Backpropagation for a network with polynomial layers

Olivier Roche

November 13, 2018

1 The context : forward pass

We have a neural network with $L+1$ layers. Given an input $\mathbf{a}^l$ for a layer l, the corresponding output $\mathbf{a}^{l+1}$ for the layer $l+1$ is computed as follow (say that the layers l and $l+1$ have size m and n respectively) :

- Compute $\mathbf{z}^{l+1} := \mathbf{P}^l(\mathbf{a}^l)$ where $\mathbf{P}^l : \mathbb{R}^m \mapsto \mathbb{R}^n$ is a componentwise polynomial function $\mathbf{P}^l = (P_0^l, \dots, P_{m-1}^l)$.
- Set $\mathbf{a}^{l+1} = \sigma(\mathbf{z}^{l+1})$ where $\sigma^l : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a threshold function (eg componentwise sigmoid, componentwise ReLU, softmax...), which is required to be differentiable.

To sum it up, the neural network we're dealing with is very similar to a fully connected classical neural network, except that the transition between two layers is no longer required to be affine (as it is the case when one works with weights and biases). Rather, the transition is polynomial.

2 Training the network : backpropagation

Say we have an input data $\mathbf{x}$ with expected outcome $\mathbf{y}$, and we got output $\mathbf{a}^L$. We aim to use gradient descent to minimize a given cost function $C = C(\mathbf{y}, \mathbf{a})$. It is assumed that C is differentiable.

Convention From now on, we omit $\mathbf{y}$ and write $C = C(\mathbf{a})$.
In the sequel, if we are on layer l, we wish to see C as a function $C = C^l(\mathbf{a}^l)$.

We can write this explicitely : $C^l = C \circ (\sigma^l \circ \mathbf{P}^l) \circ \cdots \circ (\sigma^{L-1} \circ \mathbf{P}^{L-1})$. Hence :

$$\boxed{C^l = C^{l+1} \circ (\sigma^l \circ \mathbf{P}^l)}$$

Notice that $C^L = C$.

Since the superscript l in C^l is redundant, we omit it and write $\nabla_{\mathbf{a}^l} C$ instead of $\nabla_{\mathbf{a}^l} C^l$.

2.1 The gradient $\nabla_{\mathbf{a}^L} C$

Say $\mathbf{a}^L = (a_0^L, \ldots, a_n^L)$, then

$$\boxed{\nabla_{\mathbf{a}^L} C = \big(\frac{\partial C}{\partial a_1^L}(\mathbf{a}^L), \ldots, \frac{\partial C}{\partial a_n^L}(\mathbf{a}^L)\big)}$$

2.2 Backpropagating the gradient

Using the identity $C^l = C^{l+1} \circ (\sigma^l \circ \mathbf{P}^l)$ and the chain rule for gradient, we get :

$$\boxed{\nabla_{\mathbf{a}^l} C = \big(\mathrm{D}_{\mathbf{z}^{l+1}} \sigma^l \circ \mathrm{D}_{\mathbf{a}^l} \mathbf{P}^l\big)^T (\nabla_{\mathbf{a}^{l+1}} C)}$$

Hence, we can easily compute the gradient $\nabla_{\mathbf{a}^l} C$ for $O \leqslant l \leqslant L$.

2.3 Altering the coefficients of the polynomials $\mathbf{P}^l$

Say the layers l and $l+1$ have n and m neurons respectively. Consider the polynomial P_i^l, where $0 \leqslant i < m$. Write $P_i^l = \sum w_{M,i} M$ where the sum ranges over all monomials[1] of bounded degree in the variables $X_0, \ldots X_{n-1}$. Let N be a monomial occuring in P^l.

Being given the activation $\mathbf{a}^l$, we see the cost function C as a function of the parameter $\mathbf{w}_N := (w_{N,i})_{i<m}$.

Puting $g_N(\mathbf{w}_N) = P^l(\mathbf{a}^l)$, one gets $C(\mathbf{w}_N) = C^{l+1}\big(\sigma^l(g_N(\mathbf{w}_N))\big)$, hence

$$\nabla_{\mathbf{w}_N} C = \big(\mathrm{D}_{z^{l+1}} \sigma^l \circ \mathrm{D}_{\mathbf{w}_N} g_N\big)^T (\nabla_{\mathbf{a}^{l+1}} C)$$

Since $\mathrm{D}_{\mathbf{w}_N} g_N = N(\mathbf{a}^l)\,\mathrm{Id}_m$, we get :

$$\boxed{\nabla_{\mathbf{w}_N} C = \big(N(\mathbf{a}^l) \cdot \mathrm{D}_{z^{l+1}} \sigma^l\big)^T (\nabla_{\mathbf{a}^{l+1}} C)}$$

[1] We consider 1 to be a monomial, the constant term of P_i^l is hence $w_{1,i}$.