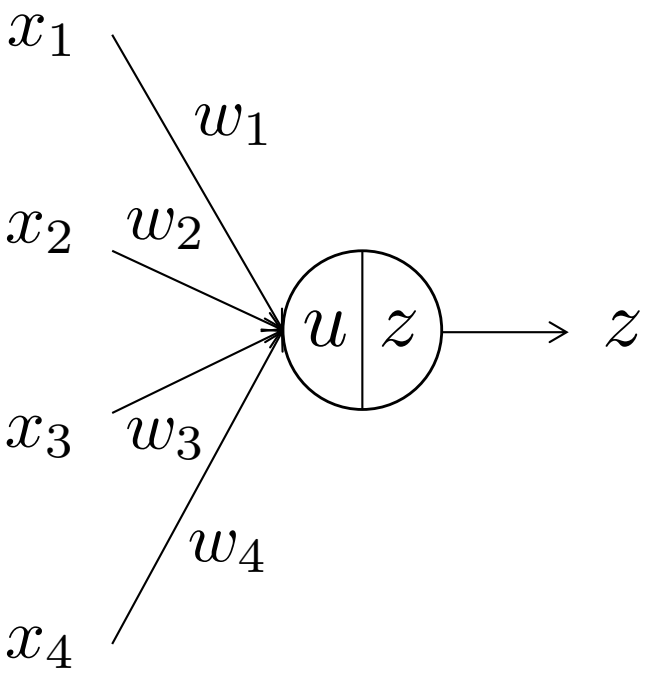
コンピュータビジョン
深層学習基礎 I

ユニットの働きと活性化関数

$$u = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

$$z = f(u)$$



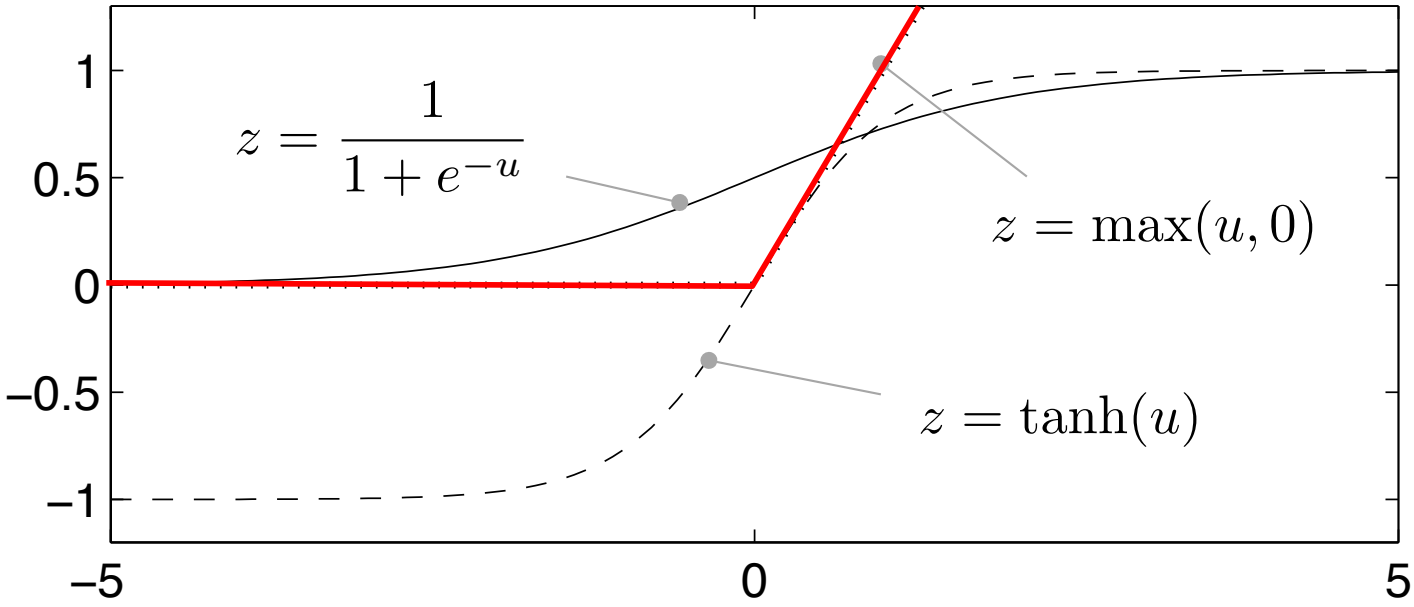
ユニットの働きと活性化関数

$$u = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

$$z = f(u)$$

ReLU: Rectified Linear Unit

pReLU: parametric -- [He+15]
Maxout [Goodfellow+13]



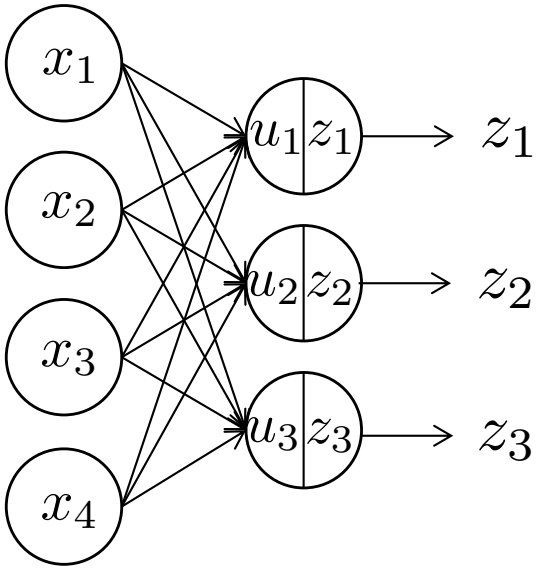
単層ネットワーク（全結合層）

$$u_j = \sum_{i=1}^{I} w_{ji} x_i + b_j$$

$$z_j = f(u_j)$$

$$\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{z} = \mathbf{f}(\mathbf{u})$$



$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_J \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_I \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_J \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_J \end{bmatrix},$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1I} \\ \vdots & \ddots & \vdots \\ w_{J1} & \cdots & w_{JI} \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} f(u_1) \\ \vdots \\ f(u_J) \end{bmatrix}$$

多層ネットワーク

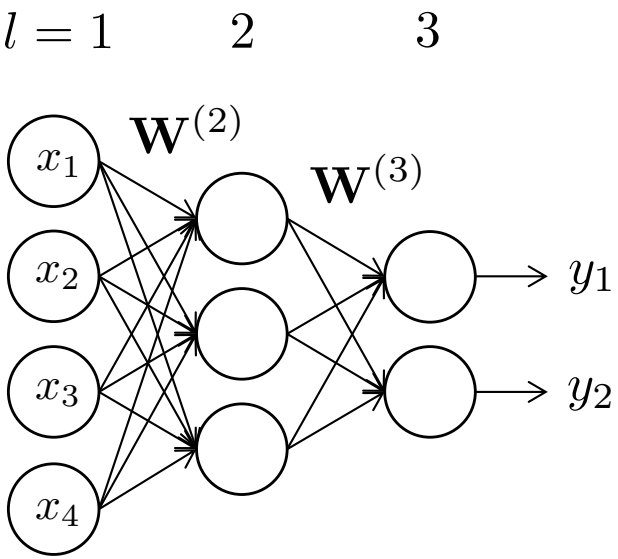
1層（入力層） $\mathbf{x} \equiv \mathbf{z}^{(1)}$

