THE DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF FLORIDA

The Deep Neural Networks: Formulation and Implementation

Dihong Gong
May 11, 2015

1 Model

We model the deep neural networks as a stack of single-layer feed-forward neural networks, where each layer is modelled a function $f_\theta(x) : R^D \rightarrow R^d$ transforming a set of feature maps $X = \{X_1 \dots X_m\}$ into another set of feature maps $Y = \{Y_1 \dots Y_n\}$. The dimension of input feature maps is $h_X \times w_X$ and the dimension of the output feature maps is $h_Y \times w_Y$. This model can construct a wide variety of feed-forward networks, such as Multiple Layer Perceptrons (MLP), GoogleNet, Deep Boltzmann Machines (DBM) and Deep Autoencoders. The figure 1.1 shows a mapping example.

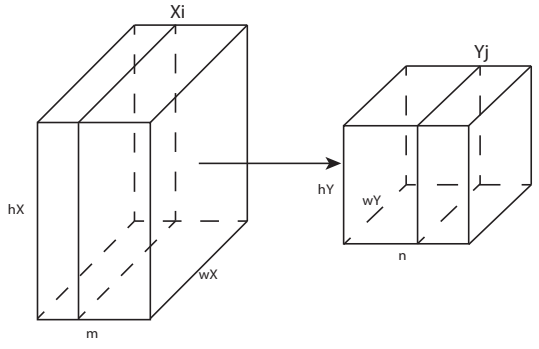

Figure 1.1: An illustration for a functional unit.

2 Formulations

The model in section 1 consists of a series of connecting blocks mapping one set of feature maps to another. Different networks have different connecting blocks that are connected in different manners. The blocks, which we refer as "functional units", represent a wide variety of operations such as convolution, response normalization, dropout, etc.

The constructed networks are trained with gradient-based algorithms such as stochastic gradient descend, using back propagation. The back propagation consists of two phases: forward signal passing and backward error propagation. Thus, any functional units $f_\theta(x) : R^D \rightarrow R^d$ can be identified by triple $\left(f_\theta(x), \frac{\partial f_\theta(x)}{\partial \theta}, \frac{\partial f_\theta(x)}{\partial x}\right)$. In this section, we will formulate triples for different types of functional units.

2.1 Input

The Input unit admits training samples, which initiates the feature maps.

Notations	Values	Dimension	Comments
Input	X	$R^{M\times h_X \times w_X}$	The input training signals
Parameter	$mean(X)$	$R^{M\times h_X \times w_X}$	Mean-center normalization

2.2 Output

The output unit calculates the cost of the input based on supervision signals.

2.2.1 softmax

Notations	Values	Dimension	Comments
Input	X and labels T	$X \in R^{D\times 1\times 1}$, $T \in R^{D\times 1\times 1}$	$T_d = 1$ if the sample is of class d D is equal to number of classes
$E = f(x)$	$-\sum_{d=1}^{D} T_d log P_d$	scalar	$P_d = \frac{e^{X_d}}{\sum_{j=1}^{D} e^{X_j}}$
$\frac{\partial E}{\partial \theta}$	N/A	N/A	No parameters
$\frac{\partial E}{\partial X_d}$	$P_d - T_d$	$R^{1\times 1\times D}$	$d = 1 \ldots D$

2.3 Full

The fully-connected unit transmits input signals via densely connected networks.

