JDLA E資格認定プログラム

全人類がわかる ディープラーニングコース DAY 2

全人類がわかる統計学 Presents

https://to-kei.net

全コース紹介

DAY 1 順伝播型	DAY 2 前処理と工夫	DAY 3 データの学習
DAY 4 CNN	DAY 5 RNN	DAY 6 応用モデル練習

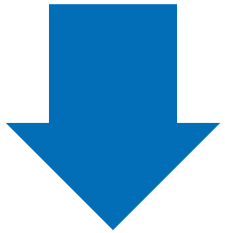
目次

ディープラーニング体系講座 DAY 2

正則化とは

機械学習の全般の問題点

- 学習データだけではなく新しい入力に対しても性能を発揮する（汎化性能が高い）アルゴリズムをどのようにして作るか
- 学習データに特化し過ぎると汎化性能が失われる（過学習）
- 学習が不十分だと、十分な性能が得られない



汎化性能と精度の高さを両立させたい

機械学習と正則化

- 汎化性能と予測精度の高さを両立させる処理を正則化という
- 複雑な深層学習モデルは過学習に陥りやすい
- 正則化には様々な手法が存在する

正則化の意義

- 例えば回帰問題における平均二乗誤差を考えると、以下の形に分解できる
（入力x、モデルによる出力y、訓練データ集合D、正解h）

$$\mathbb{E}_D[\{y(x;D)-h(x)\}^2] = \{\mathbb{E}_D[y(x;D)]-h(x)\}^2 + \mathbb{E}_D[\{y(x;D)-\mathbb{E}_D[y(x;D)]\}^2]$$

平均二乗誤差

予測値の平均にどの
程度誤差があるのか
（バイアス）

データの取り方による
予測のブレ
（バリアンス）

- これをバイアス・バリアンス分解という

正則化の意義

$$\mathbb{E}_D[\{y(x;D)-h(x)\}^2]$$

平均二乗誤差

$$=\{\mathbb{E}_D[y(x;D)]-h(x)\}^2+\mathbb{E}_D[\{y(x;D)-\mathbb{E}_D[y(x;D)]\}^2]$$

予測値の平均にどの程度誤差があるのか（バイアス）

データの取り方による予測のブレ（バリアンス）

- バリアンスが大きい、つまり過学習（過剰適合）している状態から、モデルによる予測値をできるだけブレないようにするのが正則化
- バリアンスを小さくするには、モデルの複雑さに制限をかければいい

ディープラーニング体系講座 DAY 2

パラメータノルムペナルティー

パラメータノルムペナルティー

- 正則化のアプローチの一つ
- 目的関数 $J(\theta; X, y)$ に対しノルムペナルティ $\Omega(\theta)$ を追加する

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \boldsymbol{\alpha\Omega(\theta)}$$

正則化項

- $\boldsymbol{\alpha} \in [\mathbf{0}, \infty)$ ・・・ノルムペナルティ項の重み (0 ：正則化なし)
- 深層学習では、一般にユニットの重みパラメータのノルムで正則化を行う
  （バイアスに対しては正則化を加えない）

ノルムとは

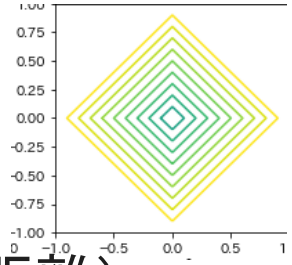
$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ とすると、Lpノルムは

$$\|\boldsymbol{x}\|\boldsymbol{p} = (\sum_{k=1}^{n} |x_k|^{\boldsymbol{p}})^{\frac{1}{\boldsymbol{p}}}$$

