Batched Recursive Neural Networks

Kushal Arora
arorak@amazon.com

Abstract

In this article, I propose a novel way to do a batch implementation of the recursive neural network models.

1 General Formulation

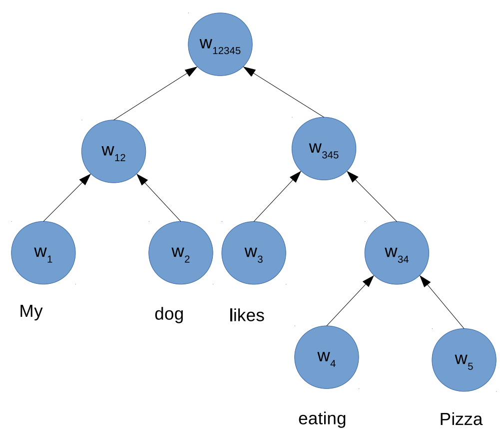

Figure 1: Composition tree for sentence 'My dog like eating Pizza.'

Let D be a dataset containing tokenized sentences. Let $W = w_1 w_2 .. w_n$ be a sentence of length n in this data set. Let $t(W)$ be the tree structured associated with the sentence W. Figure 1 depicts one such tree for a sentence W_{12345}.
Let $R(t, W)$ be the set of binary rules and leaf nodes used in the derivation of sentence W. Equation 1 depicts the rule set $R(t, W_{12345})$ for the sentence W_{12345} shown in Figure 1.

$$
\begin{aligned}
R(t, W_{12345}) = \{& w_{12} \leftarrow w_1 w_2, \\
& w_{34} \leftarrow w_3 w_4, \\
& w_{345} \leftarrow w_3 w_{45}, \\
& w_{12345} \leftarrow w_{12} w_{345}, \\
& w_1, w_2, w_3, w_4, w_5\}
\end{aligned} \tag{1}
$$

1.1 Recurrent Neural Network Scoring Equation

Let d be the dimensions of the embedding space and $X_{i..k}$, $X_{k+1..j}$ be the $d \times 1$ embedding vector corresponding to sequences $w_{i..k}$ and $w_{k+1..j}$ respectively. Using standard recursive neural network composition function, $X_{i..j}$ can be computed as

$$X_{i..j} = f\left(H\begin{bmatrix} X_{i..k} \\ X_{k+1..j} \end{bmatrix}\right)$$

f here is a non linearity like $tanh$ and H is a $d \times 2d$ composition matrix.
The score for the node $X_{i..j}$ can calculated as

$$s(w_{i..j} \leftarrow w_{i..k} w_{k+1..j}) = s(w_{i..j}) = g(UX_{i..j})$$

For example, let's consider sentiment classification task with recursive neural network architecture[1]. Let K be the caridinality of the target classes.

$$s(w_{i..j}) = softmax(UX_{i..j})$$

U in this case is a $K \times d$ sentiment classification matrix.
Let $T(w_{i..j})$ be a one-hot vector with sentiment label for node $w_{i..j}$, the likelihood for the node $\mathcal{L}(w_{i...j}, T)$ will be

$$\mathcal{L}(w_{i..j}, T) = T(w_{i...j})^T s(w_{i..j})$$

The likelihood for the whole sentence $\mathcal{L}(W; t, T)$ can now be calculated as

$$\mathcal{L}(W, T; t) = \prod_{w_{i..j} \leftarrow w_{i..k} w_{k+1..j} \in R(t,W)} \mathcal{L}(w_{i..j}, T)$$

1.2 An Alternate formulation

Let $\theta(W)$[1] be a $n \times n \times n$ sparse tensor such that

$$\theta(W,t)[i,j,k] = \begin{cases} 1 & w_{i...j} \leftarrow w_{i..k} w_{k+1..j} \in R(t,W) \\ 0 & otherwise \end{cases} \tag{2}$$

[1] For simplicity, we index all our matrices starting at 1, so first element of matrix $\theta(i,j,k)$ would be $\theta[1,1,1]$.