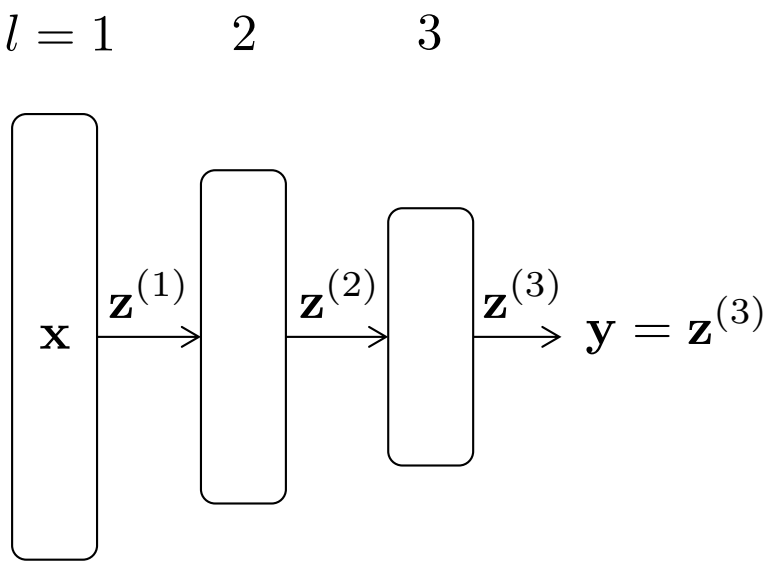
l → l+1 層への伝播

$$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}$$

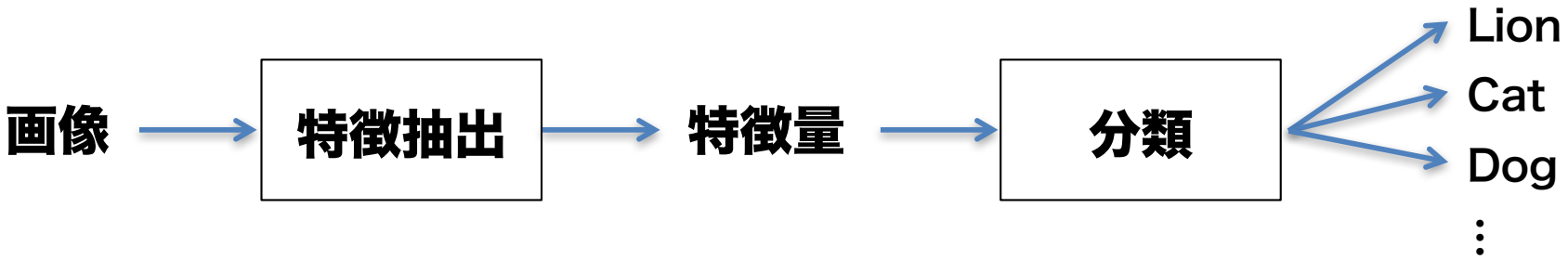
$$\mathbf{z}^{(l+1)} = \mathbf{f}\left(\mathbf{u}^{(l+1)}\right)$$

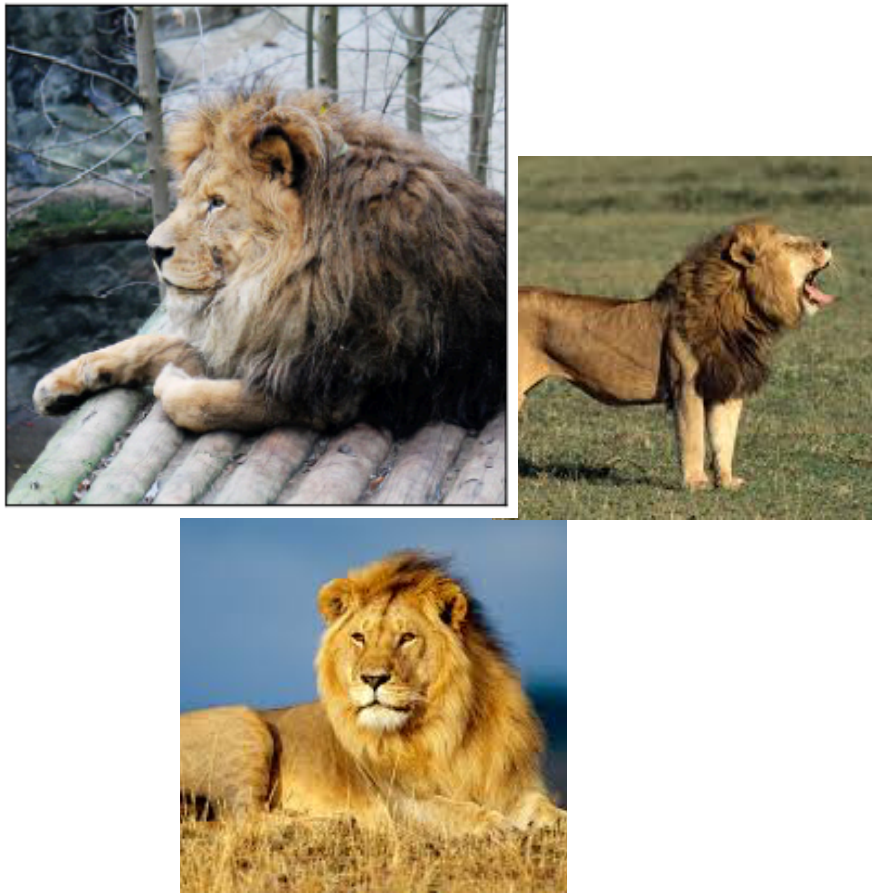
L層（出力層） $\mathbf{y} \equiv \mathbf{z}^{(L)}$





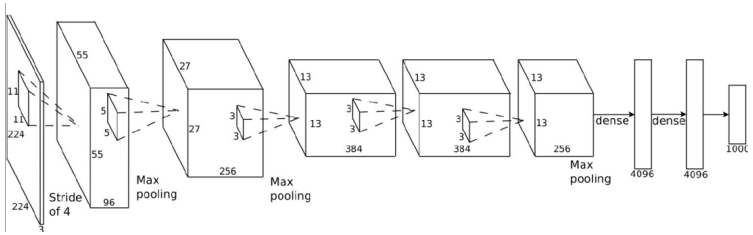
問題の難しさと深層学習

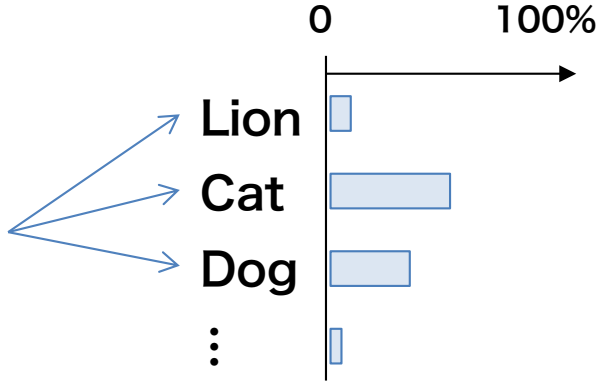




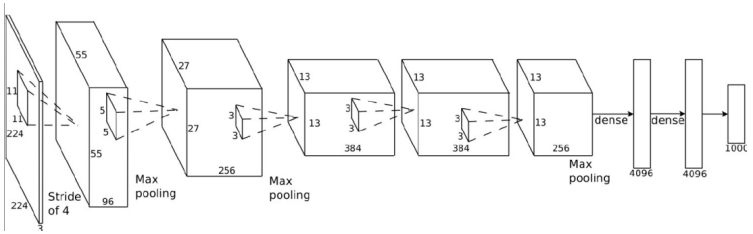
深層ニューラルネットの教師あり学習
(end-to-end)

