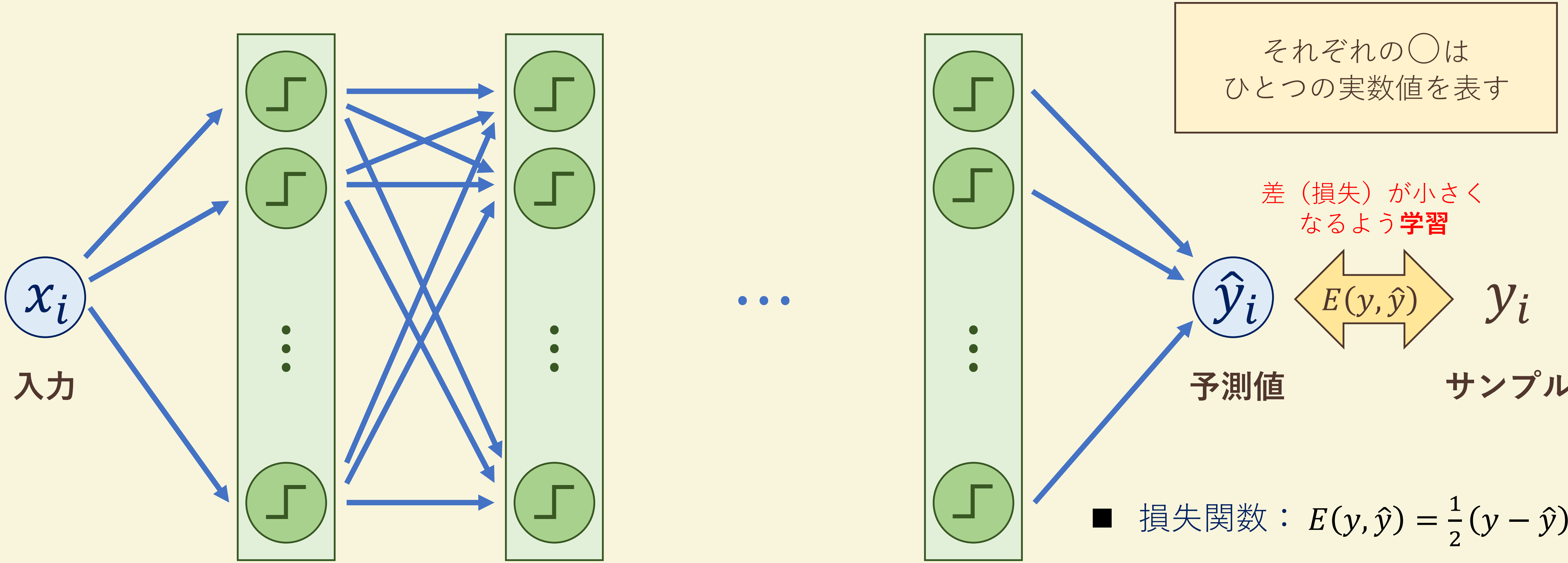
電気情報工学セミナーⅡ

実験のためのPython＆Git入門（機械学習による関数の近似実験）
～　第5回　ニューラルネットワークによる関数の近似　～

池原研究室

ニューラルネットワーク

概要



■ 損失関数：$E(y,\hat{y}) = \frac{1}{2}(y-\hat{y})^2$

重み付き和 と **しきい値処理** を繰り返す

重み（入力の〇の重要度）と
しきい値（切り捨てる値のライン）を
少しずつ調整していい感じにする

重み付き和としきい値処理の例

成績は
出席点2割　中間テスト3割　期末テスト5割
で点数をつけます。
90～100点はS(4)　80～89点はA(3)　70～79点はB(2)
60～69点はC(1)　0～59点はD(0)
として評定が決まります。