Let $\pi(w_{i..j})$ be the inside score of the sequence $w_{i..j}$. The inside score is defined recursively as:

$$\pi(w_{ii}) = \mathcal{L}(w_{ii}, T)$$

and

$$\pi(w_{i..j}; \theta) = \sum_{k=i+1}^{j-1} \theta(W,t)[i,j,k]\mathcal{L}(w_{i..j}, T)\pi(w_{i..k})\pi(w_{k+1..j}) \tag{3}$$

As $\theta(W,t)$ is an indicator tensor, we can write score of the sentence W, $\mathcal{L}(W,t)$ in terms of π as:

$$\mathcal{L}(W,t) = \pi(w_{1..n}; \theta)$$

1.3 A Matrix Formulation for Inside Score.

Now, let $\Pi(W)$[2] be a $n \times n$ matrix of inside scores whose rows are starting index and columns are the span sizes i.e. index $[i,j]$ would correspond to inside score $\pi(w_{i..(i+j)})$. First row of this matrix would be inside scores of all the leaf nodes $\pi(w_{ii})$ and each row would contain $n-i$ entries. Equation 4 shows an example matrix for $\Pi(W_{12345})$.

$$\Pi(W_{12345}) = \begin{bmatrix} \pi(w_1) & \pi(w_{12}) & \pi(w_{123}) & \pi(w_{1234}) & \pi(w_{12345}) \\ \pi(w_2) & \pi(w_{23}) & \pi(w_{234}) & \pi(w_{2345}) & 0 \\ \pi(w_3) & \pi(w_{34}) & \pi(w_{345}) & 0 & 0 \\ \pi(w_4) & \pi(w_{45}) & 0 & 0 & 0 \\ \pi(w_5) & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Let Π_l^i be a $l \times l$ sub-matrix starting at row $i+1$. $\bar{I}_l$ is the reflection of identity matrix I_l, $\Pi[i]_l$ is first l elements of ith row of matrix $\Pi(W)$. Let $j = i+l$, $\mathcal{L}_{ij}$ is a length l row vector such that element k, $i < k < j$, is $\mathcal{L}(w_{i..j} \leftarrow w_{i..k}w_{k+1..j})$. $[i,l]$th index of matrix Π can be calculated as follows

$$\Pi[i,l] = \pi(w_{i..j}) = (\theta[i,j,1:l] \circ \mathcal{L}_{ij})(\bar{I}_l \circ \Pi_l^i)\Pi[i]_l^T \tag{5}$$

Here, $\circ$ is a element wise multiplication.
Let's try to compute $\pi(w_{12345})$ for sentence W_{12345} from Figure 1. In this case, $i=1$, $j=5$, and l would be $j-i=4$. $\mathcal{L}_{ij}$ and $\theta[i,j,1:l]$ in this case will be

$$\mathcal{L}_{15} = [\mathcal{L}(w_{12345}, T), \mathcal{L}(w_{12345}, T), \mathcal{L}(w_{12345}, T), \mathcal{L}(w_{12345}, T)]$$

and

$$\theta[1,5,1:4] = [0,1,0,0]$$

[2]In similar vein, the first element of matrix $\Pi(W)$ would be indexed at $[1,1]$.

Now, $(\theta[ij] \circ S_{ij})$ will be

$$(\theta[i,j] \circ S_{ij}) = [0, \mathcal{L}(w_{12345}, T), 0, 0]$$

$\bar{I}_l \circ \Pi_l$, $\Pi[i]_l^T$ and $(\bar{I}_l \circ \Pi_l^i)\Pi[i]_l^T$ then would be

$$\bar{I}_l \circ \Pi_l = \begin{bmatrix} \pi(w_{2345}) & 0 & 0 & 0 \\ 0 & \pi(w_{345}) & 0 & 0 \\ 0 & 0 & \pi(w_{45}) & 0 \\ 0 & 0 & 0 & \pi(w_5) \end{bmatrix},$$

$$\Pi[i]_l^T = \begin{bmatrix} \pi(w_1) \\ \pi(w_{12}) \\ \pi(w_{123}) \\ \pi(w_{1234}) \end{bmatrix},$$

and

$$(\bar{I}_l \circ \Pi_l^i)\Pi[i]_l^T = \begin{bmatrix} \pi(w_1)\pi(w_{2345}) \\ \pi(w_{12})\pi(w_{345}) \\ \pi(w_{123})\pi(w_{45}) \\ \pi(w_{1234})\pi(w_5) \end{bmatrix}$$

Finally, $\pi(w_{12345})$ would be

$$\pi(w_{12345}) = \mathcal{L}(w_{12345}, T)\pi(w_{12})\pi(w_{345})$$

In the formulation above, the tree structure is codified in tensor θ and doesn't need a dynamic network to compute $\mathcal{L}(W, T; t)$, hence this formulation can be used for batch computation.