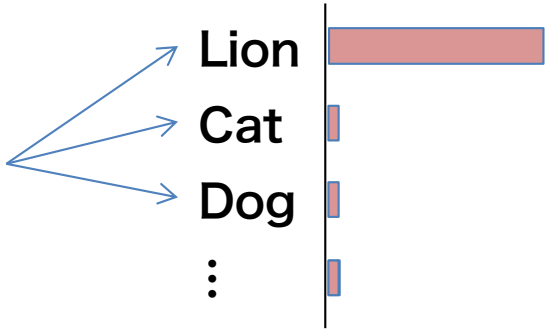
1. サンプルを入力し，結果を出力させる



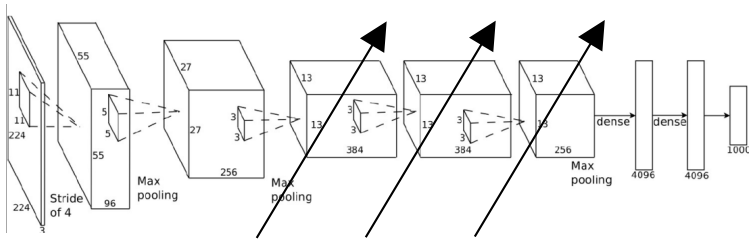


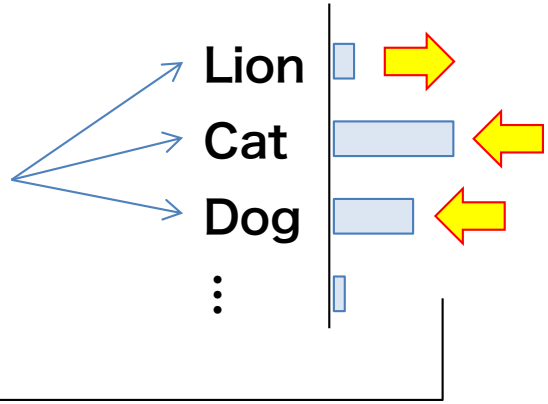
2. 正誤を評価（誤差を計算）





3. 重みを調節（誤差逆伝播法）





出力層の設計と誤差（損失）関数

クラス分類

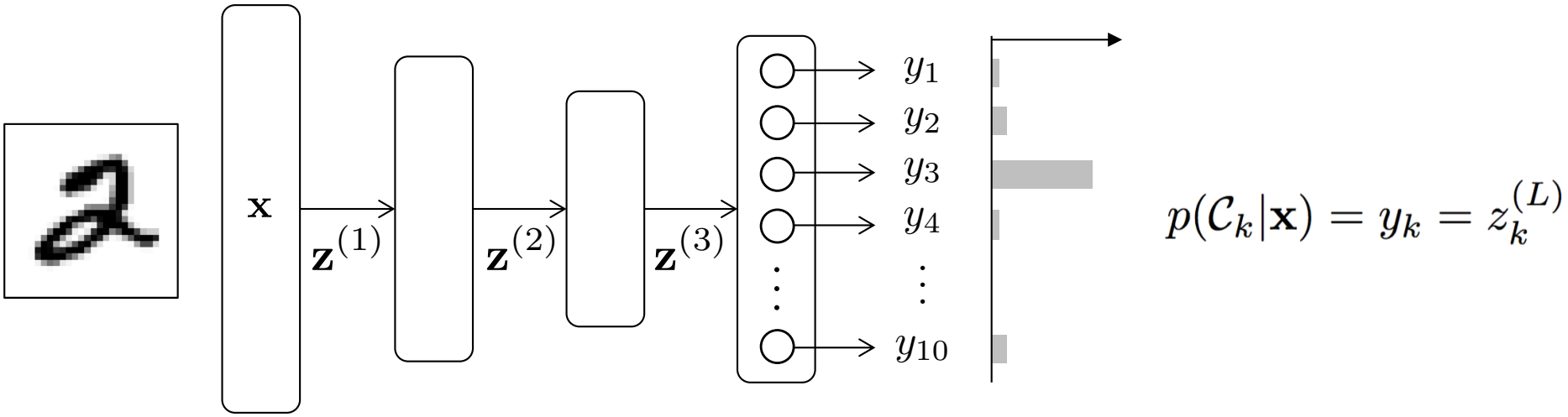
- クラス数と同数のユニット
- 活性化関数はソフトマックス
- 誤差は交差エントロピー

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log y_k(\mathbf{x}_n; \mathbf{w})$$

回帰

- 目標変数と同数のユニット
- 活性化関数は tanh や恒等写像
- 誤差は差の二乗和

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{d}_n - \mathbf{y}(\mathbf{x}_n; \mathbf{w})\|^2$$