Notations	Values	Dimension	Comments
Input	X	$X \in R^{M \times h_X \times w_X}$	M is the number of feature maps. $h_X \times w_X$ is dimension of feature maps.
Parameters	W, b	$W \in R^{Q \times P}$ $b \in R^{Q \times 1}$	$P = M \times h_X \times w_X$ is # input variables. Q is # output variables. Output is of size $R^{Q \times 1 \times 1}$.
$Y = f(x)$	$h(W\,vec(X) + b)$	$Q \times 1$	$vec(\cdot)$ is a vectorization operator. Vectorization is in row-major manner.
$\frac{\partial E}{\partial Y} \cdot \frac{\partial f(x)}{\partial b_q}$	$\frac{\partial E}{\partial Y_q} h^{'}(x)$	$Q \times 1$	$q = 1 \dots Q$ Tanh: $h(x) = tanh(x), h^{'}(x) = 1 - Y_q^2$ ReLU: $h(x) = max(0,x), h^{'}(x) = \delta(Y_q > 0)$
$\frac{\partial E}{\partial Y} \cdot \frac{\partial f(x)}{\partial W_{qp}}$	$\frac{\partial E}{\partial Y} \cdot \frac{\partial f(x)}{\partial b_q} \odot vec(X)$	$Q \times P$	Operator $\odot$ represents outer product $vec(X) \in R^{1 \times P}$
$\frac{\partial E}{\partial Y} \cdot \frac{\partial f(x)}{\partial X}$	$\left(\frac{\partial E}{\partial Y} \cdot \frac{\partial f(x)}{\partial b_q}\right)^T \cdot W$	$1 \times P$	$d = 1 \dots D$

2.4 Conv

Suppose the filter size is $w \times h$, then:

$$Y_j = b_j + \sum_{i=1}^{m} X_i * W_{i,j} \tag{2.1}$$

$$\frac{\partial E}{\partial X_i} = \sum_{j=1}^{n} \frac{\partial E}{\partial Y_j} * W_{i,j} \tag{2.2}$$

$$\frac{\partial E}{\partial b_j} = \sum_{(x,y)} \frac{\partial E}{\partial Y_j^{(x,y)}} \tag{2.3}$$

$$\frac{\partial E}{\partial W_{ij}^{(p,q)}} = \sum_{y=0}^{h_Y} \sum_{x=0}^{w_Y} \frac{\partial E}{\partial Y_j^{(x,y)}} X_i^{(p+x-\frac{w}{2}, q+y-\frac{h}{2})} \tag{2.4}$$

Where Y_j is the j^{th} output feature map. The b_j is the bias parameter, and $W_{i,j}$ is a filter of size $h \times w$. The operator $*$ means convolution operation. The equation 2.4 is equivalent to pad X_i with zeros($h/2$ and $w/2$ on each edges), and run a convolution: $\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial Y_j^{(x,y)}} * X_i^{'}$.

$$\frac{\partial Y_j^{(x,y)}}{\partial W_j^{(p,q)}} = \sum_{i=1}^{m} S(i,j) X_i^{(p+c(x)-\frac{w}{2}, q+c(y)-\frac{h}{2})} \tag{2.5}$$

$$\frac{\partial Y_j^{(x,y)}}{\partial b_j} = 1 \tag{2.6}$$

$$\frac{\partial E}{\partial W_j^{(p,q)}} = \sum_{y=0}^{h_Y} \sum_{x=0}^{w_Y} \frac{\partial E}{\partial Y_j^{(x,y)}} \frac{\partial Y_j^{(x,y)}}{\partial W_j^{(p,q)}} \tag{2.7}$$

$$\frac{\partial E}{\partial b_j} = \sum_{y=0}^{h_Y} \sum_{x=0}^{w_Y} \frac{\partial E}{\partial Y_j^{(x,y)}} \tag{2.8}$$

$$\frac{\partial Y_j^{(x,y)}}{\partial X_i^{(u,v)}} \tag{2.9}$$

2.5 Activation

The activation function applies to thl og P de input feature maps in a element-wise manner. Thus, the output dimension is equal to the input dimension.

ReLU activation

For ReLU activation, whose activation function is defined as $f(x) = max(x,0)$, its derivative is 1 if $x > 0$ and 0 otherwise. Thus, this layer doesn't contain any parameter to be tuned. The purpose of this layer using ReLU activation is to "block" errors back propagate to previous neurons whose activation is lower than or equal to 0. Thus,

$$\frac{\partial E}{\partial x_i^{(u,v)}} = \frac{\partial E}{\partial y_i^{(u,v)}} M_i(u,v) \tag{2.10}$$

Where $M_i(u,v)$ is a binary mask matrix, where $M_i(u,v) = 1$ if $f(x_i^{(u,v)}) > 0$, and 0 otherwise.

2.6 DropOut

2.7 Response Normalization

2.8 Pooling

2.9 Full Connection

3 OpenCL Implementation