と表される

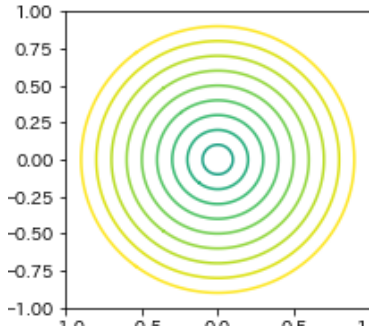
- L1ノルム（マンハッタン距離）

$$\|\boldsymbol{x}\|_1 = \sum_{k=1}^{n} |x_k|$$



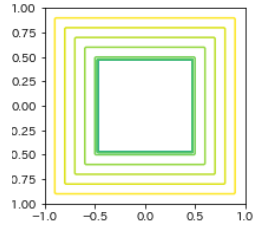
- L2ノルム（ユークリッド距離）

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{k=1}^{n} x_k^2}$$



- L∞ノルム（maxノルム、最大絶対値）

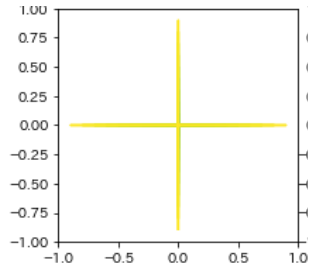
$$\|x\|_\infty \simeq \max_k |x_k|$$



- L0ノルム（非ゼロの個数）

$$\|x\|_0 \simeq \sum_{k=1}^{n} \delta(x_k)$$

$$\delta(i) = \begin{cases} 1 & i \neq 0 \\ 0 & i = 0 \end{cases}$$



ノルムとは

$x = (x_1, x_2, \ldots, x_n)$ とする。

- Lpノルム

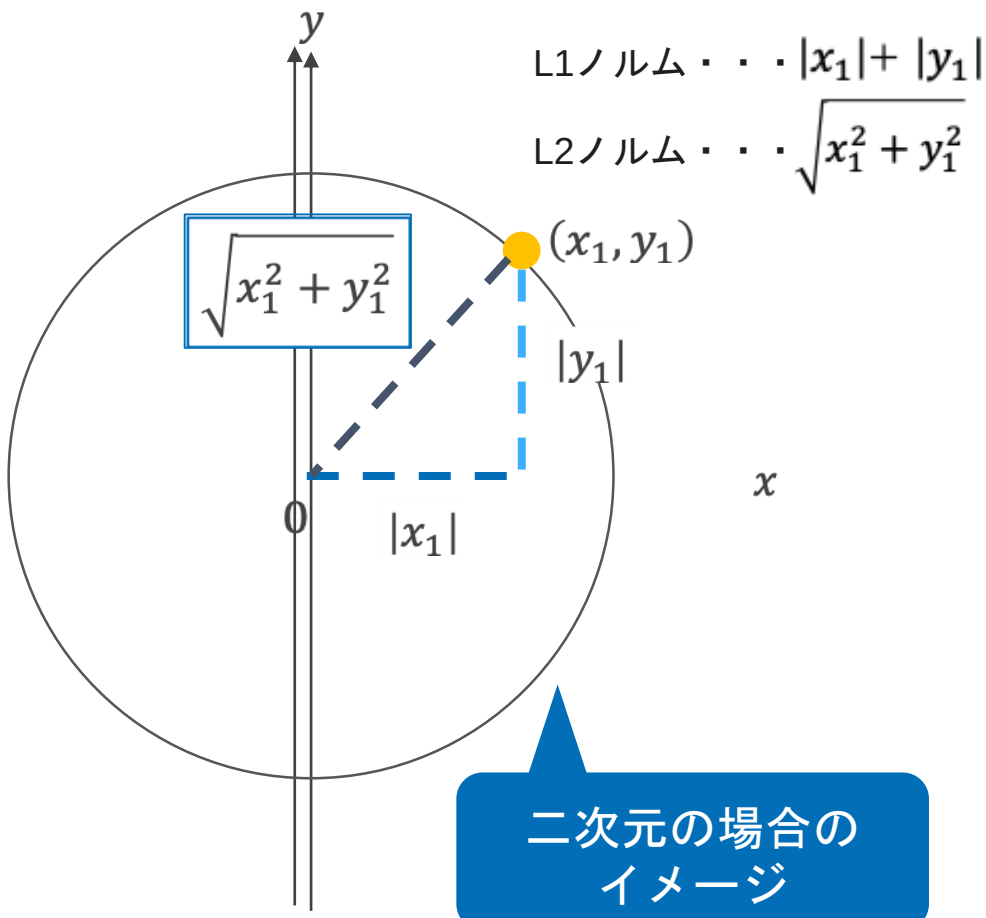
$$\|x\|p = (\sum_{k=1}^{n} |x_k|^p)^{\frac{1}{p}}$$

