しおん研-ゼロつくゼミ

第５回

今日やること

- 誤差逆伝播法の理論
- 単純なレイヤの実装
- 活性化関数レイヤの実装
- Affineレイヤ・Softmaxレイヤの実装

学習のフレームワーク（復習）

- 目的は、推論に最適な（＝もっとも適切な出力を返す）パラメータを見つけること！

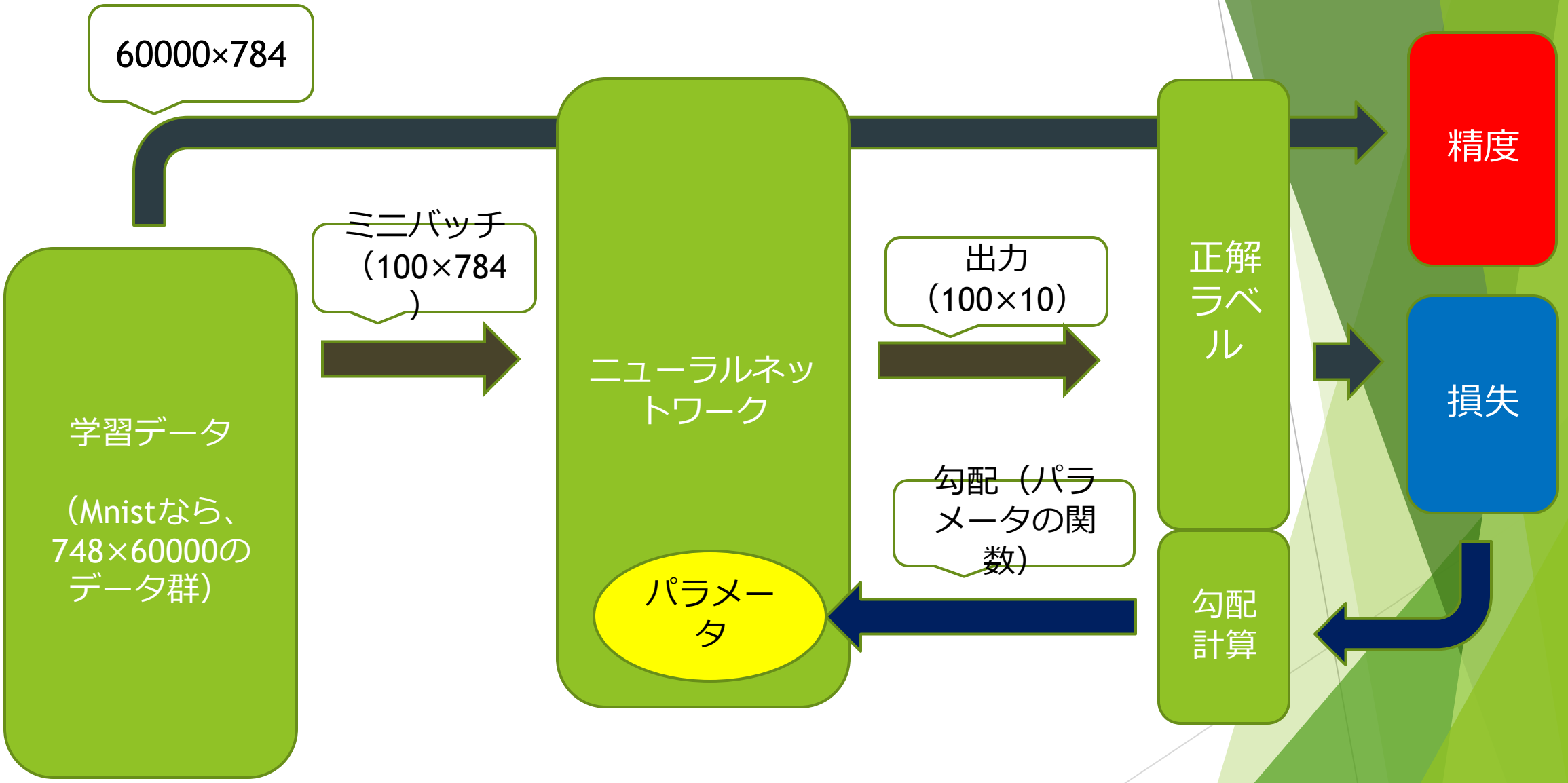
- そのためにはどうすればいい？

→ある推論に対して、それがどれくらい正しいか評価する関数を定義し、その評価が最大になるようにパラメータを定めれば良いのでは？

学習データX→出力データY = f(X,W)

正解データt

評価関数L(W)
＝損失関数


60000×784
ミニバッチ（100×784）
学習データ
（Mnistなら、748×60000のデータ群）
ニューラルネットワーク
パラメータ
出力（100×10）
正解ラベル
精度
損失
勾配（パラメータの関数）
勾配計算

