

# 誤差逆伝播法等に用いる 計算グラフ の基本パーツ

DeepLearning(/tags/DeepLearning) 1781

深層学習(/tags/%E6%B7%B1%E5%B1%A4%E5%AD%A6%E7%BF%92) 466

機械学習(/tags/%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92) 2626

プログラミング(/tags/%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%9F%E3%83%B3%E3%82%B0) 462

10

いいね

0

コメント

いいね

(/ceptree)

(/piyo56)

(/k-kudan)

(/jun1\_0803)

(/tttamaki)

(/kenmatsu4)

(/nicd\_batt)

(/yan\_hisa\_)

(/akiraa)

(/takavfx)

t-tkd3a (/t-tkd3a)

2017年04月17日に更新

(/t-tkd3a/items/031c0a4dbf25fd2866a3/revisions)

3

ストック

計算グラフ(computational graph)の 基本パーツ の図解です。

ニューラルネットワークの誤差逆伝播法を学習時に 計算グラフを 知りました。  
局所的な計算を連鎖させていく考え方で、理解しやすくて良いですね。

## 計算グラフの基本パーツ

本記事の図では、黒線・黒文字は順伝搬、赤線・赤文字は逆伝播を示します。

$z=f(x)$

(<https://camo.qiitusercontent.com/40d0f220140bcac696dea3658b82da5e67257d58/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f32633435643336342d326465662d396661642d3535336332d6332356565316430356531382e706e67>)

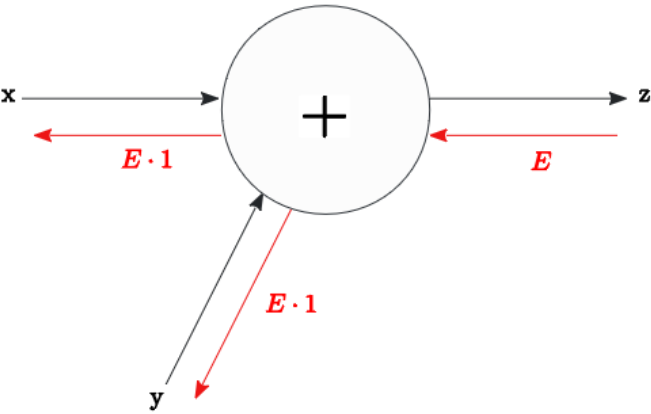
順方向  $z=f(x)$  の計算の場合、  
その逆伝播は 入力値  $E$  に  $f(x)$  の微分値を掛けた値が 出力となる。

## 加算ノード

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

<https://qiita.com/t-tkd3a/items/031c0a4dbf25fd2866a3>

1/5

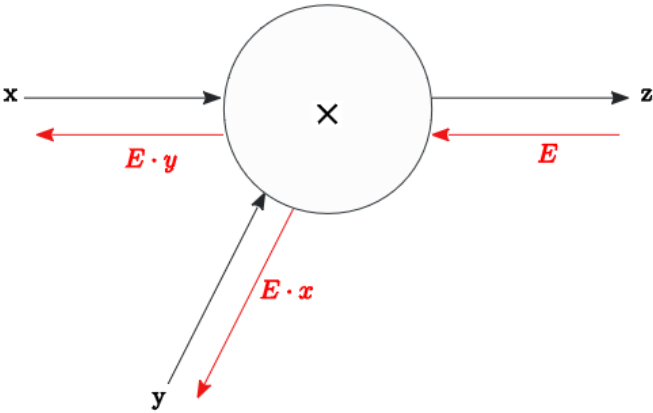


(<https://camo.qiitautercontent.com/c152942642de8c4d3398648ea233e09302ea0430/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f39336236663539662d613237382d663832302d383564662d6631336132383734313932632e706e67>)

加算ノードの逆伝播は、 **入力値  $E$  をそのまま** 伝達する。

順方向  $z = x + y$  に対し、  
 $x$  の微分値  $\frac{\partial z}{\partial x}$  は 1  
 $y$  の微分値  $\frac{\partial z}{\partial y}$  も 1  
のため、入力  $E$  を そのまま伝達 となる。