- L1ノルム (マンハッタン距離)

$$\|x\|_1 = \sum_{k=1}^{n} |x_k|$$

- L2ノルム(ユークリッド距離)

$$\|x\|_2 = \sqrt{\sum_{k=1}^{n} x_k^2}$$



回帰と正則化項の組み合わせ

- Lasso（ラッソ）回帰 —— ペナルティ項にL1正則化を用いる

$$\lambda E(w_0, w_1, \ldots, w_n) = \lambda \sum_{k=1}^{n} |w_k|$$

- Ridge（リッジ）回帰 —— ペナルティ項にL2正則化を用いる

$$\lambda E(w_0, w_1, \ldots, w_n) = \frac{\lambda}{2} \sum_{k=1}^{n} w_k^2$$

- Elastic Net ———— L1正則化とL2正則化を両方用いたもの
- 係数 λを大きくすれば、正則化の作用を強めることができる

L2正則化

- 重みパラメータのL2ノルムを正則化項として加える
- 重み減衰（weight decay）と言われる
- 重みが全体的に小さくなる方向に進んでいく

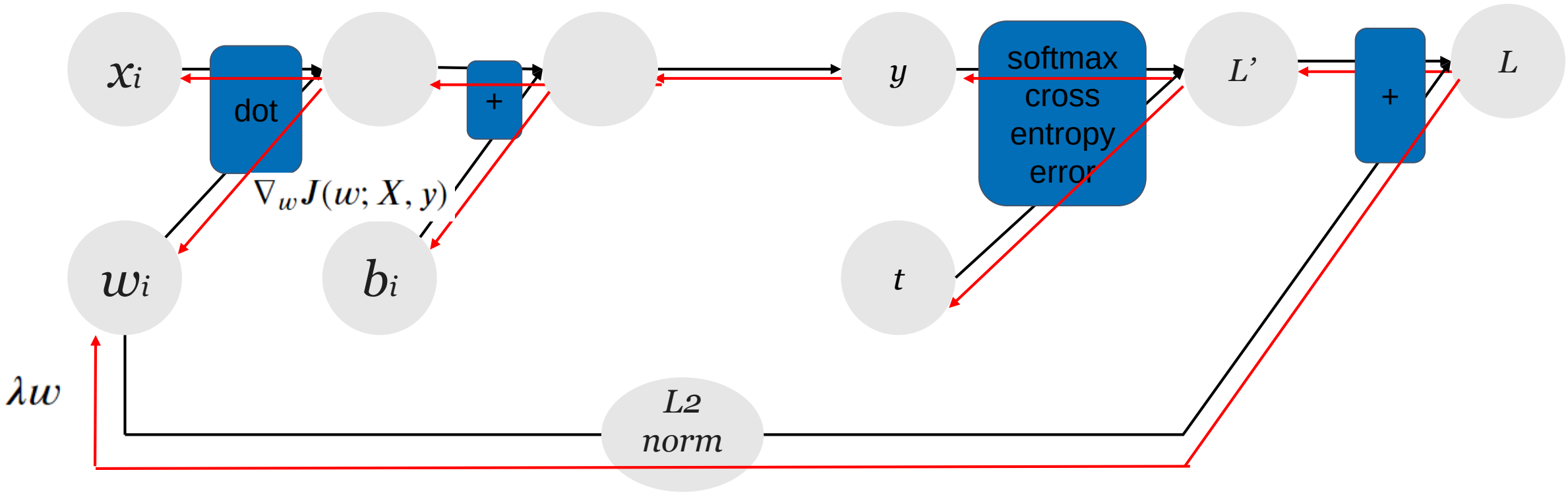
$$\tilde{J}(w; X, y) = \frac{\lambda}{2}\|w\|_2^2 + J(w; X, y)$$