学習のフレームワーク

これまでの勾配計算：数値微分

- 一つのパラメータを更新するのに、ニューラルネットの推論課程を２回行う必要がある！

- つまり、一回の学習で、順方向計算を

  ２×（パラメータの総数）　回行わなければならない！

- パラメータが増えると計算が終わらない・・・

数値微分に代わる勾配計算法：誤差逆伝播法

（例題）

- Polina さんは毎日、ロシアのスーパーでりんごを２個とオレンジを３個買っています。ここで、

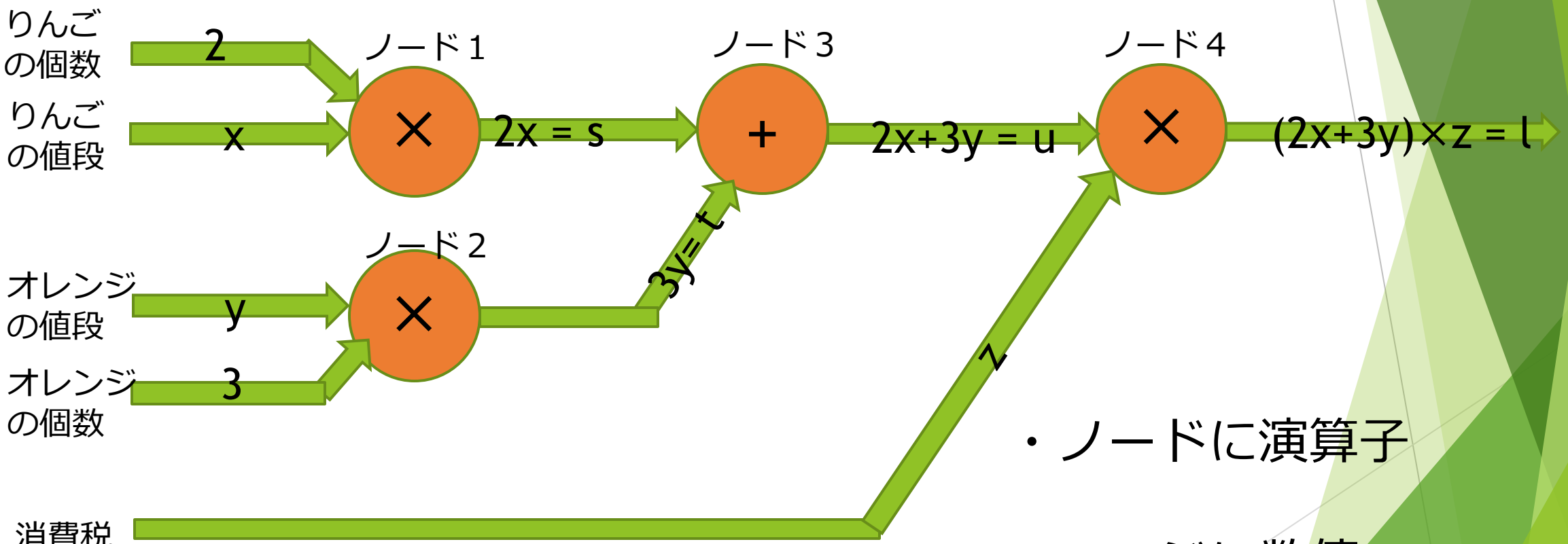