$$p(\mathcal{C}_k|\mathbf{x}) = y_k = z_k^{(L)}$$

ソフトマックスと交差エントロピー

- 目標出力 d は正解クラスのみ１，それ以外は０をとるクラス数 K と同数の要素を持つベクトル（1-of-K符号化）

$$\mathbf{d} = [d_1, d_2, \ldots, d_K]$$

- 出力層にはソフトマックス活性化関数

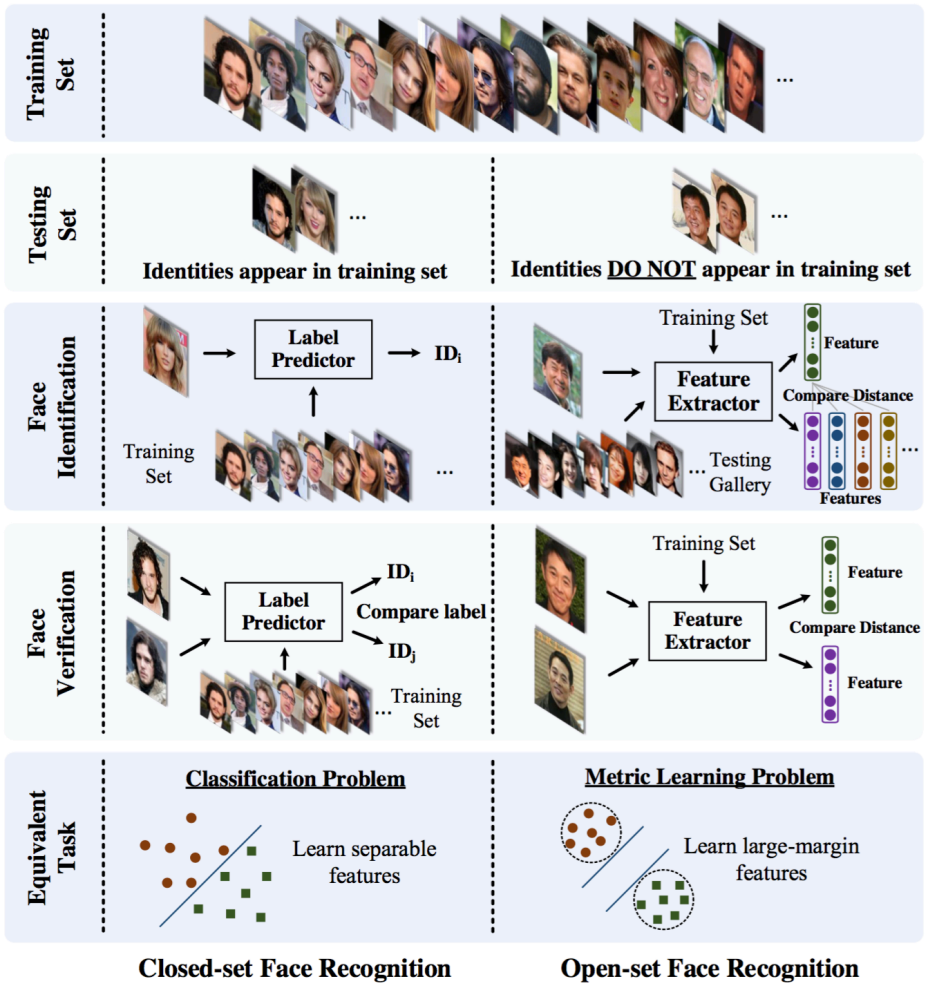
$$y_k \equiv z_k^{(L)} = \frac{\exp(u_k^{(L)})}{\sum_{j=1}^{K} \exp(u_j^{(L)})} \quad \left(\sum_{k=1}^{K} y_k = 1 \right)$$

- 各ユニットの出力は各クラスの事後確率であると解釈
  - ユニットの総和は１
- ネットの出力と目標出力の差（誤差）：交差エントロピー

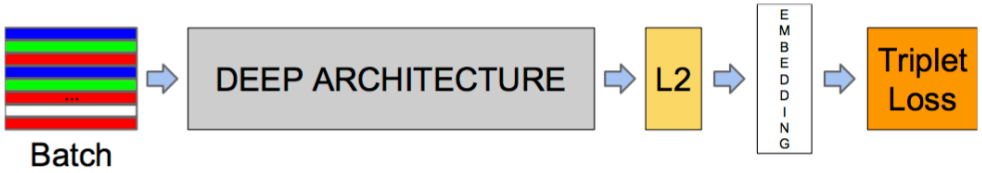
$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} d_{nk} \log y_k(\mathbf{x}_n; \mathbf{w})$$

距離計量学習（Metric Learning）

- 顔認識のように学習とテストでクラスラベルを共有しない場合



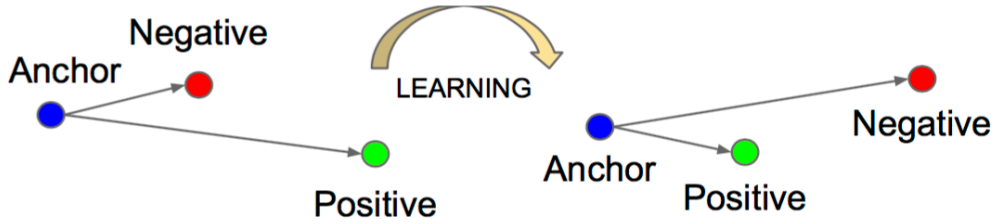
[Liu+, SphereFace, 2017]



[Schroff+, DeepFace, 2015]

Tripletロス：

$$L = \|f(x_a) - f(x_p)\|_2^2 - \|f(x_a) - f(x_n)\|_2^2$$



[Schroff+, DeepFace, 2015]

Contrastiveロス：

$$L = \begin{cases} \|f(x_i) - f(x_j)\|_2^2, & \text{if } i \text{ and } j \text{ same identity} \\ \max(0, m - \|f(x_i) - f(x_j)\|_2)^2, & \text{otherwise} \end{cases}$$

勾配降下法による学習

- E(w) の勾配方向にパラメータを繰り返し小修正

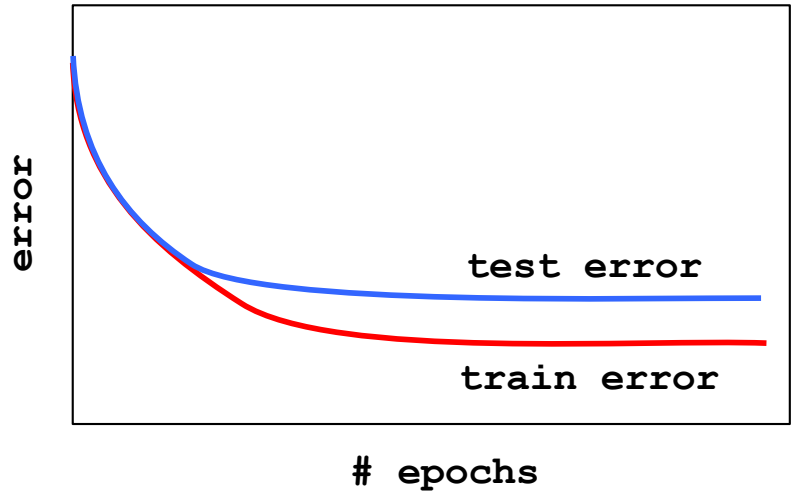
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E$$

ϵ：学習係数（学習率）

$$\nabla E \equiv \frac{dE}{d\mathbf{w}} = \left[\frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_M} \right]^{\top}$$

