Gradient Calculation

Chih-Jen Lin
National Taiwan University
Last updated: March 19, 2019
Outline

1. Introduction
2. Gradient Calculation
3. Discussion
Outline

1. Introduction
2. Gradient Calculation
3. Discussion
Many deep learning courses have contents like

- fully-connected networks
- its optimization problem
- its gradient (back propagation)
- ...
- other types of networks (e.g., CNN)
- ...

If I am a student of such courses, after seeing the significant differences of CNN from fully-connected networks, I wonder how the back propagation can be done.
The problem is that back propagation for CNN seems to be very complicated.

So fewer people talk about details.

Challenge: can we clearly describe it in a simple way?

That’s what we would like to try here.
Outline

1. Introduction
2. Gradient Calculation
3. Discussion
Gradient Calculation

Gradient I

Consider two layers $m$ and $m + 1$. The variables between them are $W^m$ and $b^m$, so we aim to calculate

$$
\frac{\partial f}{\partial W^m} = \frac{1}{C} W^m + \frac{1}{l} \sum_{i=1}^{l} \frac{\partial \xi_i}{\partial W^m},
$$

(1)

$$
\frac{\partial f}{\partial b^m} = \frac{1}{C} b^m + \frac{1}{l} \sum_{i=1}^{l} \frac{\partial \xi_i}{\partial b^m}.
$$

(2)

Note that (1) is in a matrix form.
Following past developments such as Vedaldi and Lenc (2015), it is easier to transform them to a vector form for the derivation.
For the convolutional layers, recall that

\[
S^{m,i} = W^m \text{mat}(P^m_\phi P^m_{\text{pad}} \text{vec}(Z^{m,i}))_{h^m h^m d^m \times a^m_{\text{conv}} b^m_{\text{conv}}} + \phi(\text{pad}(Z^{m,i}))
\]

\[
b^m 1^T_{a^m_{\text{conv}} b^m_{\text{conv}}}
\]

\[
Z^{m+1,i} = \text{mat}(P^m_{\text{pool}} \text{vec}(\sigma(S^{m,i})))_{d^{m+1} \times a^{m+1} b^{m+1}}, \quad (3)
\]
Gradient Calculation

Vector Form II

- We have

$$
\text{vec}(S^{m,i}) = \text{vec}(W^m \phi(\text{pad}(Z^{m,i}))) + \text{vec}(b^m \mathbb{1}_{a_{\text{conv}}^m}^T b_{\text{conv}}^m)
$$

$$
= (\mathcal{I}_{a_{\text{conv}}^m} b_{\text{conv}}^m \otimes W^m) \text{vec}(\phi(\text{pad}(Z^{m,i}))) + 
(\mathbb{1}_{a_{\text{conv}}^m} b_{\text{conv}}^m \otimes \mathcal{I}_{d^{m+1}}) b^m
$$

$$
= (\phi(\text{pad}(Z^{m,i}))^T \otimes \mathcal{I}_{d^{m+1}}) \text{vec}(W^m) + 
(\mathbb{1}_{a_{\text{conv}}^m} b_{\text{conv}}^m \otimes \mathcal{I}_{d^{m+1}}) b^m,
$$

Chih-Jen Lin (National Taiwan Univ.)
Gradient Calculation

**Vector Form III**

where $\mathcal{I}$ is an identity matrix, and (4) and (5) are respectively from

$$
\text{vec}(AB) = (\mathcal{I} \otimes A)\text{vec}(B),
$$

$$
= (B^T \otimes \mathcal{I})\text{vec}(A),
$$

$$
\text{vec}(AB)^T = \text{vec}(B)^T (\mathcal{I} \otimes A^T),
$$

$$
= \text{vec}(A)^T (B \otimes \mathcal{I}),
$$

where $\otimes$ is the Kronecker product.
What’s the Kronecker product? If \( A \in \mathbb{R}^{m \times n} \) then
\[
A \otimes B = \begin{bmatrix}
a_{11}B & \cdots & a_{1n}B \\
\vdots & & \vdots \\
a_{m1}B & \cdots & a_{mn}B
\end{bmatrix},
\]
a much bigger matrix
Gradient Calculation

Vector Form V

- For the fully-connected layers,

\[ s^{m,i} = W^m z^{m,i} + b^m \]

\[ = (I_1 \otimes W^m) z^{m,i} + (1_1 \otimes I_{n_{m+1}}) b^m \]

\[ = ((z^{m,i})^T \otimes I_{n_{m+1}}) \text{vec}(W^m) + (1_1 \otimes I_{n_{m+1}}) b^m, \]

where (10) and (11) are from (6) and (7), respectively.
An advantage of using (4) and (10) is that they are in the same form.