1.りんごが昨日と比べて10ルーブル値上がりしていた

2.オレンジが昨日と比べて5ルーブル値上がりしていた

3.暴動により、今日から消費税が20%から10%に下がった

それぞれの時、Polinaさんは昨日より何ルーブル多く支払う必要があるでしょうか。

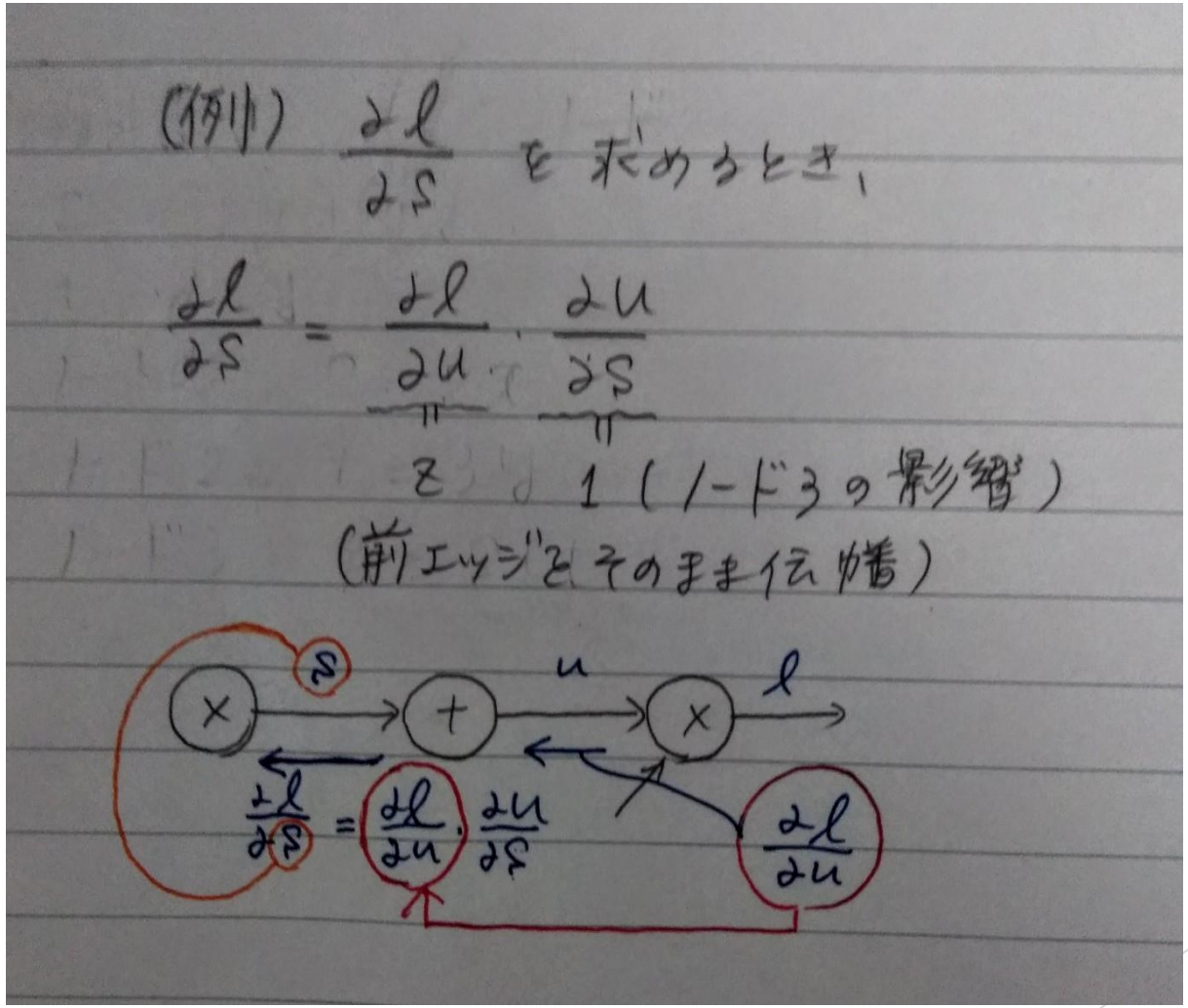
（解答例）計算グラフ



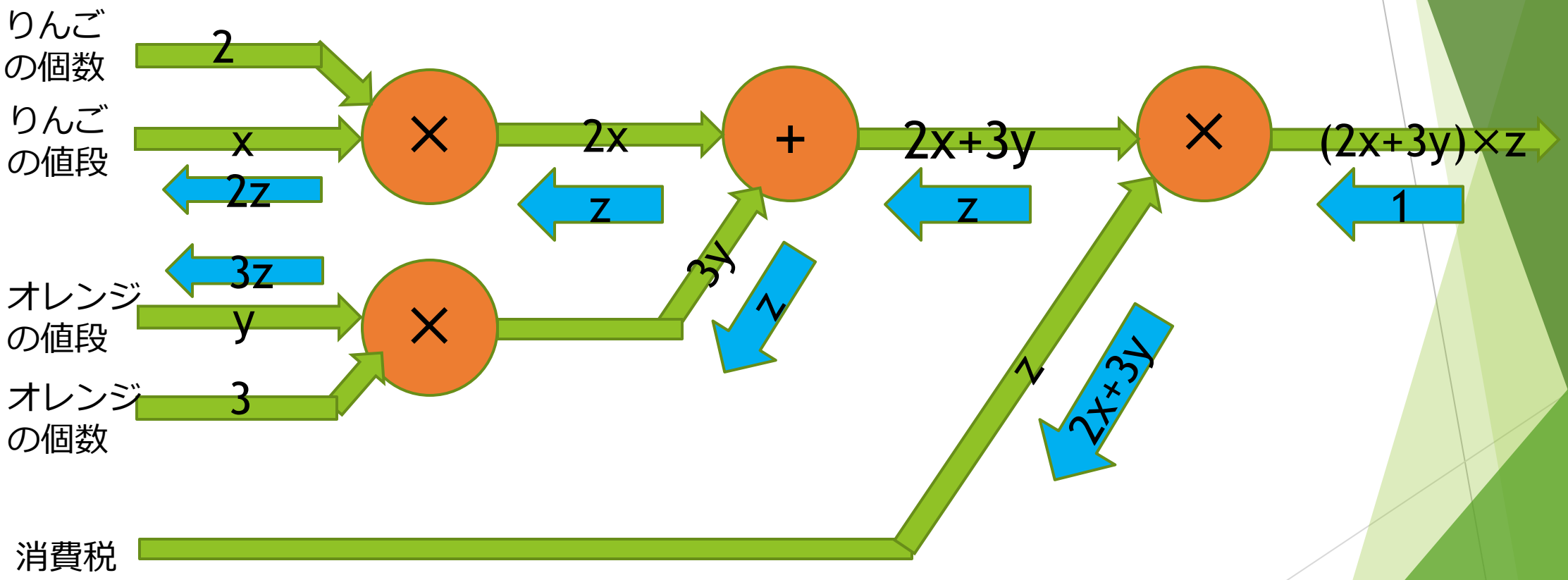
・ノードに演算子

・エッジに数値

誤差逆伝播法：勾配を逆方向に伝播する！(575)