1.4 Batched Recursive Neural Network

Let B be the batch of size b, N be the length of the longest sentence in batch B. Let $\tilde{\theta}$ be a 4 dimensional tensor with dimensions $b \times N \times N \times N$ and $\tilde{\Pi}$ be a $b \times N \times N$ tensor.
Let W_m be the mth sentence in batch B and let the length of W_m be n, we define $\tilde{\theta}$ as $\tilde{\theta}[m, 1:n, 1:n, :] = \theta(W_m)$ and $\tilde{\Pi}$ as $\tilde{\Pi}[m, 1:n, 1:n] = \Pi(W_m)$.
Let $l = j - i$ and $\tilde{I}_l$ is a $b \times l \times l$ tensor with $\tilde{I}_l[m, :, :] = \bar{I}_l$, we can compute $\tilde{\Pi}[:, i, j]$ as:

$$\tilde{\Pi}[:, i, j] = (\tilde{\theta}[:, i, j, 1:l] \circ \tilde{\mathcal{L}}_{ij}[:])(\tilde{I}_l[:] \circ \tilde{\Pi}_l^i[:])\tilde{\Pi}[:, i]_l^T \tag{6}$$

Here, $\tilde{\mathcal{L}}_{ij}$ is a $l \times b$ matrix such that$\tilde{\mathcal{L}}_{ij}[m] = \mathcal{L}_{ij}(W_m)$. Figure 2 shows the Equation 6.
Similarly, we can compute $\tilde{\mathcal{L}}_{ij}$ in the batched manner. Let $\tilde{X}_{i.k.j}$ is a $b \times 2d$ matrix defined as

$$\tilde{X}_{i.k.j}[m, :] = \begin{bmatrix} X_{i..k}^m \\ X_{k+1..j}^m \end{bmatrix}$$

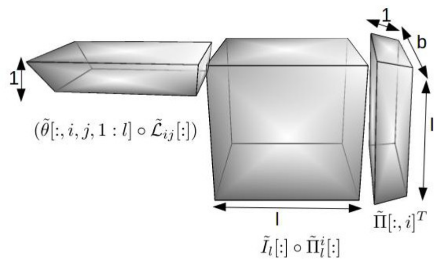

Figure 2: Computation of $\tilde{\Pi}[:, i, j]$

Here, $X^m_{i..k}$, $X^m_{k+1..j}$ are the embedding of $w^m_{i..k}$, $w^m_{k+1..j}$ which are span $[i, k]$ and $[k+1, j]$ of sentence W_m. Also, such that $w^m_{i..j} \leftarrow w^m_{i..k} w^m_{k+1..j} \in R(W_m)$.
Let $\tilde{H}$ be a $b \times 2d \times d$ matrix such that $\tilde{H}[m, :, :] = H$. The $\tilde{X}_{i..j}$ will be

$$\tilde{X}_{i..j}[:] = f\left(\tilde{H}[:]\tilde{X}_{i.k.j}[:]\right)$$

Let $\tilde{U}$ be a $b \times K \times d$ matrix such that $\tilde{U}[m, :, :] = U$, let $w^m_{i..j}$ be the i, jth span of sentence W_m. Let$\tilde{T}$ is a $b \times N \times N \times K$ matrix such that $\tilde{T}[m, i, j, :] = T(w^m_{i..j})$. Now, we can compute $\tilde{\mathcal{L}}_{ij}$ as

$$\tilde{\mathcal{L}}_{ij}[:] = \tilde{T}[\, :, i, j, :] softmax(\tilde{U}[:] X_{i..j}[:])$$

1.5 Implementation Details

Let's start by computing $(\tilde{I}_l[:] \circ \tilde{\Pi}^i_l[:])\tilde{\Pi}[i, :]^T_l$. Let $\tilde{\Pi}^i_l$ be $b \times l \times l$ tensor such that $\tilde{\Pi}^i_l = \tilde{\Pi}[:, i+1 : (i+1+l), 1 : l]$, $\tilde{\Pi}_i$ be $b \times l$ matrix such that $\tilde{\Pi}_i = \tilde{\Pi}[:, i, 1 : l]$ and Π_{out} be a b length vector that contains the output of $(\tilde{I}_l[:] \circ \tilde{\Pi}^i_l[:])\tilde{\Pi}[i, :]^T_l$ operation.