成績も**重み付き和**と**しきい値処理**でつけられている！！

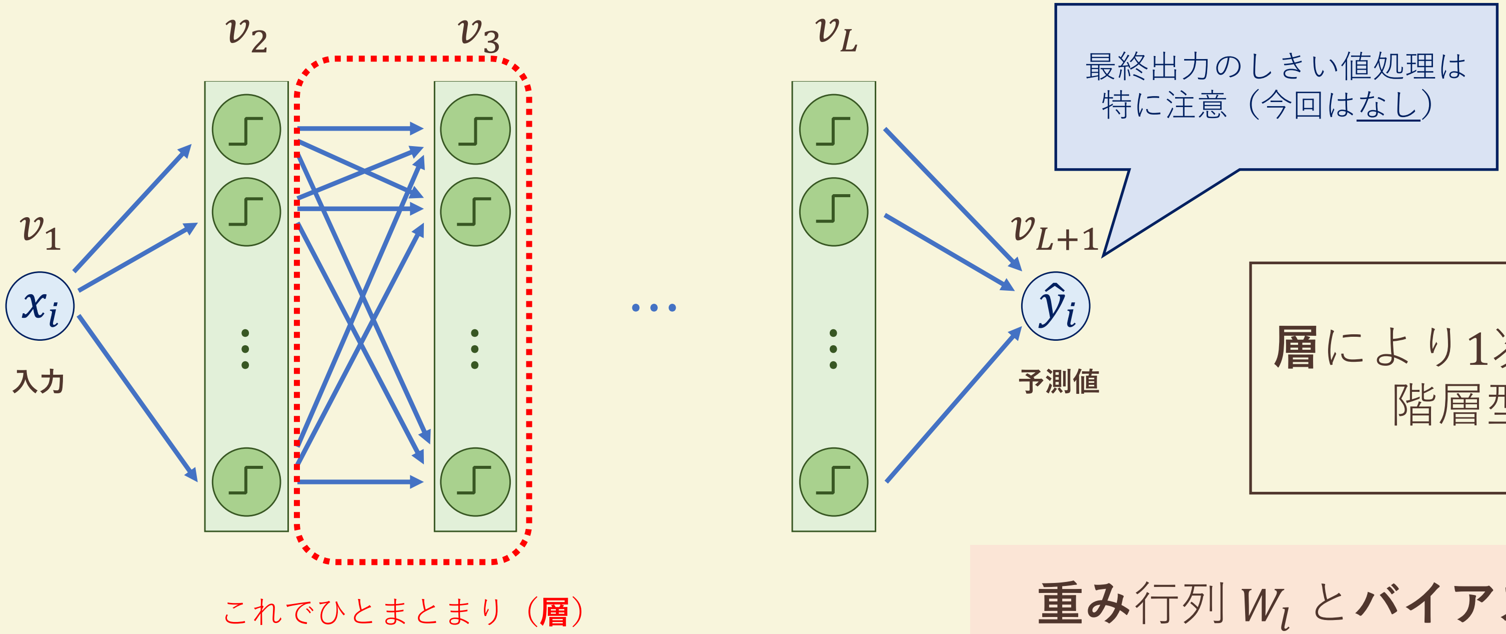
（例）出席点：90点　中間テスト：70点　期末テスト：80点の場合

評価用の点数：$0.2 \cdot 90 + 0.3 \cdot 70 + 0.5 \cdot 80 = 79$　評定：B (2)