$$\nabla_w \tilde{J}(x; X, y) = \boxed{\lambda w} + \nabla_w J(w; X, y)$$

勾配にλwが加わる

L2正則化

- 計算グラフで考えると



L1正則化

- 重みパラメータのL1ノルムを正則化項として加える
- 重みのうちいくつかが0に近づいていく
- Lasso回帰が特徴量選択の意義を持つのと同様に、重要な重みだけを選択することに相当する

$$\tilde{J}(w; X, y) = \lambda \|w\|_1 + J(w; X, y)$$

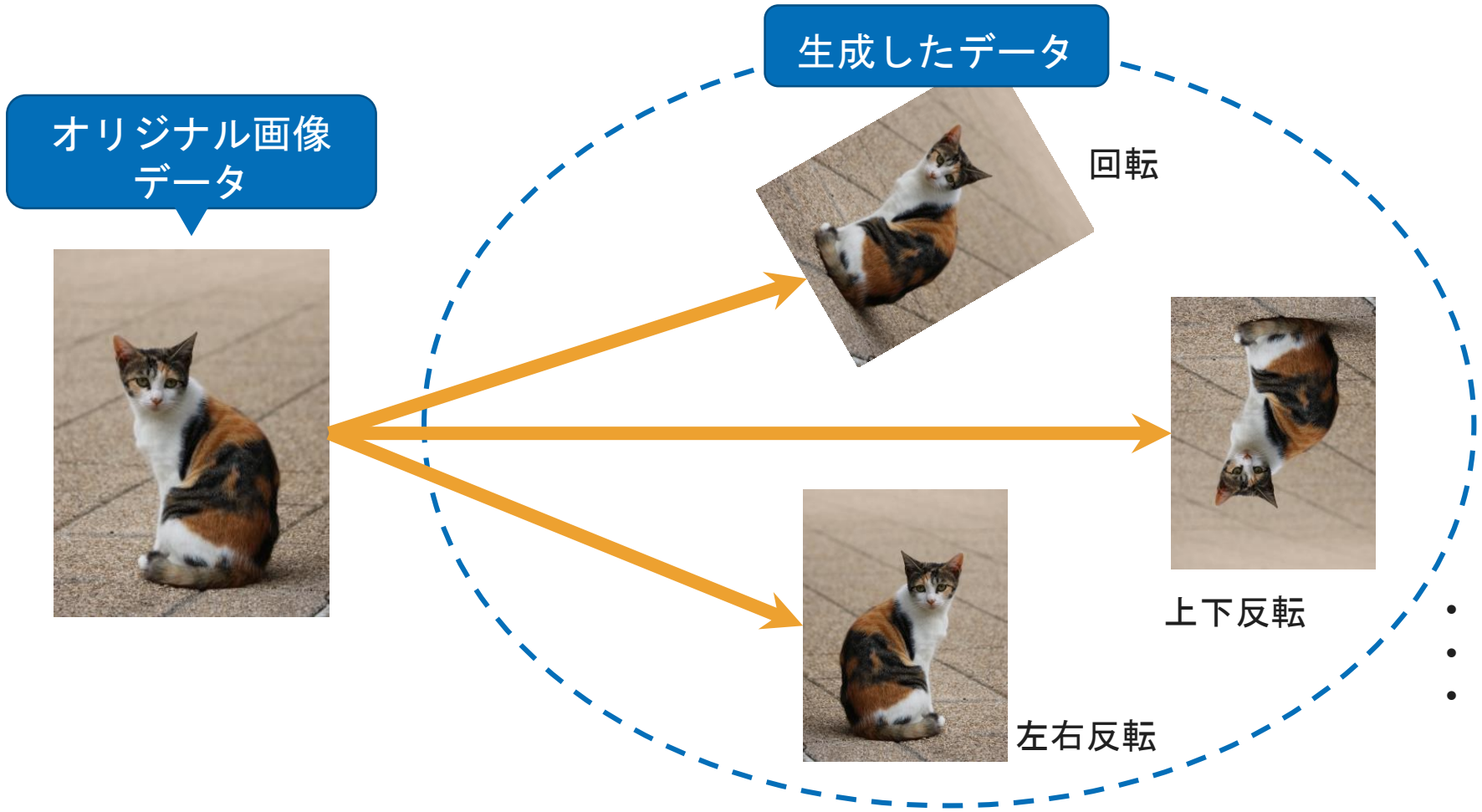
ディープラーニング体系講座 DAY 2

データ集合の拡張

機械学習におけるデータの生成

- もともとあるデータから新しいデータを作る
- データの不足を補うアプローチとして有効な方法の一つ
- 画像認識の分野で古くから行われている
  - 既存の画像から新しい画像を生成する
  - 新しく生成した画像をもとに新たなデータを作る

画像認識における拡張データの生成例



- 色調の変化
- サイズ変更
- ランダムノイズの混入

などの方法も

主な画像データの拡張手法

- horizontal flip ——— 画像を水平方向に回転させる
- vertical flip – 画像を垂直方向に回転させる
- random crop ——— 一枚の画像から指定された大きさでランダムに切り出す
- scale augmentation ——— 画像のスケールを変化させてから切り出しを行う
- random rotation —— 画像をランダムに回転させる
- cutout ——— 画像の一部にマスクをかける
- random erasing —— 画像内の箇所をランダムに選び画像を消去する
- elastic distortion —— 画像をいくつかの領域に細分化しそれぞれランダムに移動させる

elastic distortion（弾性変形）

- データ拡張を行う手法の一つ
- 手書き文字認識などのように形状変化が生じる問題を対象
- 形状に変形を加えてデータ拡張させる
  - バイリニア補完
  - バイキュービック補完
- 着目画素に対して乱数で移動量を決める
- 手書き文字で生じそうな形状の変化を加えることが出来る