insideKernel($\tilde{\Pi}^i_l$, $\tilde{\Pi}_i$, Π_{out}, l):

```
b = blockIdx
t= threadIdx
```
Π_{out}[b]+=$\tilde{\Pi}^i_l$[b,t,l-t]*$\tilde{\Pi}_i$[b,t]

Let's define a compacting operation which given a vector of one-hot vectors, returns an array of indexes that were 1 in the input matrix. Let $\tilde{\theta}_{ij}$ be $b \times l$ one hot matrix such that $\tilde{\theta}_{ij} = \theta[:, i, j, 1 : l]$ and θ^{out}_{ij} is a length $b \times 2$ matrix for output such that $\theta^{out}_{ij}[:, 1]$ is the batch index and $\theta^{out}_{ij}[:, 2]$ is the split index k, we define an kernel compactIdxKernel as

```
compactIdxKernel($\tilde{\theta}_{ij}$,$\theta_{ij}^{out}$,$n$)
```

n here is the length of matrix θ_{ij}^{out}.
Let $w_{i..j} \leftarrow w_{i..k}w_{k+1..j} \in R(W_m)$. Let's define a likelihoodKernel which takes in the length d embedding vectors $X_{i..k}$ and $X_{k+1..j}$ corresponding to $w_{i..k}$ and $w_{k+1..j}$, $w_{i..j}$'s label $T_{i..j}$ and return $w_{i..j}$'s embedding vector $X_{i..j}$ and its likelihood $\tilde{\mathcal{L}}_{ij}$.

likelihoodKernel($X_{i..k}$,$X_{k+1..j}$,$T_{i..j}$,$X_{i..j}$,$\tilde{\mathcal{L}}_{ij}$)

Let $\tilde{X}$ be a $b \times N \times N \times d$ tensor such that $\tilde{X}[m,i,j]$ is the embedding vector for span i,j from sentence m in batch B, let $\tilde{T}$ be a $b \times N \times N$ label matrix such that $T[m,i,j] = T_{i..j}^m$ is the label for span i,j from sentence m in batch B and $\tilde{\mathcal{L}}_{ij}$ is a length b vector such that $\tilde{\mathcal{L}}_{ij}[m] = \mathcal{L}_{ij}(W_m)$ i.e. mth index in $\tilde{\mathcal{L}}_{ij}$ is the score of span i,j from sentence m in batch B.
Using these definitions, we now define another kernel scoringKernelBatch which takes as input tensors $\tilde{X}$, $\tilde{T}$, θ_{ij}^{out} and returns S_{ij}

```
scoringKernelBatch($\tilde{X}$,i,j,$\tilde{T}$,$\theta_{ij}^{out}$, $\tilde{\mathcal{L}}_{ij}$):
    i = blockIdx
    b = $\theta_{ij}^{out}$[i,1]
    k = $\theta_{ij}^{out}$[i,2]
    scoringKernel($\tilde{X}$[b,i,k],$\tilde{X}$[b,k+1,j],$\tilde{T}$[b,i,j],$\tilde{X}$[b,i,j],$\tilde{\mathcal{L}}_{ij}$[b])
```

Finally, we define insideMethod that computes $\tilde{\Pi}[:,i,j]$

```
insideMethod(i,l,$\tilde{\Pi}$,$\tilde{\theta}$,$\tilde{T}$,$\tilde{X}$, b):
    $\tilde{\Pi}_l^i$=$\tilde{\Pi}$[:,i+1:(i+l+1),1:l]
    $\tilde{\Pi}_i$=$\tilde{\Pi}$[:,i,1:l]
    $\Pi_{out}$= zeros(b)
    insideKernel($\tilde{\Pi}_l^i$,$\tilde{\Pi}_i$,$\Pi_{out}$,l)

    $\tilde{\theta}_{ij}$= $\theta[:,i,j,1:l]$
    $\theta_{ij}^{out}$ = zeros(b)

    compactIdxKernel($\tilde{\theta}_{ij}$,$\theta_{ij}^{out}$,$n$)

    $\tilde{\mathcal{L}}_{ij}$ = zeros(b)
    scoringKernelBatch($\tilde{X}$,i,i+l,$\tilde{T}$,$\theta_{ij}^{out}$,$\tilde{\mathcal{L}}_{ij}$)

    $\Pi[:,i,j] = \tilde{\mathcal{L}}_{ij} \circ \Pi_{out}$
```

References

[1] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)* (2013), vol. 1631, Citeseer, p. 1642.