半導体系の評定は低いけど
数学系の評定は十分高いから
ウチの会社で活躍できそうだな

各授業で出された評定が
さらなる評価につながる

数式による表現



層により1次元配列 v_l を順次変換していく
階層型ニューラルネットワーク

$$v_{l+1} = \sigma_l(W_l v_l + b_l)$$

青は重み付き和　緑はしきい値処理

重み行列 W_l と**バイアス**ベクトル b_l の各要素を学習

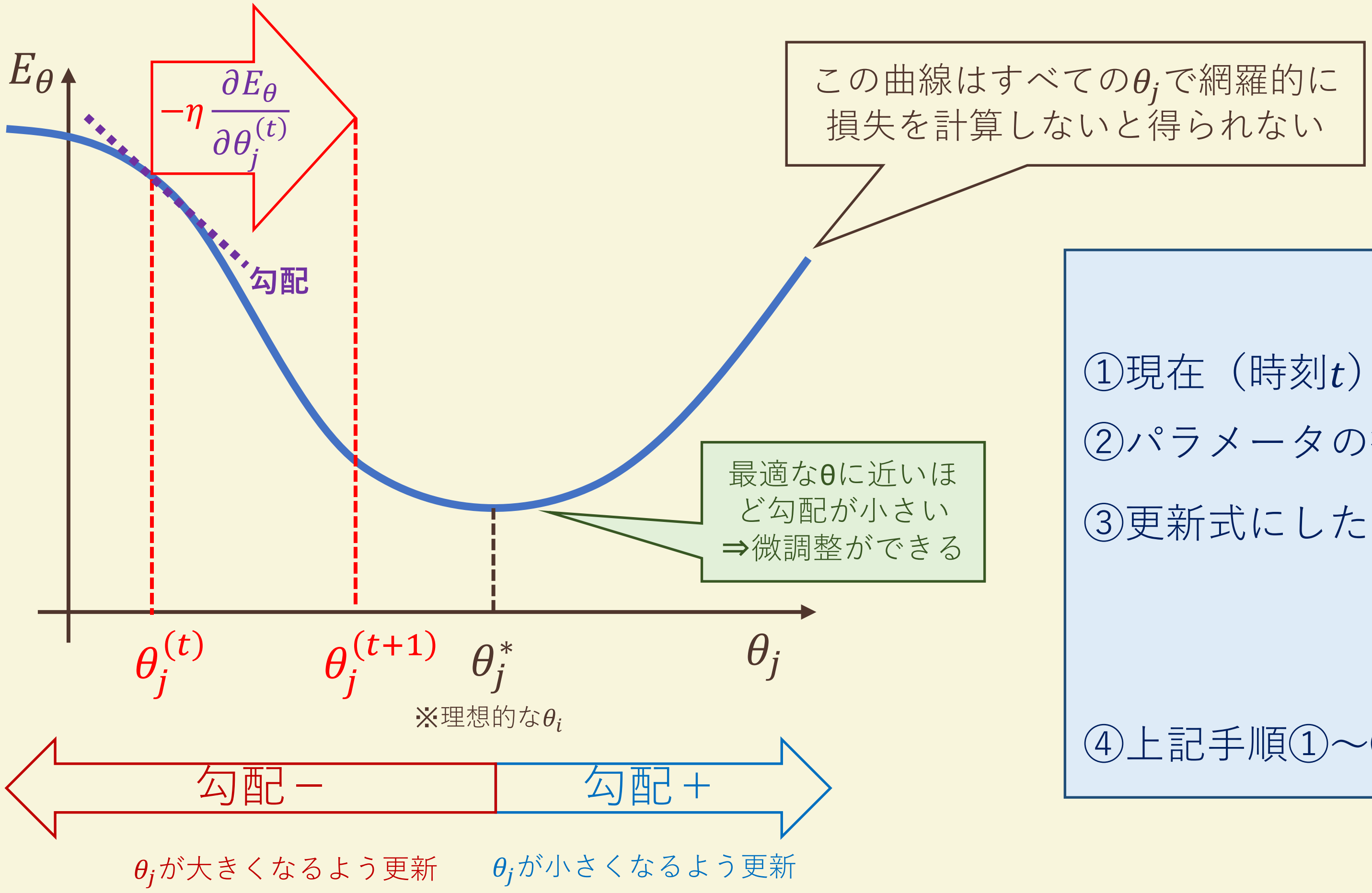
σ_l は**活性化関数**といい、配列の各要素を独立に処理

今回は $\mathrm{ReLU}(z) = \begin{cases} z \ (z > 0) \\ 0 \ (z \le 0) \end{cases}$ を採用