Further, if for fully-connected layers we define

$$\phi(\text{pad}(z^{m,i})) = I_{nm}z^{m,i}, \quad L^c < m \leq L + 1,$$

then (5) and (11) are in the same form.

Thus we can derive the gradient of convolutional and fully-connected layers together.
For convolutional layers, from (5),

\[
\frac{\partial \xi_i}{\partial \text{vec}(W^m)^T} = \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \frac{\partial \text{vec}(S^{m,i})}{\partial \text{vec}(W^m)^T}
\]

\[
= \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \left( \phi(\text{pad}(Z^{m,i}))^T \otimes I_{d_m+1} \right)
\]

\[
= \text{vec} \left( \frac{\partial \xi_i}{\partial S^{m,i}} \phi(\text{pad}(Z^{m,i}))^T \right)^T
\]

where (12) is from (9).
Gradient Calculation

We applied chain rule here

Note that we define

$$\frac{\partial y}{\partial (x)^T} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_{|x|}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{|y|}}{\partial x_1} & \cdots & \frac{\partial y_{|y|}}{\partial x_{|x|}} \end{bmatrix}, \quad (13)$$

where $x$ and $y$ are column vectors.
Thus if

\[ y = Ax \]

then

\[ \frac{\partial y}{\partial (x)^T} = \begin{bmatrix} A_{11} & A_{12} & \cdots \\ A_{21} \\ \vdots \end{bmatrix} = A \]
Similarly

\[
\frac{\partial \xi_i}{\partial (b^m)^T} = \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \frac{\partial \text{vec}(S^{m,i})}{\partial (b^m)^T} \\
= \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \left( 1_{a_{\text{conv}}^m b_{\text{conv}}^m} \otimes I_{d^{m+1}} \right) \\
= \text{vec} \left( \frac{\partial \xi_i}{\partial S_{m,i}} 1_{a_{\text{conv}}^m b_{\text{conv}}^m} \right)^T , \quad (14)
\]

where (14) is from (9).
To calculate (12), $\phi(\text{pad}(Z_{m,i}^m))$ has been available from the forward process of calculating the function value.

In (12) and (14), $\partial \xi_i / \partial S_{m,i}$ is also needed.

We will show that it can be obtained by a backward process.
Gradient Calculation

Calculation of $\partial \xi_i / \partial S_{m,i}$

- What we will do is to assume that $\partial \xi_i / \partial Z_{m+1,i}$ is available.
- Then we show details of calculating $\partial \xi_i / \partial S_{m,i}$ and $\partial \xi_i / \partial Z_{m,i}$ for layer $m$.
- Thus a back propagation process.
- We have the following workflow:

$$Z_{m,i} \leftarrow \text{padding} \leftarrow \text{convolution} \leftarrow \sigma(S_{m,i}) \leftarrow \text{pooling} \leftarrow Z_{m+1,i}.$$  \hspace{1cm} (15)
Gradient Calculation

Calculation of $\frac{\partial \xi_i}{\partial S^{m,i}}$

- Assume the RELU activation function is used:

$$\frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} = \frac{\partial \xi_i}{\text{vec}(\sigma(S^{m,i}))} \frac{\text{vec}(\sigma(S^{m,i}))}{\text{vec}(S^{m,i})^T}$$

- Note that:

$$\frac{\text{vec}(\sigma(S^{m,i}))}{\text{vec}(S^{m,i})^T}$$

is a squared diagonal matrix.

- Recall that we assume:

$$\sigma'(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{otherwise}
\end{cases}$$
Calculation of $\frac{\partial \xi_i}{\partial S^{m,i}}$

- We can define

$$I[S^{m,i}]_{(p,q)} = \begin{cases} 1 & \text{if } S^{m,i}_{(p,q)} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

and have

$$\frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} = \frac{\partial \xi_i}{\text{vec}(\sigma(S^{m,i}))} \odot \text{vec}(I[S^{m,i}])^T$$

where $\odot$ is Hadamard product (i.e., element-wise products)

- Q: can we extend this to other activation functions?
Calculation of $\partial \xi_i / \partial S^{m,i}$ IV

