General Backpropagation

José Eduardo Ochoa Luna
Dr. Ciencias - Universidade de São Paulo

Maestría C.C. Universidad Católica San Pablo
Sistemas Inteligentes

6 de diciembre 2017

Forward Propagation (Andrew Ng)

Given input x, we define $a^{[0]} = x$. Then for layer $l = 1, 2, \ldots N$, where N is the number of layers of NN, we have:

- $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$

Forward Propagation (Andrew Ng)

Given input x, we define $a^{[0]} = x$. Then for layer $l = 1, 2, \ldots N$, where N is the number of layers of NN, we have:

- $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$
- $a^{[l]} = g^{[l]}(z^{[l]})$

Nonlinearities

- We assume nonlinearities $g^{[l]}$ are the same for all layers besides layer N. Why?

Nonlinearities

- We assume nonlinearities $g^{[l]}$ are the same for all layers besides layer N. Why?
- In the output layer we may be doing regression (hence $g(x) = x$)

Nonlinearities

- We assume nonlinearities $g^{[l]}$ are the same for all layers besides layer N. Why?
- In the output layer we may be doing regression (hence $g(x) = x$)
- or binary classification ($g(x) = \mathit{sigmoid}(x)$)

Nonlinearities

- We assume nonlinearities $g^{[l]}$ are the same for all layers besides layer N. Why?
- In the output layer we may be doing regression (hence $g(x) = x$)
- or binary classification ($g(x) = sigmoid(x)$)
- or Multiclass classification ($g(x) = softmax(x)$)

Loss function

Given the output of the NN $a^{[N]}$, also denoted as $\hat{y}$, we measure the loss $J(W,b) = \mathcal{L}(a^{[N]}, y) = \mathcal{L}(\hat{y}, y)$:

- For real valued regression $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

Loss function

Given the output of the NN $a^{[N]}$, also denoted as $\hat{y}$, we measure the loss $J(W, b) = \mathcal{L}(a^{[N]}, y) = \mathcal{L}(\hat{y}, y)$:

- For real valued regression $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- For binary classification using log. regression $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$

Loss function

Given the output of the NN $a^{[N]}$, also denoted as $\hat{y}$, we measure the loss $J(W, b) = \mathcal{L}(a^{[N]}, y) = \mathcal{L}(\hat{y}, y)$:

- For real valued regression $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- For binary classification using log. regression $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$
- For softmax regression over k classes, we use cross entropy $\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{k} \mathbf{1}\{y = j\} \log \hat{y}_j$. Where $\hat{y}$ is a k-dimensional vector.

Loss function

Given the output of the NN $a^{[N]}$, also denoted as $\hat{y}$, we measure the loss $J(W, b) = \mathcal{L}(a^{[N]}, y) = \mathcal{L}(\hat{y}, y)$:

- For real valued regression $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- For binary classification using log. regression $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$
- For softmax regression over k classes, we use cross entropy $\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{k} \mathbf{1}\{y = j\} \log \hat{y}_j$. Where $\hat{y}$ is a k-dimensional vector.
- If we use y to instead denote the k dimensional vector of zeros with a single 1 at lth position, cross entropy can also be expressed as $\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{k} y_j \log \hat{y}$

Backpropagation

Let us define

$$\delta^{[l]} = \nabla_{z^{[l]}} \mathcal{L}(\hat{y}, y)$$

We can define a three step recipe for computing the gradients with respect to every $W^{[l]}, b^{[l]}$

Step 1

For output layer N, we have

$$\delta^{[N]} = \nabla_{z^{[N]}} \mathcal{L}(\hat{y}, y)$$

Sometimes we may compute this term directly (e.g $g^{[N]}$ is the softmax funct.), whereas other times ($g^{[N]}$ is sigmoid) we can apply the chain rule:

$$\nabla_{z^{[N]}} \mathcal{L}(\hat{y}, y) = \nabla_{\hat{y}} \mathcal{L}(\hat{y}, y) o (g^{N})'(z^{[N]})$$

where, $(g^{N})'(z^{[N]})$ denotes the element wise derivative w.r.t. z^{N}

Step 2

For $l = N-1, N-2, \ldots, 1$, we have

$$\delta^{[l]} = (W^{[l+1]T}\delta^{[l+1]}) o g'(z^{[l]})$$

where o, denotes the elementwise product.

Step 3

Finally, we can compute the gradients for layer l as

$$\nabla_{W^{[l]}} J(W, b) = \delta^{[l]} a^{[l-1]T}$$

$$\nabla_{b^{[l]}} J(W, b) = \delta^{[l]}$$