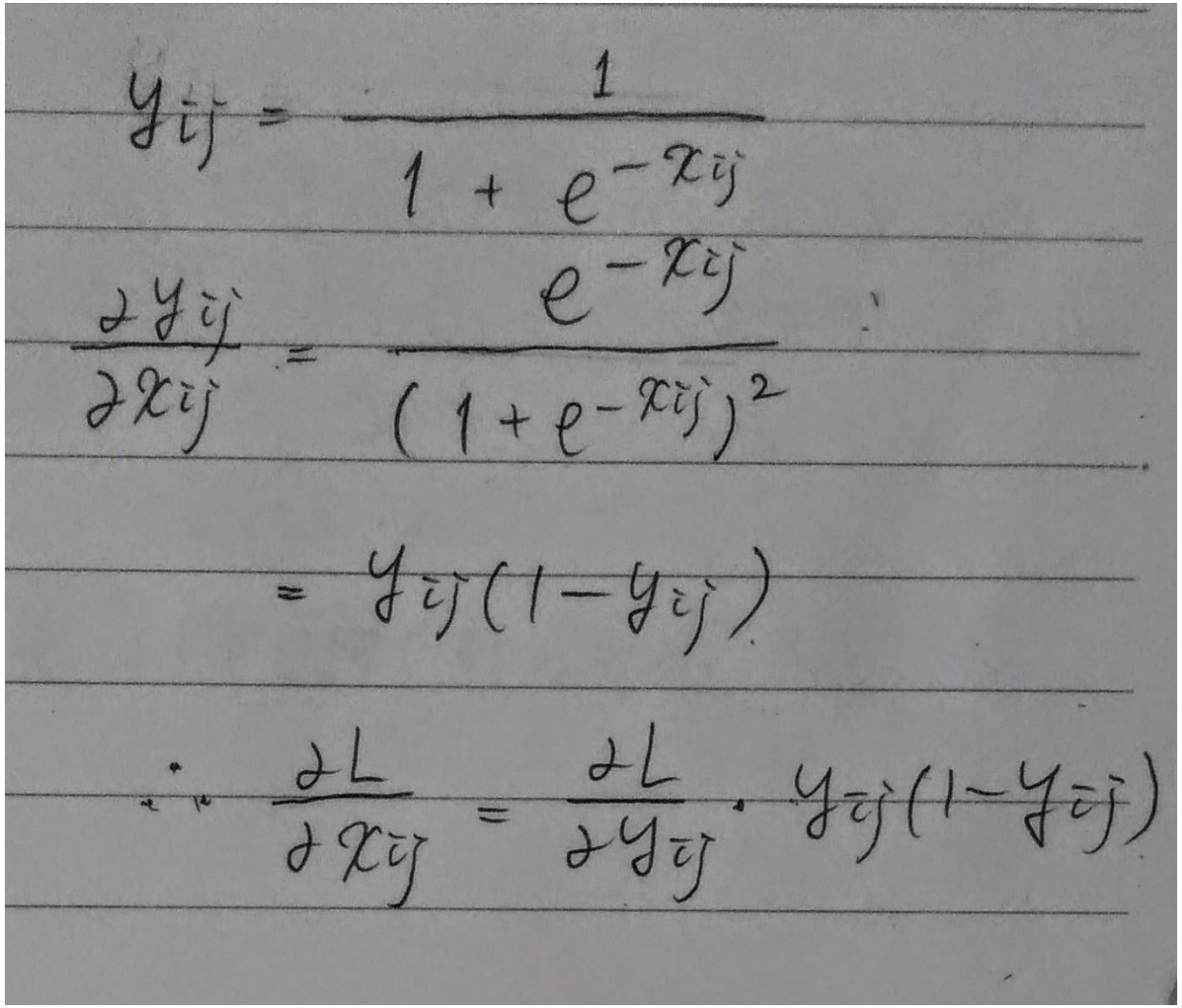
勾配を逆方向に伝播する！



活性化関数レイヤの逆伝播（Sigmoid）

- Sigmoidレイヤ

活性化関数レイヤの逆伝播（Sigmoid）



$$y_{ij} = \frac{1}{1 + e^{-x_{ij}}}$$

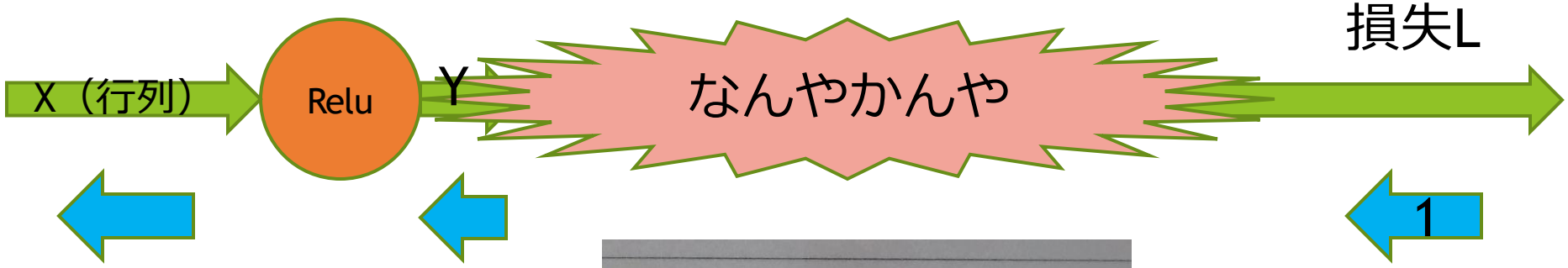
$$\frac{\partial y_{ij}}{\partial x_{ij}} = \frac{e^{-x_{ij}}}{(1+e^{-x_{ij}})^2}$$