パラメータ（重み）はランダムに初期化

- 学習に使っていないサンプルで汎化性能を予測評価



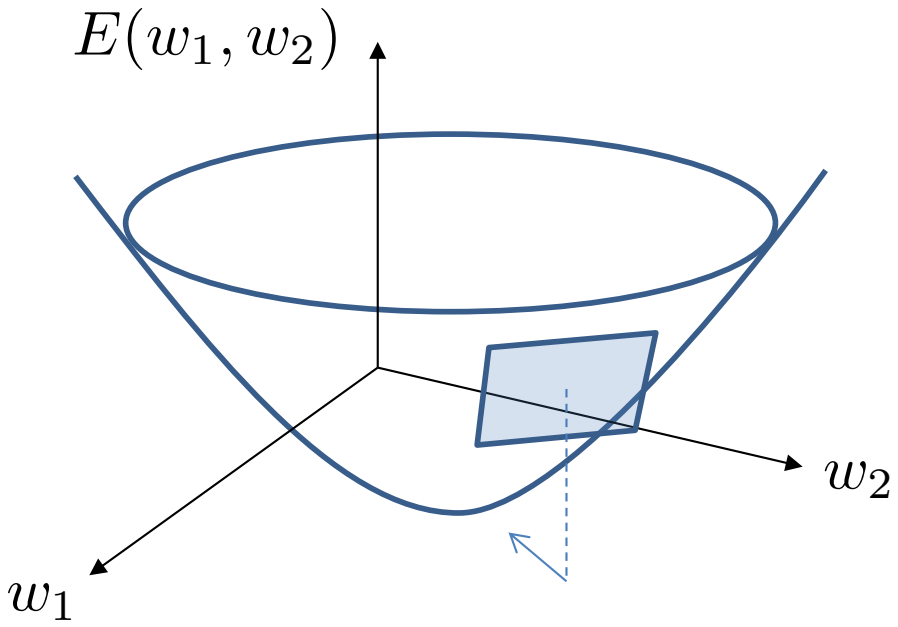
勾配降下法による最小化

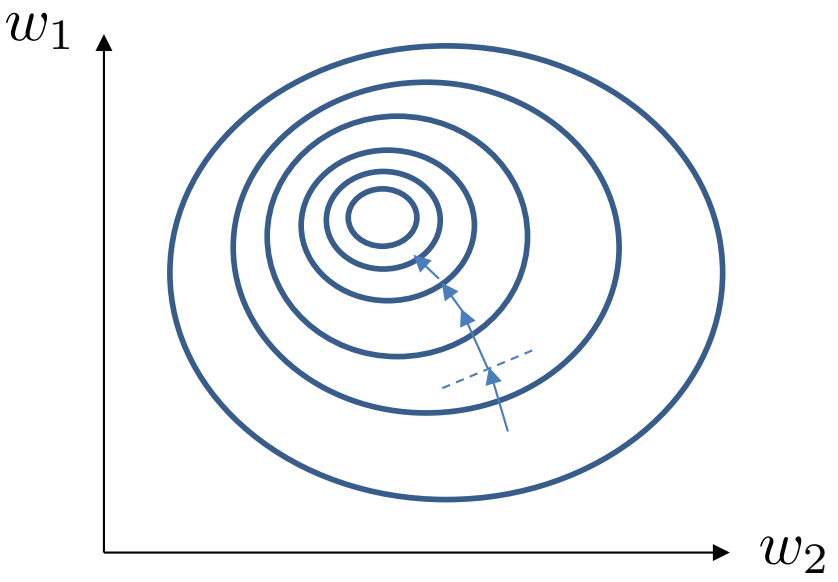
- *E*(w) の勾配方向にパラメータを繰り返し小修正

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E$$

ε：学習係数（学習率）

$$\nabla E \equiv \frac{dE}{d\mathbf{w}} = \left[\frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_M} \right]^\top$$





勾配の計算

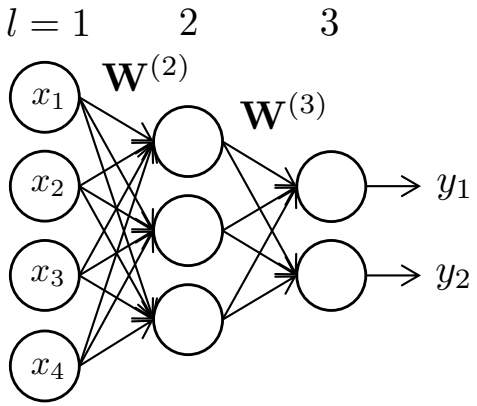
- 勾配の計算は大変

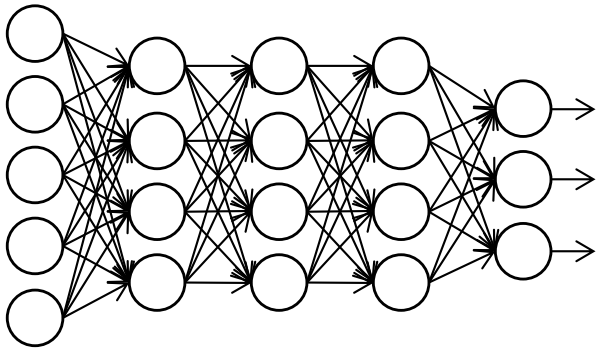
$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = (\mathbf{y}(\mathbf{x}_n) - \mathbf{d}_n)^\top \frac{\partial \mathbf{y}}{\partial w_{ji}^{(l)}}$$

２乗誤差の場合

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{d}_n - \mathbf{y}(\mathbf{x}_n; \mathbf{w})\|^2$$

$$\begin{aligned}
\mathbf{y}(\mathbf{x}) &= \mathbf{f}(\mathbf{u}^{(L)}) \\
&= \mathbf{f}(\mathbf{W}^{(L)}\mathbf{z}^{(L-1)} + \mathbf{b}^{(L)}) \\
&= \mathbf{f}(\mathbf{W}^{(L)}\mathbf{f}(\mathbf{W}^{(L-1)}\mathbf{z}^{(L-2)} + \mathbf{b}^{(L-1)}) + \mathbf{b}^{L}) \\
&= \mathbf{f}(\mathbf{W}^{(L)}\mathbf{f}(\mathbf{W}^{(L-1)}\mathbf{f}(\cdots\mathbf{f}(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)})\cdots)) + \mathbf{b}^{(L)})
\end{aligned}$$

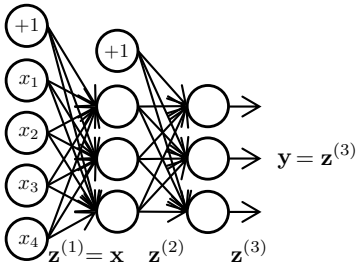


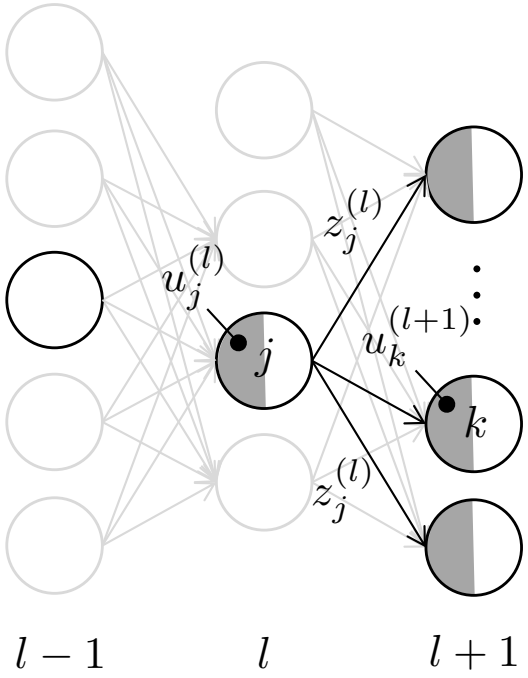


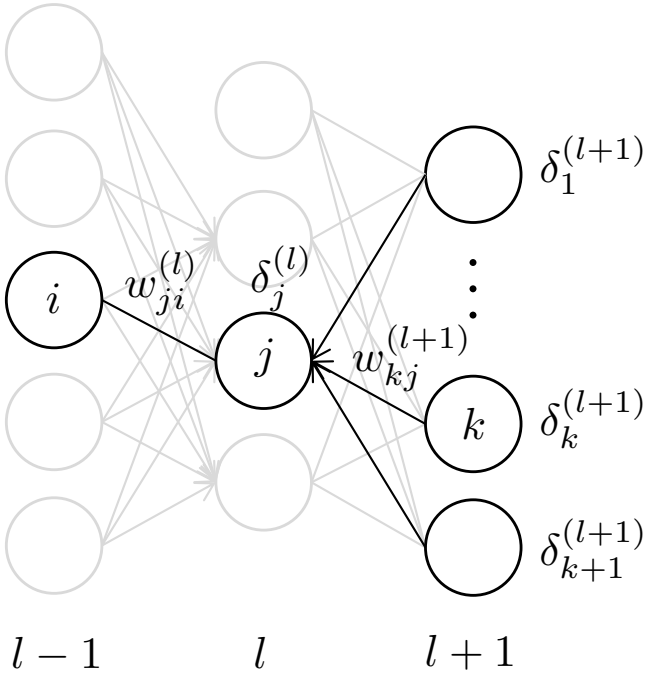
誤差逆伝播
(BP: Back Propagation)