勾配降下法による重み・バイアスの最適化

✓漸化式に基づくパラメータ更新を繰り返して損失を小さくする

θ_j は W_l や b_l の要素



←NNはパラメータ数が膨大なため
すべての組み合わせを試すのは現実的ではない

勾配降下法

①現在（時刻t）のパラメータ$\theta_j^{(t)}$で損失E_θを計算

②パラメータの微小変動に対する損失の変化量$\dfrac{\partial E_\theta}{\partial \theta_j^{(t)}}$を計算

③更新式にしたがって時刻$t+1$のパラメータを計算

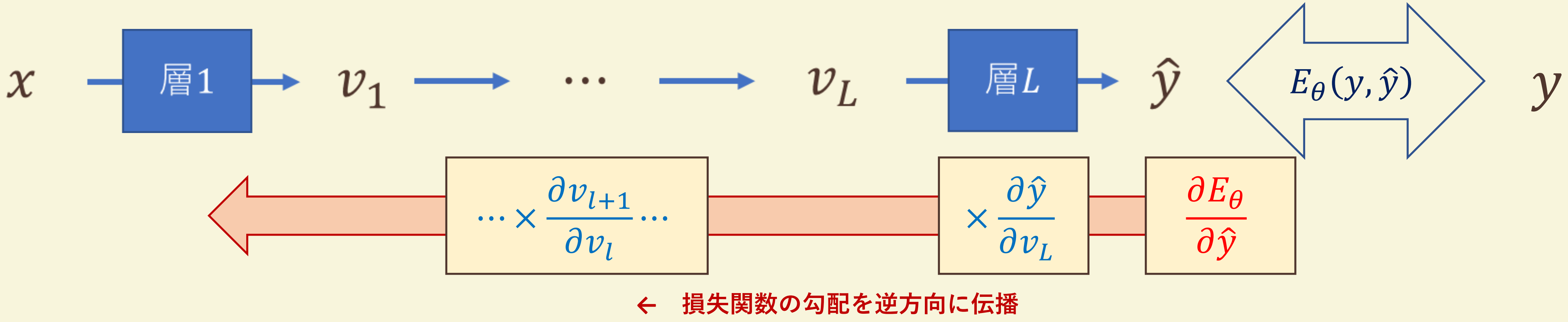
$$\theta_j^{(t+1)} = \theta_j^{(t)} - \eta \frac{\partial E_\theta}{\partial \theta_j^{(t)}}$$

（η：学習率）

④上記手順①～③を繰り返す

誤差逆伝播法による勾配計算

✓勾配降下法を使う上で必要な$\frac{\partial L_\theta}{\partial \theta_i}$を求める方法



θ_iが第 l 層のパラメータの場合

$$\frac{\partial E_\theta}{\partial \theta_i} = \frac{\partial E_\theta}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v_L} \cdot \frac{\partial v_L}{\partial v_{L-1}} \cdot \cdots \cdot \frac{\partial v_{l+2}}{\partial v_{l+1}} \cdot \frac{\partial v_{l+1}}{\partial \theta_i}$$