$$= y_{ij}(1-y_{ij})$$

$$\therefore \frac{\partial L}{\partial x_{ij}} = \frac{\partial L}{\partial y_{ij}} \cdot y_{ij}(1-y_{ij})$$

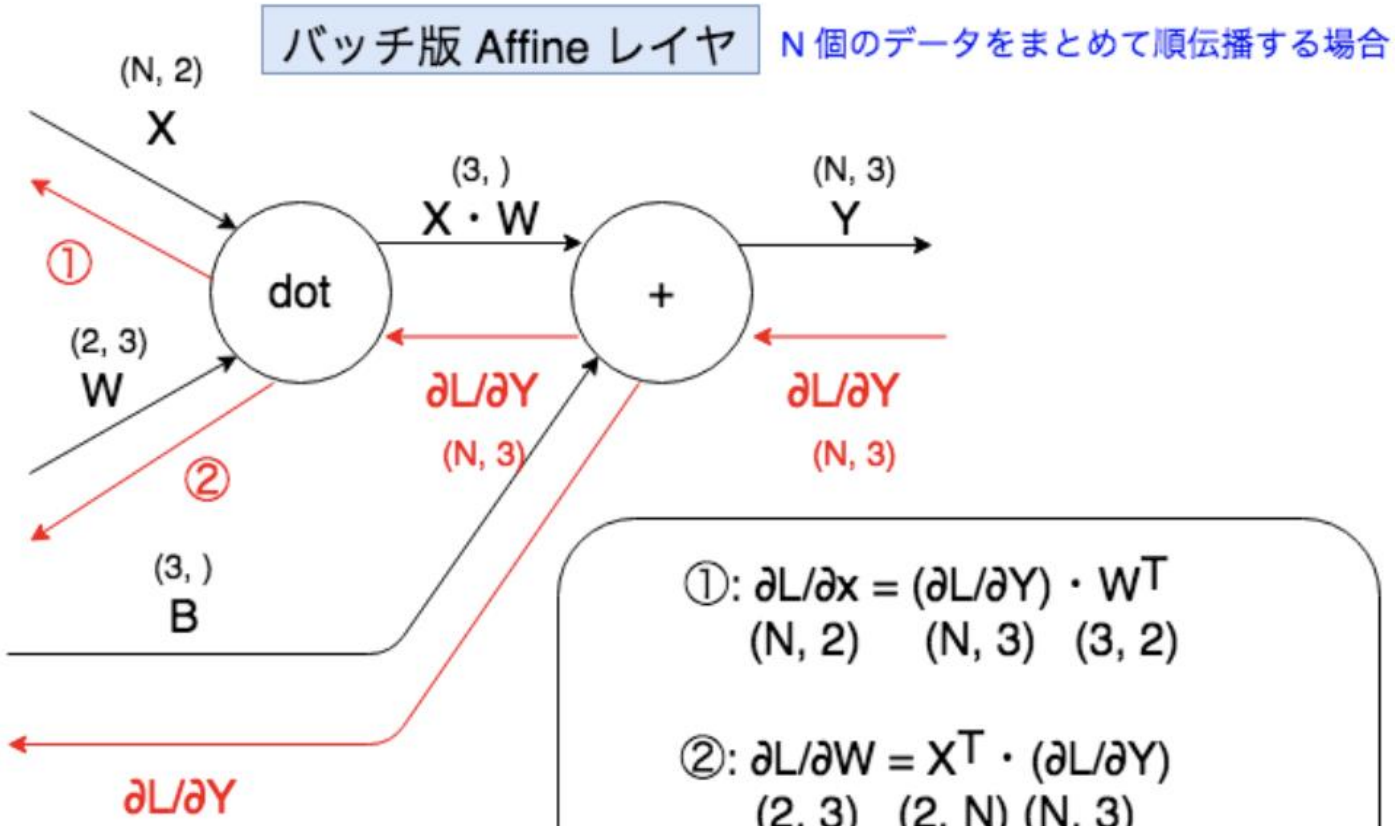