”デルタ”を定義： $\delta_j^{(l)} \equiv \dfrac{\partial E_n}{\partial u_j^{(l)}}$ ⇒ 勾配は $\dfrac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$

各層のデルタは逆伝播可能： $\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} \left(w_{kj}^{(l+1)} f'(u_j^{(l)}) \right)$

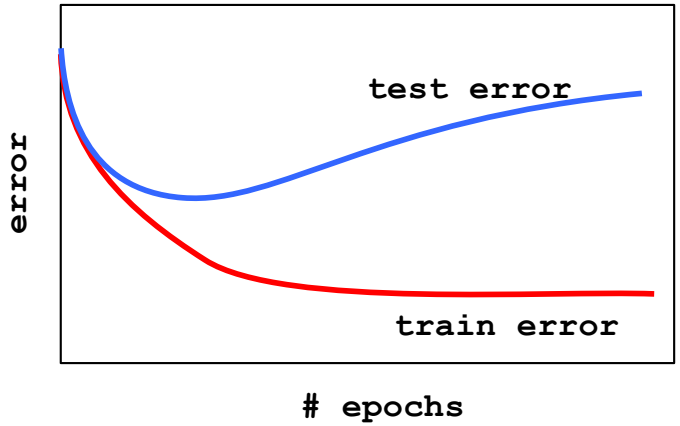




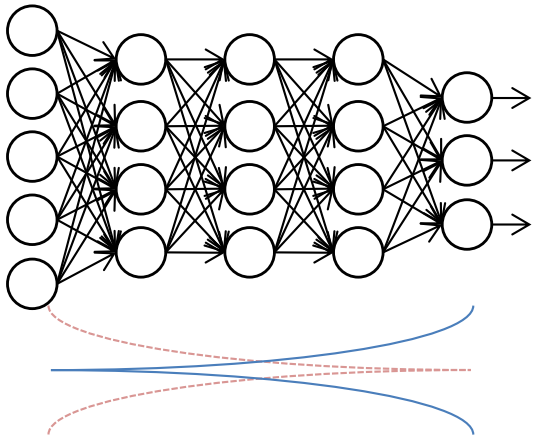


ディープネットの学習の困難さ

- 多層になると過適合（過学習）・ローカルミニマム



- 勾配消失（vanishing gradient）問題
  - デルタが急速に小さく，あるいは大きくなり制御できない



確率的勾配降下法
(SGD: Stochastic Gradient Descent)

バッチ学習

- 全サンプルについての誤差の総和を最小化

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E$$

確率的勾配降下

- サンプル１個に関する誤差を最小化することを，サンプルをランダムに選びなおして繰り返す

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E_n$$

毎回（tごとに）異なるEnを最小化していることに

ミニバッチとモメンタム

- ミニバッチ：数百個まで程度のサンプル集合ごとにパラメータ更新

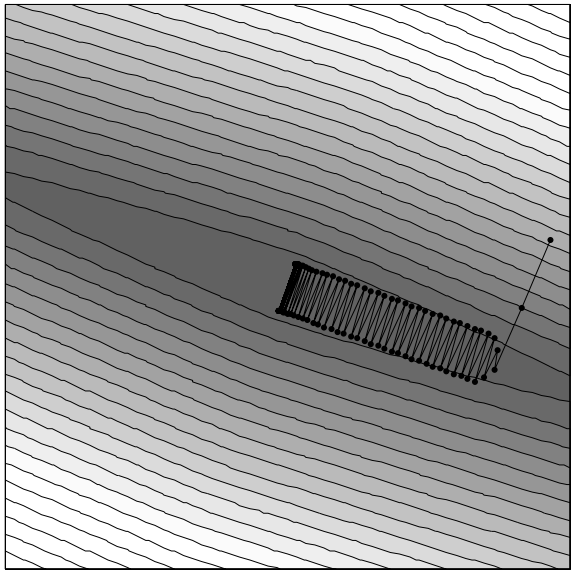
$$E_t(\mathbf{w}) = \frac{1}{N_t} \sum_{n \in \mathcal{D}_t} E_n(\mathbf{w})$$

- モメンタム：前回修正量の何割か(0.5~0.9)を今回修正量に加算

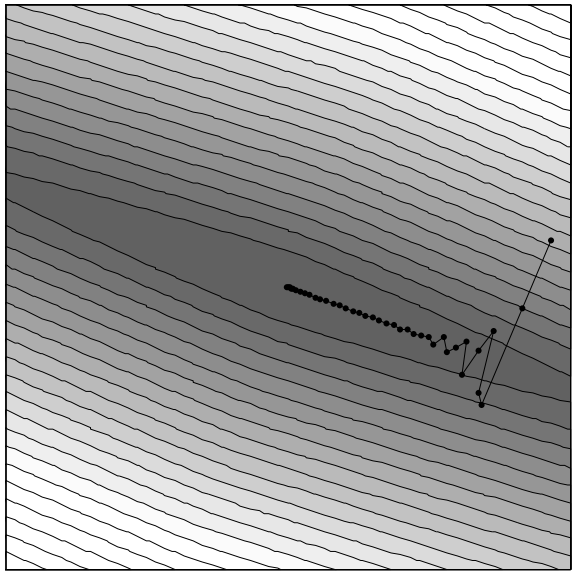
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E_t + \mu \Delta \mathbf{w}^{(t-1)}$$

前回修正量

$$\Delta \mathbf{w}^{(t-1)} \equiv \mathbf{w}^{(t-1)} - \mathbf{w}^{(t-2)}$$



モメンタムなし



モメンタムあり

重み減衰（Weight decay）

- 重みの冗長性を縛る制約（正則化）
- 際限なく重みが発散することを防ぐ
- λは普通ごく小さい値

$$E_t(\mathbf{w}) \equiv \frac{1}{N_t} \sum_{n \in \mathcal{D}_t} E_n(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

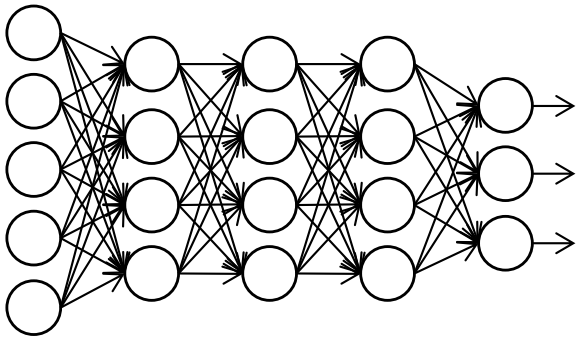
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \left(\frac{1}{N_t} \sum \nabla E_n + \lambda \mathbf{w}^{(t)} \right)$$

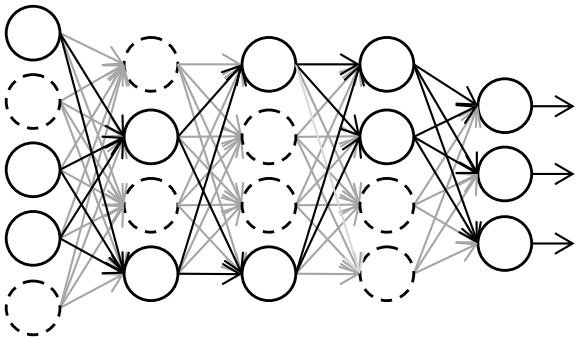
表 **3.1**　異なる正則化使用時の手書き数字認識 MNIST の分類精度の比較（文献 [17] から抜粋）.