最終出力に対する
損失関数の勾配

各層の入力に対する
出力の勾配

θ_iが所属する層の
出力に与える影響

勾配と導関数を区別しよう

Q. $f(x) = x^2 + 1$ の $x = 3$ における勾配は？

導関数

$$f'(x) = 2x$$

x から勾配を求めるための**関数**

（$x = 3$ における）
勾配

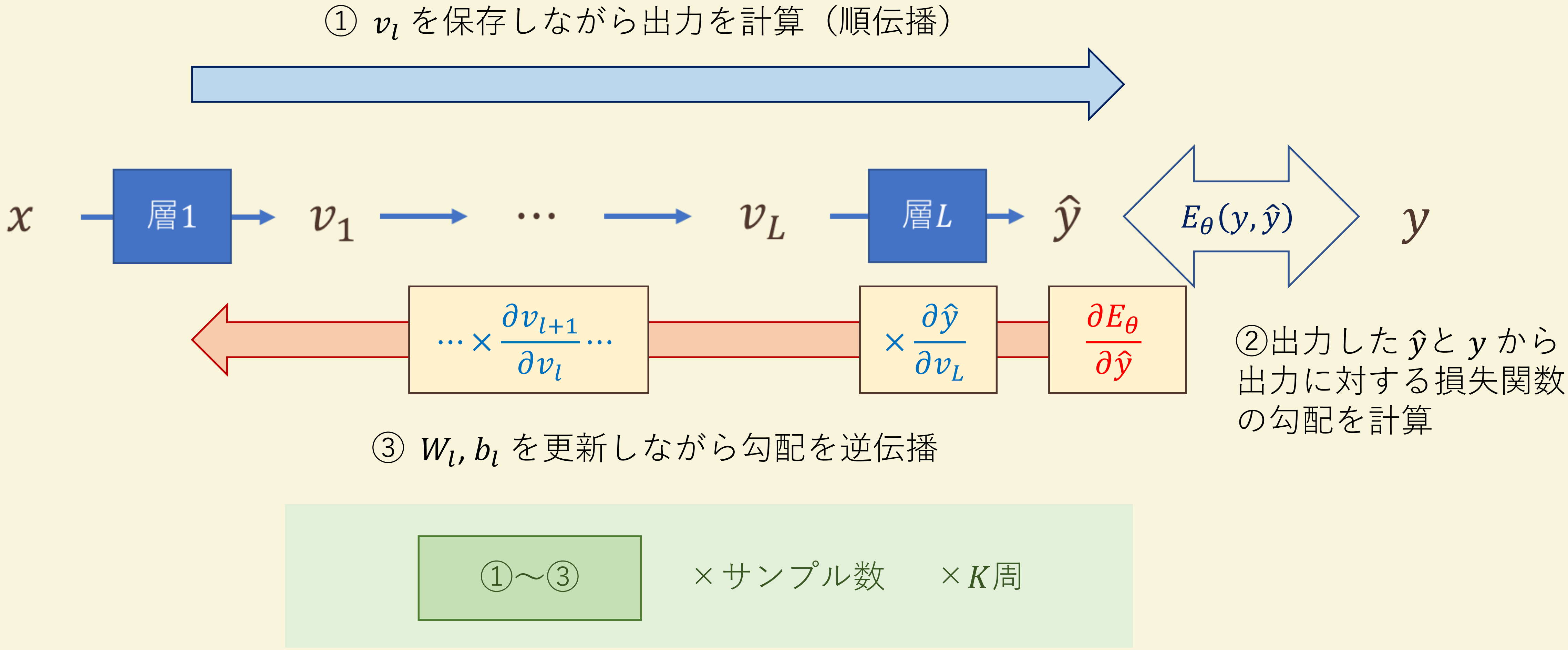
$$f'(3) = 2 \cdot 3 = 6$$

ある x における**傾きの値**

勾配を求めるには
関数の入力の値も必要に

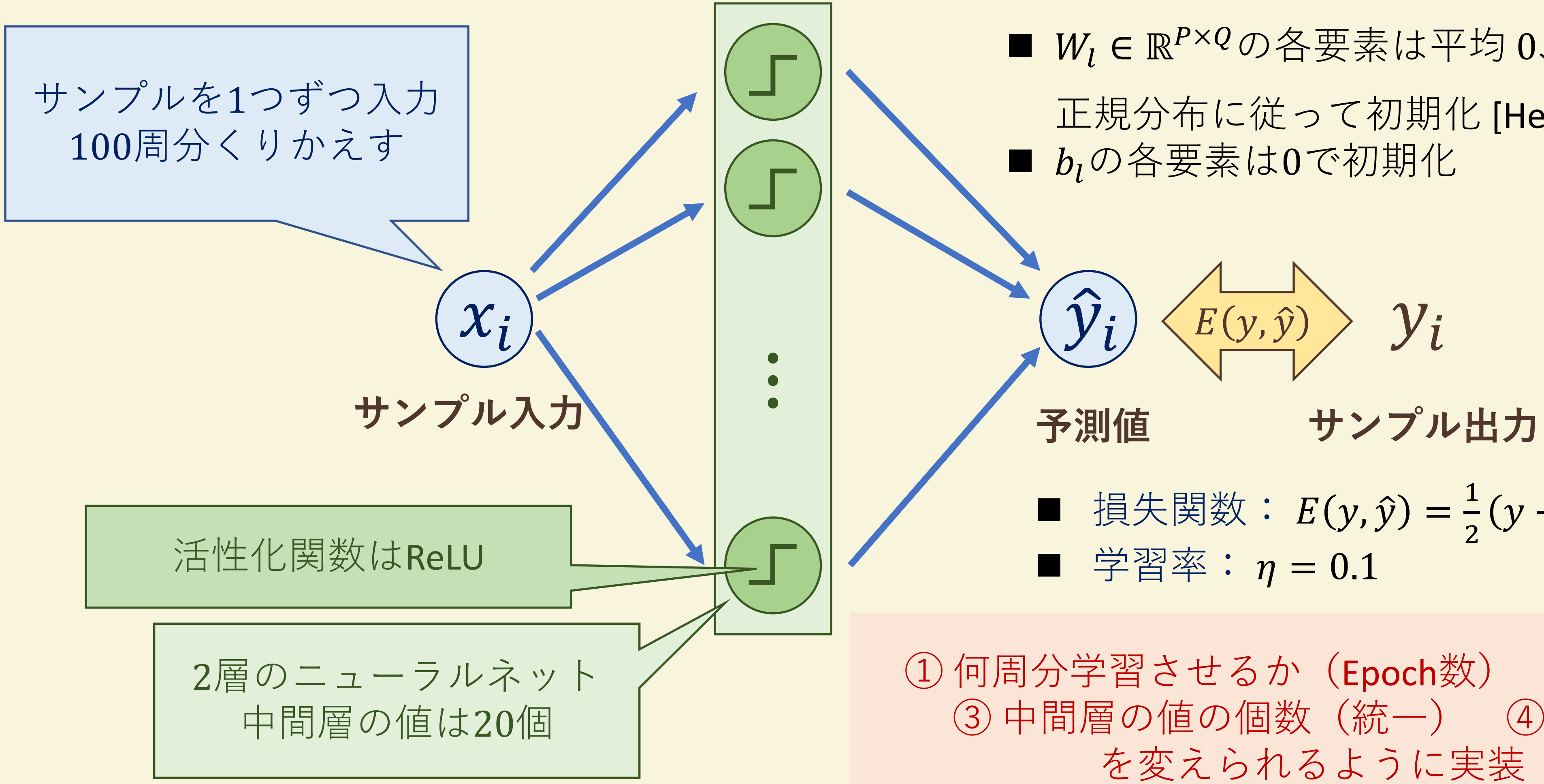
計算途中の値（v_l）も保存しておく

ニューラルネットワーク学習の流れ（まとめ）



実践編

実装してみよう



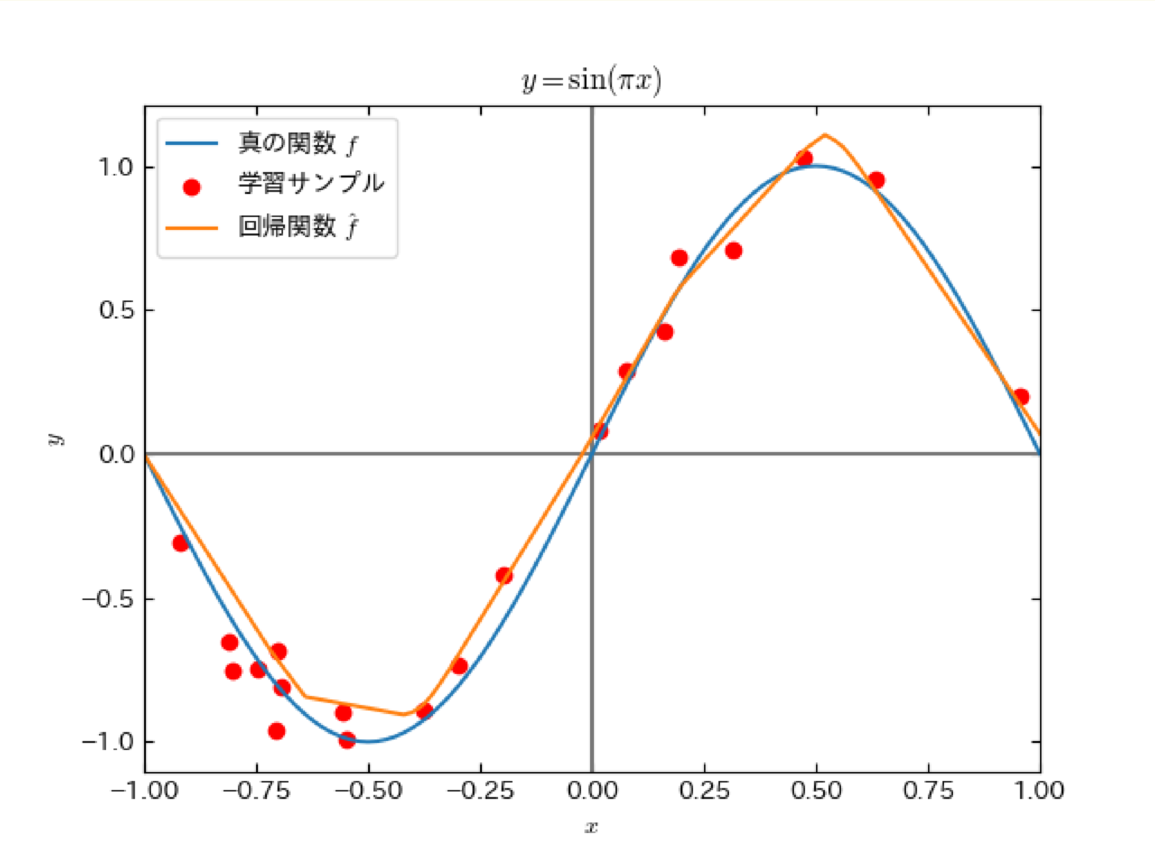