活性化関数レイヤの逆伝播（Relu）

- Reluレイヤ

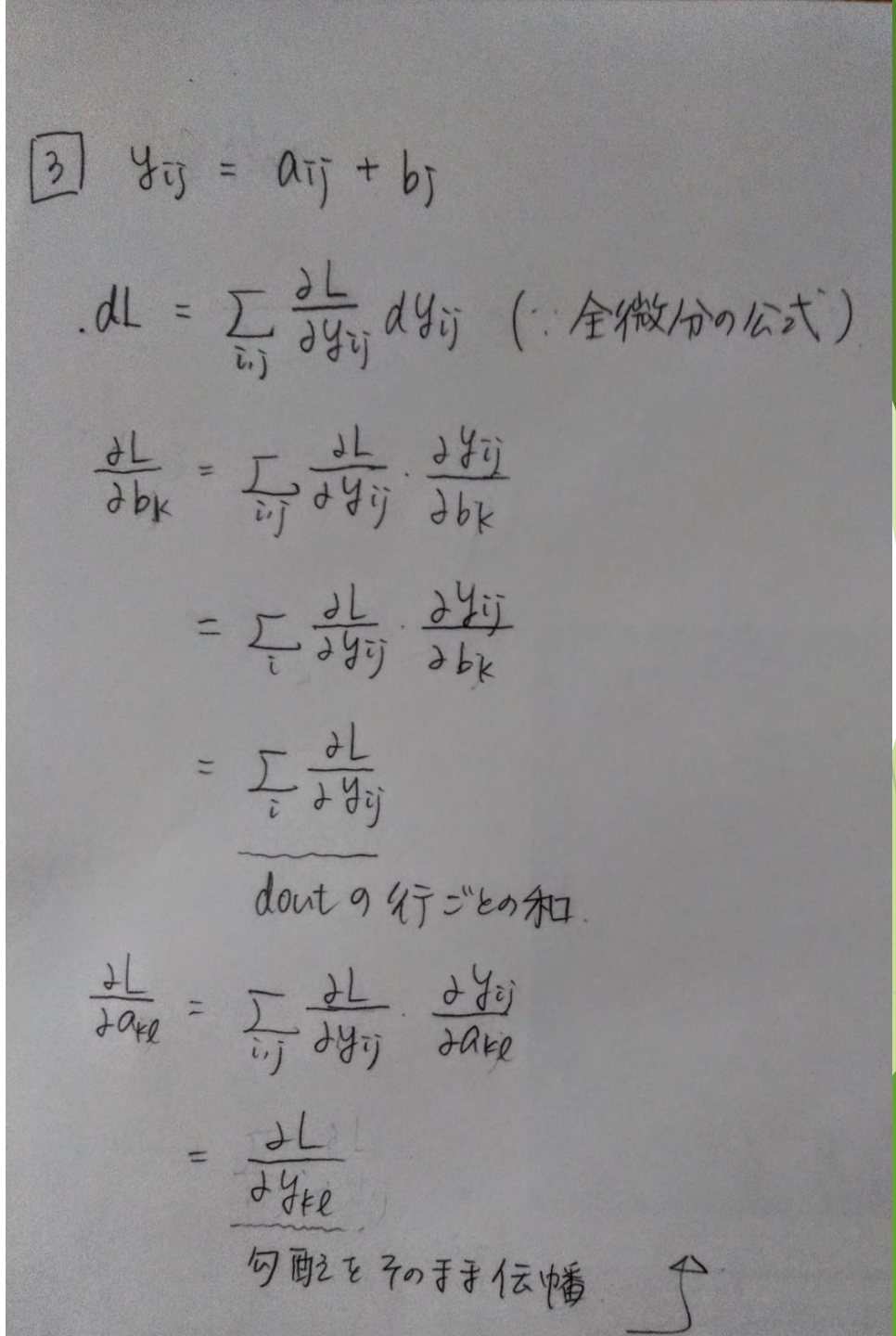


$$\frac{\partial y_{ij}}{\partial x_{ij}} = \begin{cases} 1 \ (x_{ij} > 0) \\ 0 \ (x_{ij} \leq 0) \end{cases}$$