## 乗算ノード



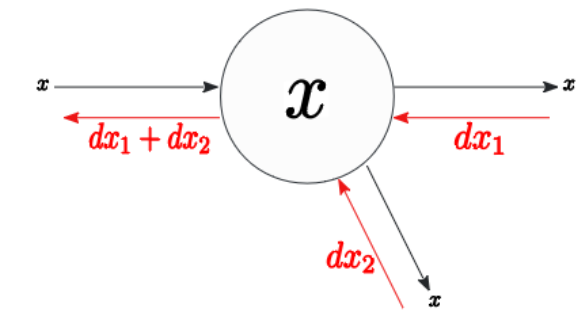
(<https://camo.qiitautercontent.com/da64457dcc8cd66bc959e02fed5e42961fd84da1/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f39383336373436332d626631382d623838392d633463652d6433373339303935386630632e706e67>)

加算ノードの逆伝播は、 **入力値  $E$  × 順伝播を代入した値** を伝達する。

順方向  $z = xy$  に対し、  
 $x$  の微分値  $\frac{\partial z}{\partial x}$  は  $y$   
 $y$  の微分値  $\frac{\partial z}{\partial y}$  は  $x$   
のため、入力値  $E$  × 順伝播を代入した値を 伝達となる。

## 分岐ノード

順伝播の式中の2か所で  $x$  が参照されている場合に、 $x$  の分岐と扱う。

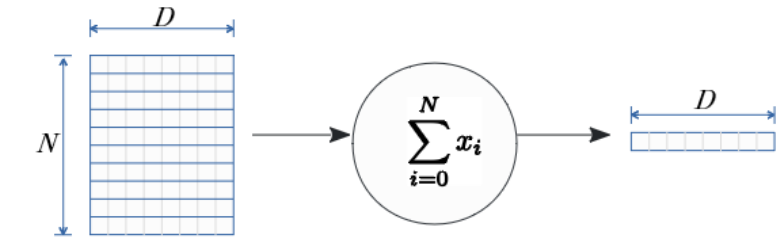


(<https://camo.qiitusercontent.com/c530839184c13271bdbb295c5d7c5ef0e35501ca/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f61346661353663662d643639652d633636642d623838612d3138656431356464383033352e706e67>)

逆伝播は、分岐先からの**逆伝播値の和** を伝達する。

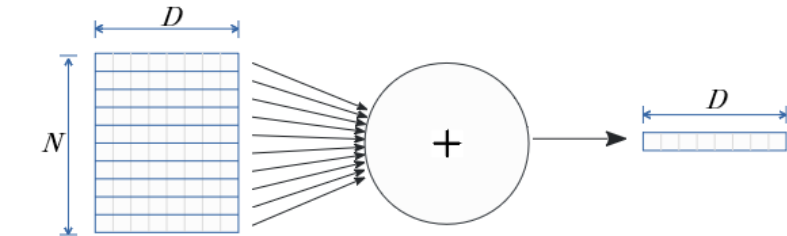
## 総和ノード

伝播する行列での一方方向で(下図の例では $N$ の方向で)総和を求める場合、 $(N,D)$ の行列は  $(D,)$ の次元になる。



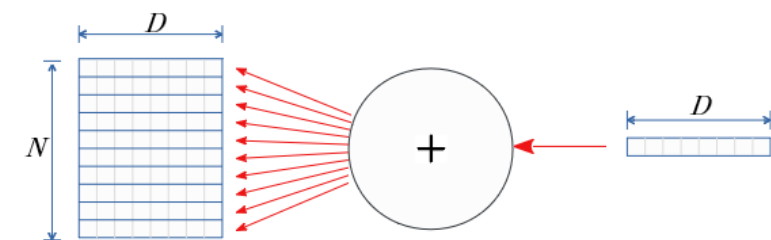
(<https://camo.qiitusercontent.com/e55d22c4d07c8b3f6f39041c0ded30132897b69e/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f64623131303439372d393439652d633534302d326631612d3430626130356432613836622e706e67>)

総和は 下図の様に **多数の加算ノード** と読みかえ出来る。



(<https://camo.qiitusercontent.com/86eaa83ca7529b24c116b6df01507f6bf9cd1a79/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f63653331303437382d646362342d303732312d353363322d3366666635373465633331312e706e67>)

総和の逆伝播は、 $(D,)$ 次元の逆伝播の入力行列を、**そのまま $N$ 個に伝達し $(N,D)$ 次元の行列を生成する。**

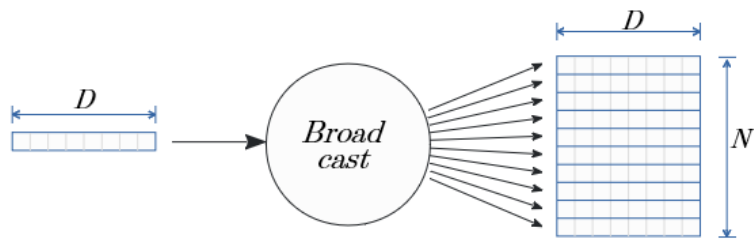


(<https://camo.qiitusercontent.com/eb046e90e3c20d3c1818d1bff2ef3306d543f007/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f63666136373235312d633434312d623465632d323863372d3135386435643331313563372e706e67>)