elastic distortionの適用例

画像データの生成で注意すべき点

- 画像と文字では意味が違ってくる場合がある
- 文字の場合「６」と「９」、「b」と「d」は意味が違う
- このようなタスクでは180度回転は不適切

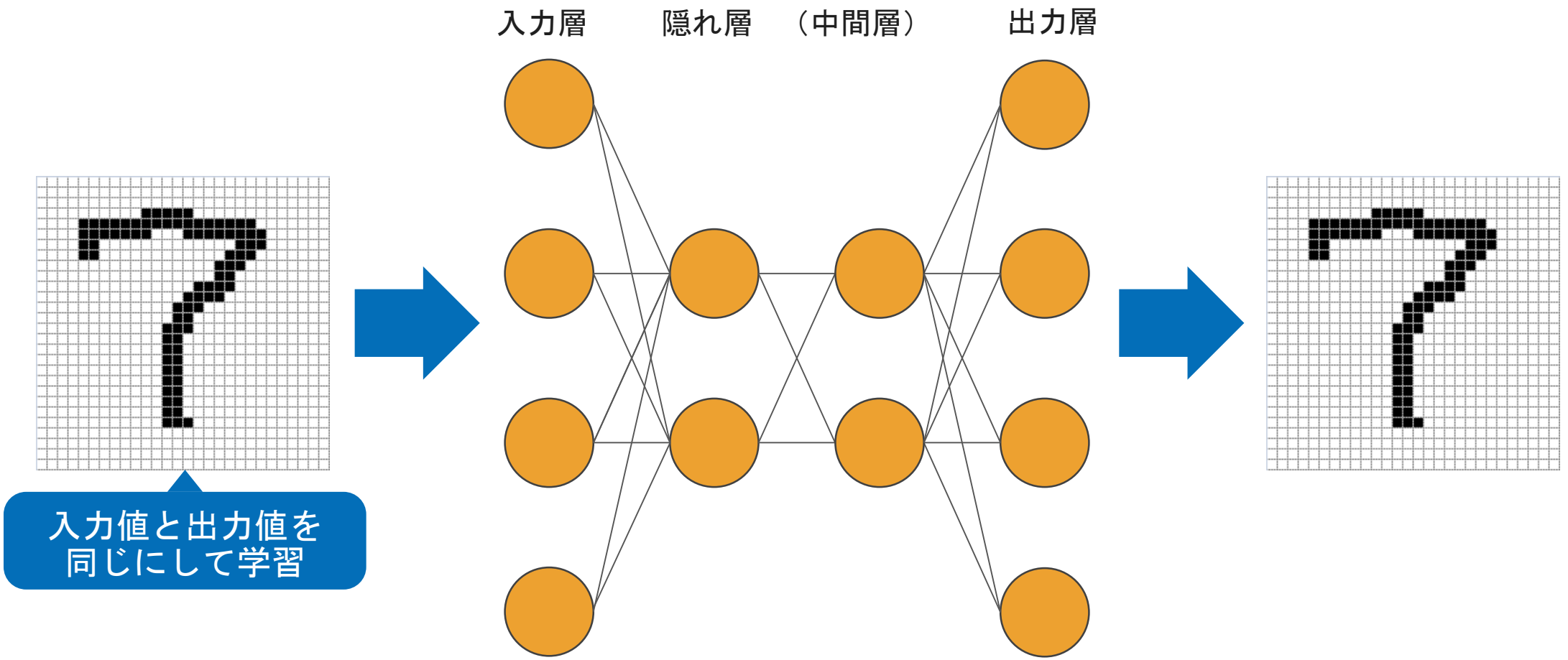
ノイズの注入

- 通常ニューラルネットワークはノイズに弱い
- 学習用データにあえてノイズを混入させることで精度を上げる
- データ集合の拡張の手法の一つ
- 深層学習ではデノイジング・オートエンコーダなどの手法