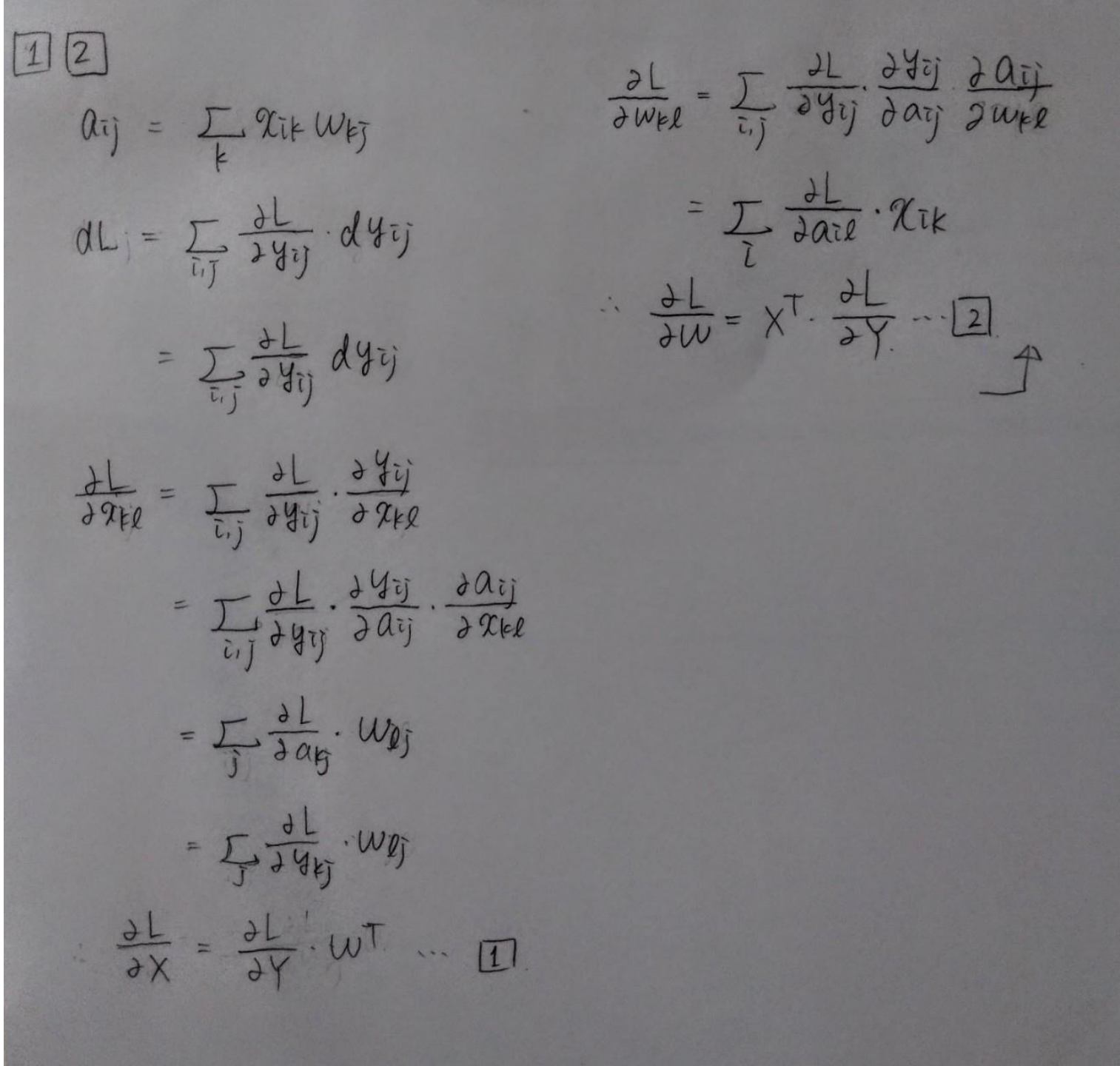
Affineレイヤの逆伝播



Affineレイヤの逆伝播（証明）

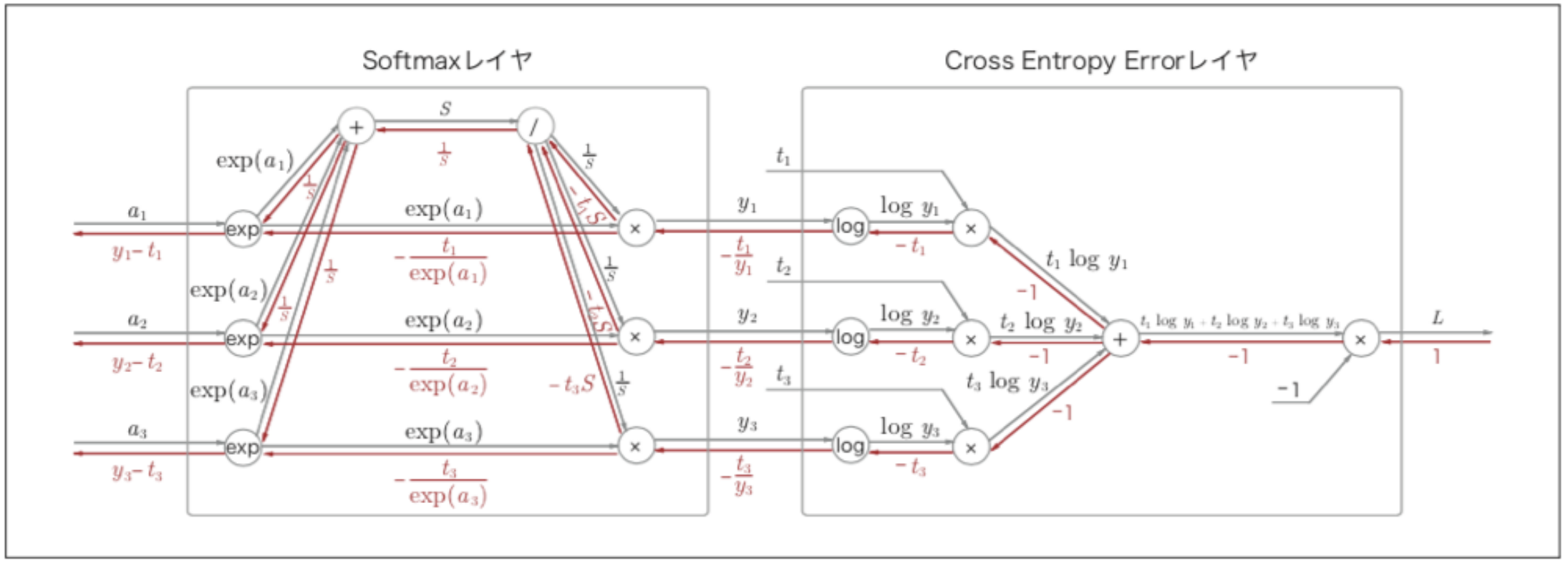


Affineレイヤの逆伝播（証明）

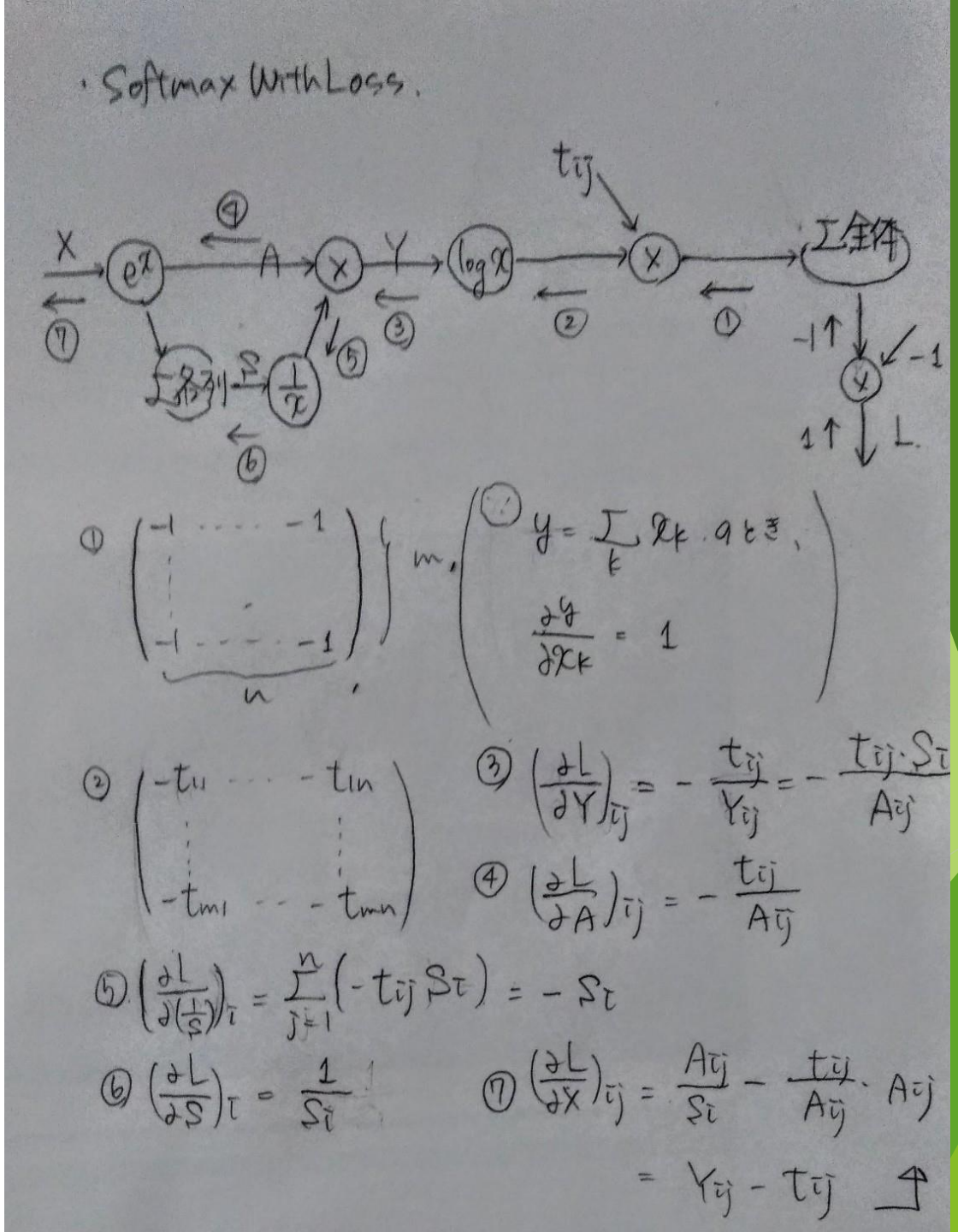


SoftmaxWithLossレイヤの逆伝播

Softmax-with-Loss レイヤ(ソ フトマックス関数と交差エントロピー誤差)の計算グラフ



SoftmaxWithLoss レイヤの逆伝播（証明）



レイヤの実装

- レイヤ＝関数だと思えばOK

- レイヤ実装の基本（ここでは）

```
class hoge:
    def __init__(ハイパーパラメータ):
        #ハイパーパラメータに加え、順伝播で算出した値を逆伝播でも使う時にインスタンス変数として格納
    def forward(self,x):
    def backward(self,dout):
```

- このレイヤを逆にたどることで、勾配が計算できるようにする！

Google Colab で実装

・加算、乗算レイヤ

・活性化関数レイヤ

・Affine、Softmaxレイヤの実装

画像の出典：いずれも https://www.dragonarrow.work/articles/176