正則化の種類	テストデータ分類誤差 (%)
重み減衰	1.62
重み上限	1.35
ドロップアウト+重み減衰	1.25
ドロップアウト+重み上限	1.05

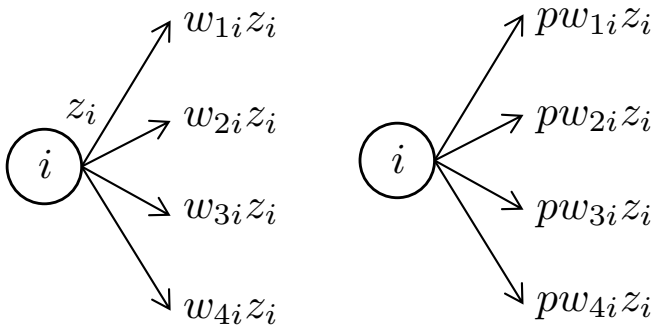
Dropout [Hinton+12]

- 学習時一定の確率(p)で中間層のユニットを「無効化」





- 予測時：ユニット出力を p 倍
  - 学習時の補償



- 過適合を避けるために過剰な自由度を制限する正則化
- 予測時には複数の異なるモデルの合成（⇒性能向上）と解釈も可能

重みの初期化

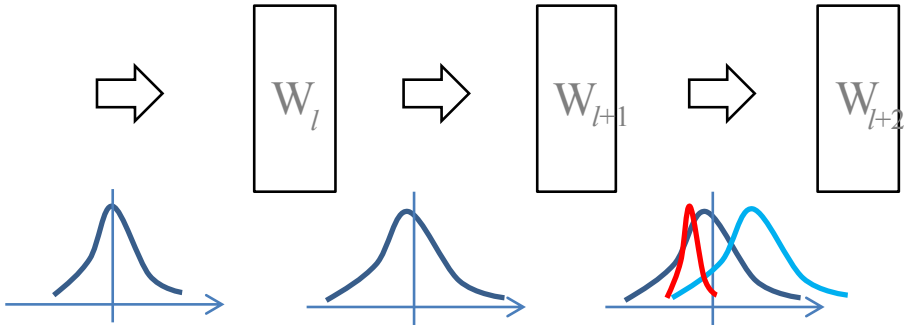
“Xavier” initialization [Glorot-Bengio10], PReLU[He+15]

- 重み w_{ji} はガウス分布に従う乱数で生成　　$w_{ji} \sim N(0, \sigma^2)$

  ➔ σをどう選ぶ？

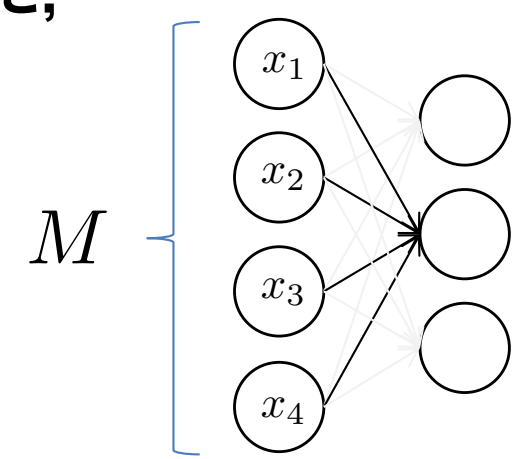
lからl+1層の順伝播計算：

$$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}$$
$$\mathbf{z}^{(l+1)} = f(\mathbf{u}^{(l+1)})$$



- $z^{(l)}$ と $z^{(l+1)}$ の分散が同じオーダであるために,
  - “Xavier” initialization

$$\mathrm{Var}(w) = 1/M \quad \Rightarrow \quad w \sim N(0, 1/\sqrt{M})$$



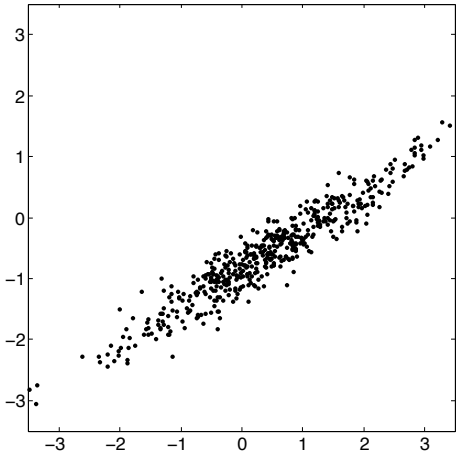
- f がReLUの場合($E(z)\neq 0$)
  - ”MSRA” initialization

$$\mathrm{Var}(u^{(l+1)}) = \frac{1}{2} M \mathrm{Var}(w^{(l+1)}) \mathrm{Var}(u^{(l)})$$

特徴量の正規化（標準化）

$$\mathbf{x}_n = [x_{n1}, x_{n2}, \ldots, x_{nI}]^\top$$

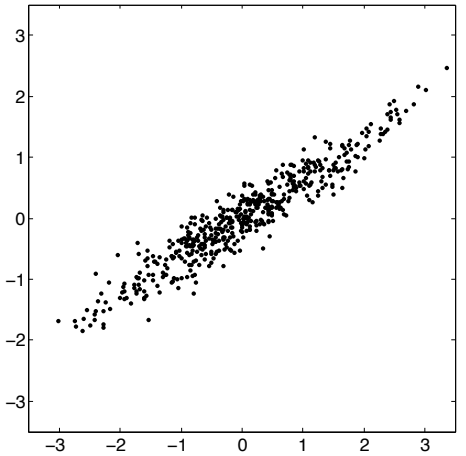
0）入力サンプルの分布



1）平均を差し引く

$$x_{ni} \leftarrow x_{ni} - \bar{x}_i$$

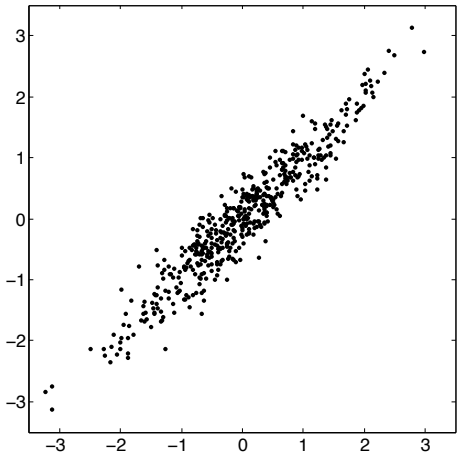
$$\bar{x}_i \equiv \sum_{n=1}^{N} x_{ni}/N$$