- Next

$$\begin{align*}
\frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} &= \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \frac{\partial \text{vec}(Z^{m+1,i})}{\partial \text{vec}(\sigma(S^{m,i}))} \frac{\partial \text{vec}(\sigma(S^{m,i}))}{\partial \text{vec}(S^{m,i})} \\
&= \left( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \frac{\partial \text{vec}(Z^{m+1,i})}{\partial \text{vec}(\sigma(S^{m,i}))} \right) \odot \text{vec}(I[S^{m,i}])^T
\end{align*}$$

(16)

$$\begin{align*}
&= \left( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \frac{\partial \text{vec}(Z^{m+1,i})}{\partial \text{vec}(\sigma(S^{m,i}))} \right) \odot \text{vec}(I[S^{m,i}])^T \\
&= \left( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} P^{m,i}_{\text{pool}} \right) \odot \text{vec}(I[S^{m,i}])^T
\end{align*}$$

(18)
Calculation of $\frac{\partial \xi_i}{\partial S^m_i}$

- Note that (18) is from (3)
In the end we calculate $\frac{\partial \xi_i}{\partial Z^{m,i}}$ and pass it to the previous layer.

$$
\frac{\partial \xi_i}{\partial \text{vec}(Z^{m,i})^T} = \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \frac{\partial \text{vec}(S^{m,i})}{\partial \text{vec}(\phi(\text{pad}(Z^{m,i})))^T} \frac{\partial \text{vec}(\phi(\text{pad}(Z^{m,i})))}{\partial \text{vec}(\text{pad}(Z^{m,i}))^T}
$$

$$
= \frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} \left( I_{a_{\text{conv}}}^{m} b_{\text{conv}}^{m} \otimes W^{m} \right) P_{\phi}^{m} P_{\text{pad}}^{m}
$$

$$
= \text{vec} \left( (W^{m})^T \frac{\partial \xi_i}{\partial S^{m,i}} \right)^T P_{\phi}^{m} P_{\text{pad}}^{m}
$$
Calculation of $\frac{\partial \xi_i}{\partial S^m_i}$ VII

where (21) is from (8).
For fully-connected layers, by the same form in (10), (11), (4) and (5), we immediately get results from (12), (14), (18) and (21).

\[
\frac{\partial \xi_i}{\partial \text{vec}(W^m)^T} = \text{vec} \left( \frac{\partial \xi_i}{\partial s^{m,i}_j} (z^{m,i})^T \right)^T \\
\frac{\partial \xi_i}{\partial (b^m)^T} = \frac{\partial \xi_i}{\partial (s^{m,i})^T} 
\]
Fully-connected Layers II

\[
\frac{\partial \xi_i}{\partial (\mathbf{z}^{m,i})^T} = \left( (\mathbf{W}^m)^T \frac{\partial \xi_i}{\partial (\mathbf{s}^{m,i})} \right)^T \mathcal{I}_{nm}
\]

\[
= \left( (\mathbf{W}^m)^T \frac{\partial \xi_i}{\partial (\mathbf{s}^{m,i})} \right)^T, \quad (25)
\]

where

\[
\frac{\partial \xi_i}{\partial (\mathbf{s}^{m,i})^T} = \frac{\partial \xi_i}{\partial (\mathbf{z}^{m+1,i})^T} \odot I[\mathbf{s}^{m,i}]^T. \quad (26)
\]

- Finally, we check the initial values of the backward process.
Fully-connected Layers III

- Assume that the the square is used and in the last layer we have an identity activation function.

- Then

\[
\frac{\partial \xi_i}{\partial z_{L+1,i}} = 2(z_{L+1,i}^* - y^i), \quad \text{and} \quad \frac{\partial \xi_i}{\partial s_{L,i}} = \frac{\partial \xi_i}{\partial z_{L+1,i}}.
\]
Recall we said that in
\[
\frac{\partial \xi_i}{\partial \mathcal{W}^m} = \frac{\partial \xi_i}{\partial S^m,i} \phi(\text{pad}(Z^{m,i}))^T,
\]

$Z^{m,i}$ is available from the forward process

Therefore

$Z^{m,i}, \forall m$

are stored.
But we also need $S^{m,i}$ for

$$\frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} = \left( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} P_{m,i}^{\text{pool}} \right) \odot \text{vec}(I[S^{m,i}])^T$$

Do we need to store both $Z^{m,i}$ and $S^{m,i}$?

