска* (full gradient, FG) [1]:

$$w^k = w^{k-1} - \alpha_k \nabla Q(w^{k-1}).$$

Если функционал $Q(w)$ выпуклый, гладкий и имеет минимум w^*, то имеет место следующая оценка сходимости:

$$Q(w^k) - Q(w^*) = O(1/k).$$

Если функционал состоит из большого числа слагаемых (т.е. n велико), то градиентный спуск может оказаться слишком трудоемким. В этих случаях можно воспользоваться методом *стохастического градиента* (stochastic gradient) [2]:

$$w^k = w^{k-1} - \alpha_k \nabla q_{i_k}(w^{k-1}),$$

где i_k — случайно выбранный номер слагаемого из функционала. Для выпуклого и гладкого функционала может быть получена следующая оценка:

$$\mathbb{E}\left[Q(w^k) - Q(w^*)\right] = O(1/\sqrt{k}).$$

Таким образом, метод стохастического градиента имеет менее трудоемкие итерации по сравнению с полным градиентом, но и скорость сходимости у него существенно меньше.

Недавно был предложен метод *стохастического градиента* (stochastic average gradient) [3], который сочетает в себе быстроту итераций стохастического градиента и высокую скорость сходимости полного градиента. Перед началом итераций в нем выбирается начальное приближение w^0, и инициализируются вспомогательные переменные y_i^0, соответствующие градиентам слагаемых функционала:

$$y_i^0 = \nabla q_i(w^0), \qquad i = 1, \ldots, n.$$

На k-й итерации выбирается случайное слагаемое i_k и обновляются вспомогательные переменные:

$$y_i^k = \begin{cases} \nabla q_i(w^{k-1}), & \text{если } i = i_k; \\ y_i^{k-1} & \text{иначе.} \end{cases}$$

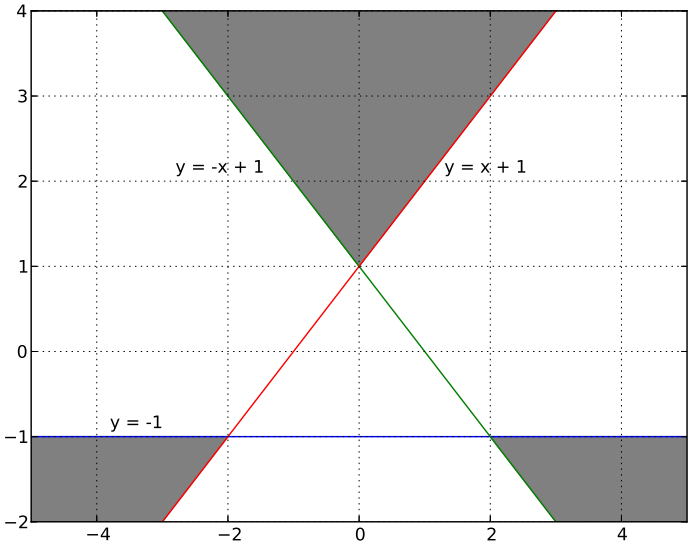

Рис. 1.

Иными словами, пересчитывается один из градиентов слагаемых. Наконец, делается градиентный шаг:

$$w^k = w^{k-1} - \alpha_k \sum_{i=1}^{n} y_i^k.$$

Данный метод имеет следующий порядок сходимости для выпуклых и гладких функционалов:

$$\mathbb{E}\left[Q(w^k) - Q(w^*)\right] = O(1/k).$$

На практике для настройки нейросетей обычно используют стохастические методы из-за их скорости итераций.

§1.3 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:

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

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

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

■

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:

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

■

Список литературы

[1] *Cauchy, M. A.* (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. // Comptes rendus hebdomadaires des séances de l'Académie des sciences, 25, p. 536-538.

[2] *Robbins, H., Monro S.* (1951). A stochastic approximation method. // Annals of Mathematical Statistics, 22 (3), p. 400-407.

[3] *Schmidt, M., Le Roux, N., Bach, F.* (2013). Minimizing finite sums with the stochastic average gradient. // Arxiv.org.