Sieci neuronowe i aplikacje sztucznej inteligencji — zadanie aproksymacji funkcji (estymacji funkcji regresji)

6 października 2006

1 Zadanie

Dana jest funkcja dwóch zmiennych

$$f(x_1, x_2) = \cos(x_1 x_2)\cos(2x_1)$$

określona na dziedzinie $[0, \pi] \times [0, \pi]$. Należy wykonać aproksymację tej funkcji na podstawie wygenerowanego zbioru uczącego za pomocą sieci neuronowej typu MLP (ang. *multi-layer perceptron*) o architekturze podanej w dalszej części.

2 Zbiory uczące

Należy wygenerować kilka zbiorów uczących — tj. zbiorów par wejście/wyjście postaci $\left\{(\mathbf{x}_i, y_i^*)\right\}_{i=1,\ldots,I}$, gdzie $\mathbf{x}_i = (x_{i1}, x_{i2})$.

Będą potrzebne dwa zbiory bez szumu o rozmiarach $I = 100$ oraz $I = 1000$, oraz dwa zbiory z szumem o tych samych rozmiarach. Punkty wejściowe $\mathbf{x}_i = (x_{i1}, x_{i2})$ są zmiennymi losowymi z rozkładu jednostajnego (inaczej: równomiernego), tj. :

$$x_{i1} \sim U(0, \pi), \quad x_{i2} \sim U(0, \pi).$$

Nie chodzi tu o regularną siatkę punktów.

W przypadku zbiorów bez szumu wartości y_i^* są po prostu odczytywane ze wzoru funkcji. Natomiast w przypadku zbiorów z szumem, wartości y_i^* niech podlegają zaszumieniu o rozkładzie normalnym, tj.:

$$y_i^* = f(x_{i1}, x_{i2}) + \varepsilon, \tag{1}$$

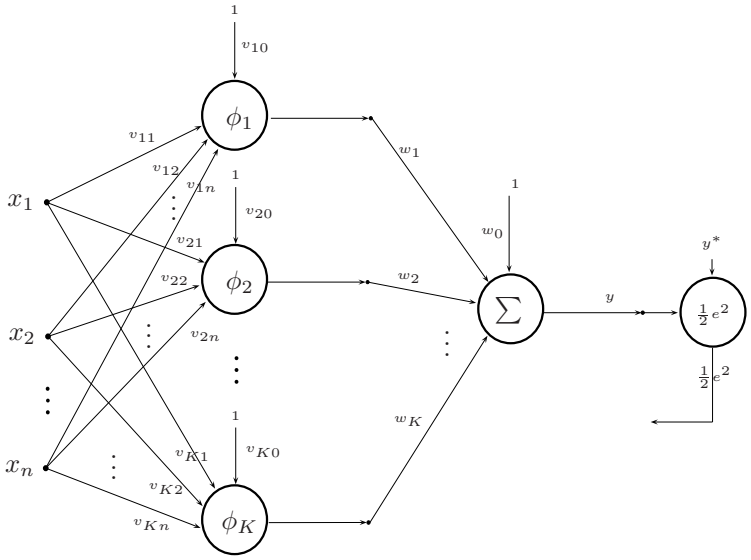
gdzie ε jest zmienną losową czerpaną z rozkładu normalnego o wartości średniej równej zeru, i odchyleniu standardowym równym 0.1 rozpiętości przeciwdziedziny funkcji f, tj. :

$$\varepsilon \sim N(0, 0.1 \cdot 2).$$

W większości języków programowania (w szczególności w MATLABie) powinna być dostępna funkcja do czerpania liczb losowych z rozkładu normalnego.

3 Architektura sieci

Sieć dwuwarstwowa z pierwszą warstwą nieliniową i drugą warstwą (neuron wyjściowy) liniową (w naszym zadaniu $n = 2$):



Wzór sygnału wyjściowego sieci:

$$y(x_1, \ldots, x_n) = w_0 + \sum_{k=1}^{K} w_k \phi_k(s_k),$$

$$\phi(s_k) = \frac{1}{1 + \exp(-s_k)},$$

$$s_k = v_{k0} + \sum_{j=1}^{n} v_{kj} x_j.$$

W programie taką sieć wystaczy reprezentować sobie jako macierz wag v_{kj} warstwy nieliniowej oraz wektor wag w_k bez potrzeby implementowania jakiegokolwiek grafu połączeń.

4 Uczenie sieci

Proponowane jest klasyczne uczenie typu *back-propagation* w trybie *online*, tj. korygowanie wag następuje każdorazowo po wykonaniu przebiegu sieci dla pojedynczej próbki (nie zaś dopiero po obejrzeniu przez sieć całego zbioru). Niech g reprezentuje dowolny parametr (wagę) sieci. Wówczas wzór na poprawkę g jest następujący:

$$g_{\mathsf{nowe}} = g_{\mathsf{stare}} + \eta(-\frac{\partial \frac{1}{2} e^2}{\partial g_{\mathsf{stare}}}),$$

$$\frac{1}{2} e^2 = \frac{1}{2}(y - y^*)^2,$$

gdzie η jest współczynnikiem uczenia, $\eta \in (0, 1]$. Wzory na poprawki gradientowe względem każdego z parametrów podane na zajęciach.

5 Sugerowane ustawienia

Sugeruje się eksperymentować na małych wartościach η — np. od około 0.01 do 0.1. Liczba K neuronów w warstwie nieliniowej niech będzie wybierana z przedziału $8 \leqslant K \leqslant 64$. Sugeruje się także zainicjalizować wagi bardzo małymi wartościami, powiedzmy z przedziału $[-0.01, 0.01]$. Przy powyższych ustawieniach liczba kroków uczenia rzędu od 50000 do $5 \cdot 10^6$.

6 Sprawdzenie jakości uogólniania po nauczeniu

Istotnym zadaniem sieci jest nie tylko umiejętność odtwarzania z małym błędem poznanego zbioru, ale tak naprawdę umiejętność uogólniania — tj. dostarczania dobrych przybliżeń dla punktów spoza zbioru uczącego. Prostym sposobem zgrubnego sprawdzenia tego jest narysowanie sobie wykresów powierzchni aproksymowanej funkcji oraz powierzchni, jaką generuje sieć neuronowa i porównanie na ile powierzchnie te są podobne (wykresy te mają znaleźć się w programie).

W naszym zadaniu chcąc policzyć w sposób dokładny jakość uogólniania należałoby policzyć całkę typu:

$$\frac{\int_0^\pi \int_0^\pi |y(x_1, x_2) - f(x_1, x_2)| dx_1 dx_2}{\int_0^\pi \int_0^\pi 1 dx_1 dx_2} = \frac{\int_0^\pi \int_0^\pi |y(x_1, x_2) - f(x_1, x_2)| dx_1 dx_2}{\pi^2},$$

czyli średni błąd bezwzględny (odległość pomiędzy powierzchniami) przypadający na każdą jednostką dziedziny.

Kto da radę, to może taką całkę policzyć (może być numerycznie). W przeciwnym wypadku można sobie oszacować tę wartość generując nowy zbiór — zbiór testujący i obliczając średni błąd popełniany na nim przez sieć.

W praktycznych zadaniach postępuje się zwykle tak, że zadany zbiór danych dzieli się na zbiór uczący i zbiór testujący, np. w proporcjach 70% i 30%. Jednakże w naszym zadaniu znamy w sposób jawny aproksymowaną funkcję, więc nic nie stoi na przeszkodzie, żeby wygenerować duży zbiór testujący (może być nawet wielokrotnie większy niż zbiór uczący).