We can avoid storing $S^{m,i}$, $\forall m$ by replacing (18) with

$$\frac{\partial \xi_i}{\partial \text{vec}(S^{m,i})^T} = \left( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \odot \text{vec}(I[Z^{m+1,i}])^T \right) P_{m,i}^{\text{pool}}.$$ (27)
Why? Let’s look at the relation between $Z^{m+1,i}$ and $S^{m,i}$

$$Z^{m+1,i} = \text{mat}(P_{\text{pool}}^{m,i} \text{vec}(\sigma(S^{m,i})))$$

$Z^{m+1,i}$ is a “smaller matrix” than $S^{m,i}$

That is, (18) is a “reverse mapping” of the pooling operation
In (18),

\[
\frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \times P_{\text{pool}}^{m,i}
\]

(28)

generates a large zero vector and puts values of \( \frac{\partial \xi_i}{\partial \text{vec}(Z^{m+1,i})^T} \) into positions selected earlier in the max pooling operation.

Then, element-wise multiplications of (28) and \( I[S^{m,i}]^T \) are conducted.

Positions not selected in the max pooling procedure are zeros after (28)
They are still zeros after the Hadamard product between (28) and $I[S^{m,i}]^T$

Thus, (18) and (27) give the same results.
Summary of Operations I

- We show convolutional layers only and the bias term is omitted.

Operations in order:

\[
\frac{\partial \xi_i}{\partial \text{vec}(S_{m,i})^T} = \left( \frac{\partial \xi_i}{\partial \text{vec}(Z_{m+1,i})^T} \otimes \text{vec}(I[Z_{m+1,i}])^T \right) P_{\text{pool}}^{m,i} \tag{29}
\]

\[
\frac{\partial \xi_i}{\partial W^m} = \frac{\partial \xi_i}{\partial S_{m,i}} \phi(\text{pad}(Z_{m,i}))^T \tag{30}
\]
\[
\frac{\partial \xi_i}{\partial \text{vec}(Z_{m,i})^T} = \text{vec} \left( (W^m)^T \frac{\partial \xi_i}{\partial S_{m,i}} \right)^T P_m^m \phi P_{\text{pad}}^m
\]

Note that after (29), we need
\[
\frac{\partial \xi_i}{\partial \text{vec}(S_{m,i})^T} \rightarrow \frac{\partial \xi_i}{\partial S_{m,i}}
\]
because in (30) and (31), matrix form is needed.

In (29), information of the next layer is used.
Instead we can do

\[
\frac{\partial \xi_i}{\partial \text{vec}(Z^{m,i})^T} \odot \text{vec}(I[Z^{m,i}])^T
\]

in the end of the current layer

Thus an implementation for one convolutional layer:

\[
\Delta \leftarrow \max(\text{vec}(\Delta)^T P_{pool}^{m,i})
\]

\[
\frac{\partial \xi_i}{\partial W^m} = \Delta \phi(\text{pad}(Z^{m,i}))^T
\]
Gradient Calculation

Summary of Operations IV

\[ \Delta \leftarrow \text{vec} \left( (W^m)^T \Delta \right)^T P^m \phi P^m_{\text{pad}} \]

\[ \Delta \leftarrow \Delta \odot I[Z^{m,i}] \]

- A sample segment of code

```python
for m = LC : -1 : 1
    if model.wd_subimage_pool(m) > 1
        dXidS = reshape(vTP(param, model, net, m, dXidS, 'pool_gradient'),
                       model.ch_input(m+1), []);
    end
    phiZ = padding_and_phiZ(model, net, m);
```

Chih-Jen Lin (National Taiwan Univ.)
Summary of Operations V

\[
\text{net.dlossdWm} = \text{dXidS*phiZ'};
\]
\[
\text{net.dlossdbm} = \text{dXidS*ones(model.wd_conv(m)*model.ht_conv(m)*S_k, 1)};
\]

if \( m > 1 \)
\[
V = \text{model.weightm'} \times \text{dXidS};
\]
\[
\text{dXidS} = \text{reshape(vTP(param, model, net, m, V, 'phi_gradient'), model.ch_input(m), [])};
\]

\[
\% \text{vTP_pad}
\]
\[
a = \text{model.ht_pad(m)}; b = \text{model.wd_pad(m)};
\]
Gradient Calculation

Summary of Operations VI

dXidS = dXidS(:, net.idx_padm + a*b*[0:S_k-1]);

% activation function
dXidS = dXidS.*(net.Zm > 0);
end
end
Outline

1. Introduction
2. Gradient Calculation
3. Discussion
We tried to have a simple way to describe the gradient calculation for CNN.

Is the description good enough? Can we do better?