## Broadcast ノード

行列のアダマール積にて

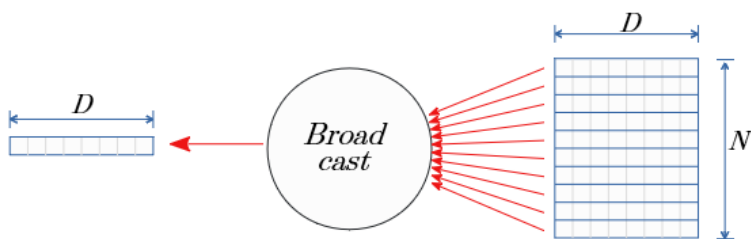
( $N, D$ )次元の行列A と ( $D, D$ )次元の行列Bの積を求める場合、Bは下図で示す Broadcast が行われ( $N, D$ )次元の行列として演算する。



(<https://camo.qiitusercontent.com/df15b4effe9f9f44597b11c996dd681731ac2465/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f34653364376535652d623866332d356262622d613635612d3737343737633730663937302e706e67>)

Broadcastは  $N$ 個への分岐ノードと読みかえが出来る。

逆伝播は、( $N, D$ )次元の行列から ( $D, D$ )次元への 総和となる。



(<https://camo.qiitusercontent.com/e0d160142a92469e1d63ce93c691066ca20d2ff0/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3136393538392f34383461663938352d616537632d373066332d366132392d3635653136613165316534622e706e67>)

## 参考文献

- ゼロから作るDeep Learning —Pythonで学ぶディープラーニングの理論と実装 ([https://www.amazon.co.jp/gp/product/4873117585/ref=as\\_li\\_tf\\_tl?ie=UTF8&camp=247&creative=1211&creativeASIN=4873117585&linkCode=as2&tag=noriaki0213-22](https://www.amazon.co.jp/gp/product/4873117585/ref=as_li_tf_tl?ie=UTF8&camp=247&creative=1211&creativeASIN=4873117585&linkCode=as2&tag=noriaki0213-22)) (<https://camo.qiitusercontent.com/f56bcf25aa95f87fe6f5829a99facc89bef3a5c4/687474703a2f2f69722d6a702e616d617a6f6e2d616473797374656d2e636f6d2f652f69723f743d6e6f7269616b69303231332d3232266c3d617332266f3d3926613d34383733313137353835>)(第 5 章)
- "Understanding the backward pass through Batch Normalization Layer" (<https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html>)

## 関連項目

Mind で Neural Network (準備編2) 順伝播・逆伝播 図解 (<http://qiita.com/t-tkd3a/items/9bf50f2e10e6a15b6ed5>)



Tweet0

G+

4



t-tkd3a (/t-tkd3a)

168 Contribution

フォロー

人気の投稿


- 高速な Convolution 処理を目指してみた。 Kn2Image方式 (/t-tkd3a/items/879a5fd6410320fe504e)
- Batch Normalization の理解 (/t-tkd3a/items/14950dbf55f7a3095600)
- 3x3 畳み込みフィルタ 結果画像 (/t-tkd3a/items/d5f52212e3b941bc36cf)
- シンプルなNNで SeLU と eLU と ReLU を見比べてみる (/t-tkd3a/items/dc163b8a762365ab5a61)
- Convolution処理の手法 Im2Col方式の図解 (/t-tkd3a/items/6b17f296d61d14e12953)

計算グラフの基本パーツ

- z=f(x)
  - 加算ノード
  - 乗算ノード
  - 分岐ノード
  - 総和ノード
  - Broadcast ノード
- 参考文献

いいね

10




(/takavfx) (/akiraa) (/yan\_hisa\_) (/nicd\_batt) (/kenmatsu4) (/tttamaki) (/jun1\_0803) (/k-kudan) (/piyo56) (/ceptree)

📁 ストック

✏️ 編集リクエスト (/drafts/031c0a4dbf25fd2866a3/edit)

...

🔗 この記事は以下の記事からリンクされています

 **Batch Normalization の理解** (/t-tkd3a/items/14950dbf55f7a3095600#\_reference-2dbc57edc651a8522f04) からリンク 6ヶ月前