- $W_l \in \mathbb{R}^{P\times Q}$の各要素は平均 0、標準偏差$\sqrt{\frac{2}{Q}}$の正規分布に従って初期化 [Heの初期化]
- b_lの各要素は0で初期化

- 損失関数： $E(y,\hat{y}) = \frac{1}{2}(y-\hat{y})^2$
- 学習率： $\eta = 0.1$

① 何周分学習させるか（Epoch数）　② 層の数
③ 中間層の値の個数（統一）　④ 学習率
を変えられるように実装

He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

出力結果の例



$y = \sin(\pi x)$
真の関数 f
学習サンプル
回帰関数 $\hat{f}$
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
y

実装のヒント

逆伝播のための勾配

$$\frac{\partial E(y,\hat{y})}{\partial \hat{y}} = \hat{y} - y$$

$$\frac{\partial \mathrm{ReLU}(z)}{\partial z} = \begin{cases} 1 \ (z > 0) \\ 0 \ (z \leq 0) \end{cases}$$

実際は $z = 0$ で勾配は定義されないが
便宜上このように定義した

$$\frac{\partial}{\partial v_p}(Wv + b)_q = W_{q,p}$$

p, q は行列のインデックス

パラメータ関連の勾配

$$\frac{\partial}{\partial W_{p,q}}(Wv + b)_p = v_q$$

$$\frac{\partial}{\partial b_p}(Wv + b)_p = 1$$

- ✓ 勾配は行列計算に落とし込める場合も多い（図を描いて整理）
- ✓ ReLUの計算にはnp.where関数を使うと便利
- ✓ Layerクラスにforward（順伝播）とbackward（逆伝播＋値更新）の関数を用意すると実装しやすい