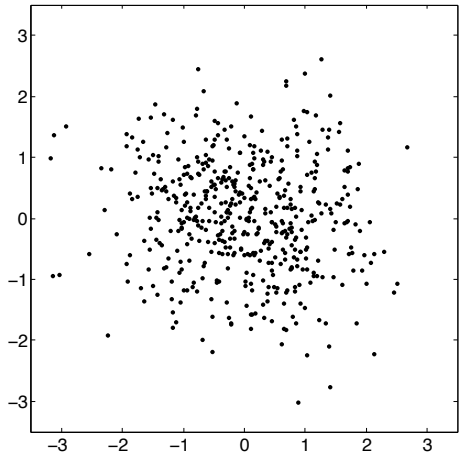
2）標準偏差で割る

$$x_{ni} \leftarrow \frac{x_{ni} - \bar{x}_i}{\sigma_i}$$

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_{ni} - \bar{x}_i)^2}$$



3）白色化成分間の相関をなくす



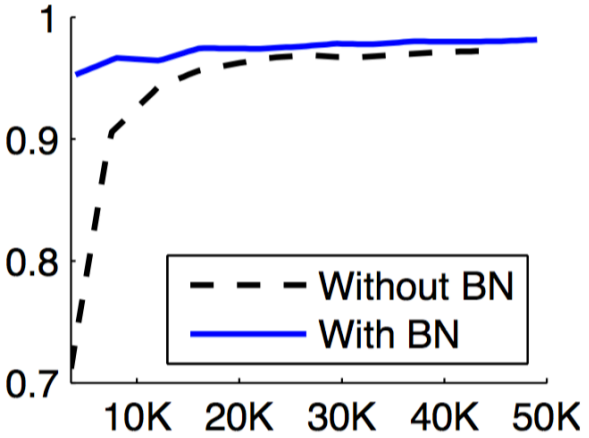
バッチ正規化

Ioffe+, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015

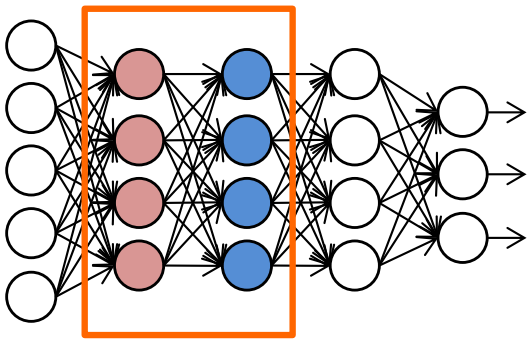
- ミニバッチ単位で各層出力の平均と分散を揃える

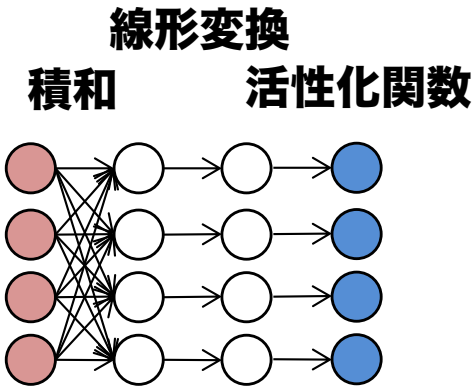
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$



- 収束性能（速度と到達点）が向上
- 線形変換層の挿入で表現可





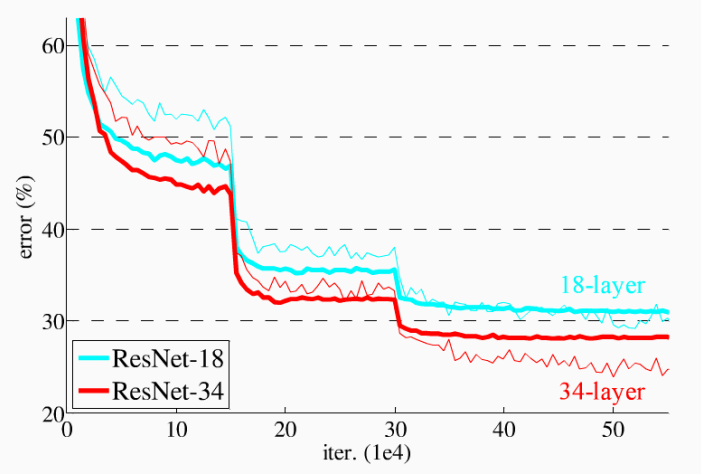
学習係数の制御

- 最も重要なハイパーパラメータ：学習係数(ϵ)
- 勾配降下法の解の更新のステップ幅
- 値の選択は，基本的に試行錯誤
  - タスク，学習データ依存

- 経験則１：学習の進捗につれて値を小さくする
  - 手動（バリデーション誤差をみながら1/10に）
  - 自動：機械的に行うこともある

$$\epsilon_t = \epsilon_0 - \alpha t$$

- 経験則２：各層で重みの更新速度が同じペースになるように



確率的勾配降下法の改良

- **標準的方法**

$$\theta_t \leftarrow \theta_{t-1} - \alpha g_t$$

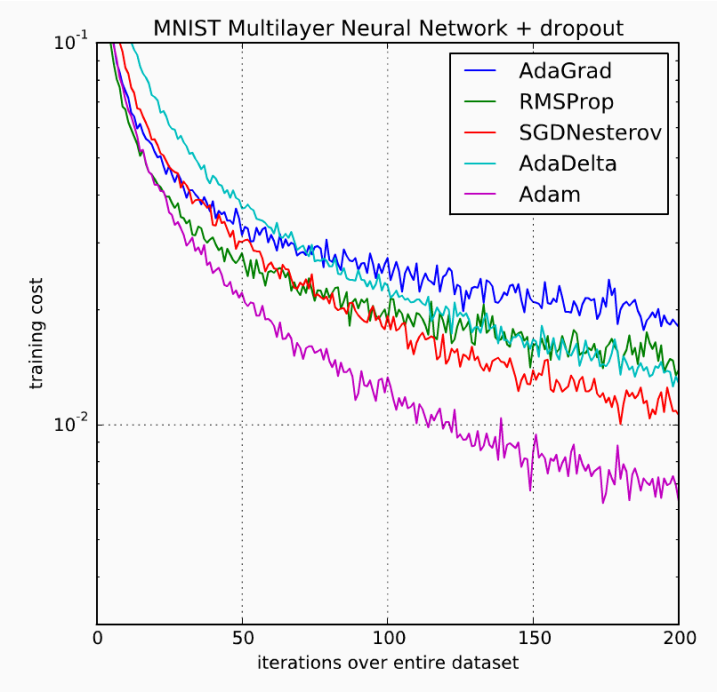
- **AdaGrad**

(adaptive gradient) [Duchi+2011]

$$\theta_t = \theta_{t-1} - \alpha g_t / \sqrt{\sum_{i=1}^{t} g_i^2}$$

- **ADAM**

(adaptive moment) [Kigma-Ba2015]



$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)