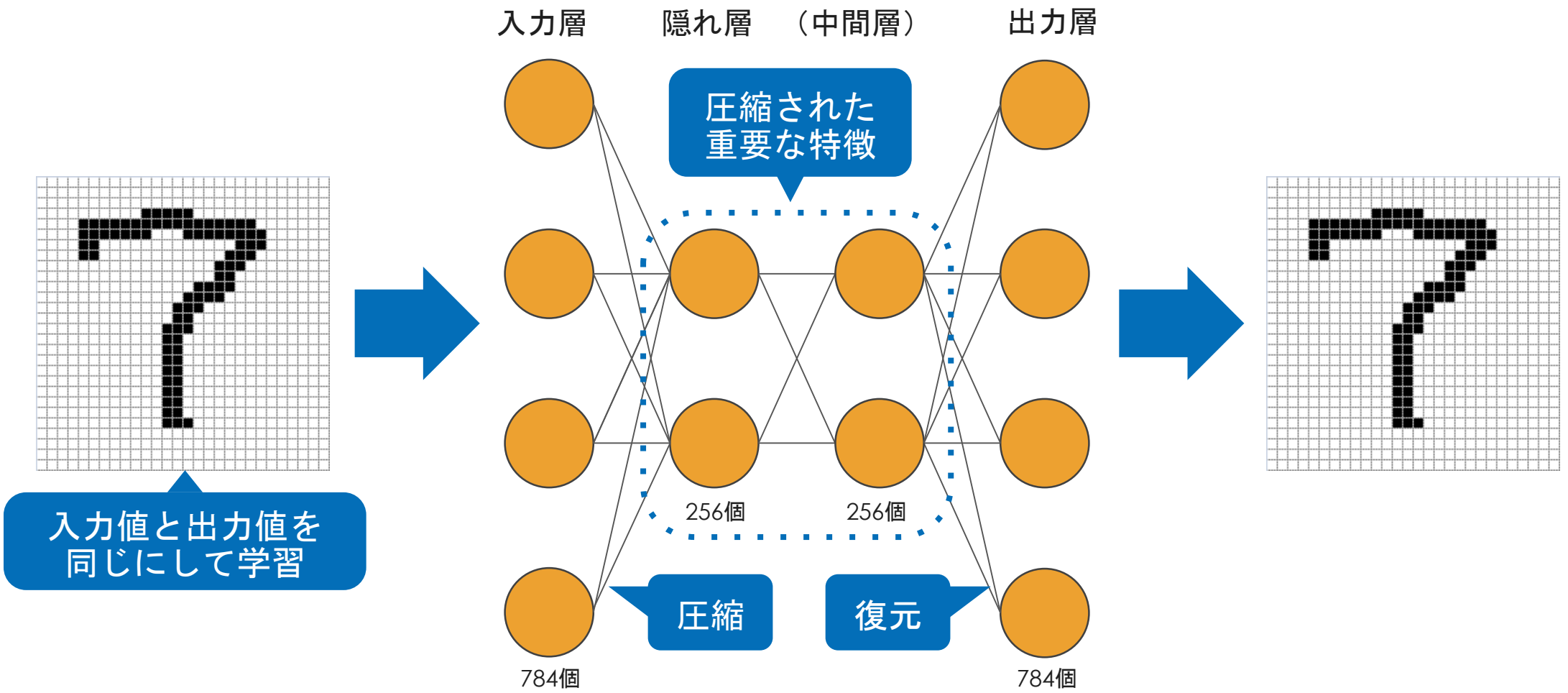
例：オートエンコーダのノイズ注入

- オートエンコーダを用いた場合
- オートエンコーダは自己符号化とも言う
- 通常のオートエンコーダ以外に、デノイジング・オートエンコーダと呼ばれる手法が存在する

オートエンコーダの仕組み①（学習）

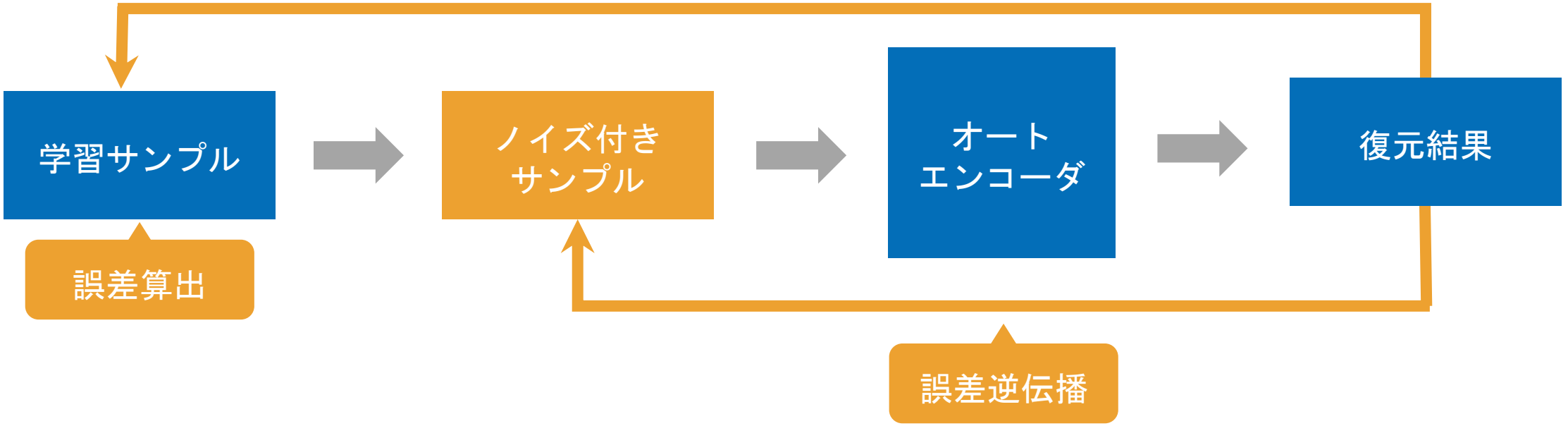


オートエンコーダの仕組み②（テスト）



デノイジング・オートエンコーダ

- 元画像からノイズを除去できるオートエンコーダ
- ネットワーク構成は変わらない
- 学習サンプルにランダムノイズを与え入力とする



デノイジング・オートエンコーダの効果

- 少ない学習データでより良い学習成果を得られる
- 入力した情報を維持したまま、よりよい特徴を抽出できる
- 入力に含まれるノイズを除去できる

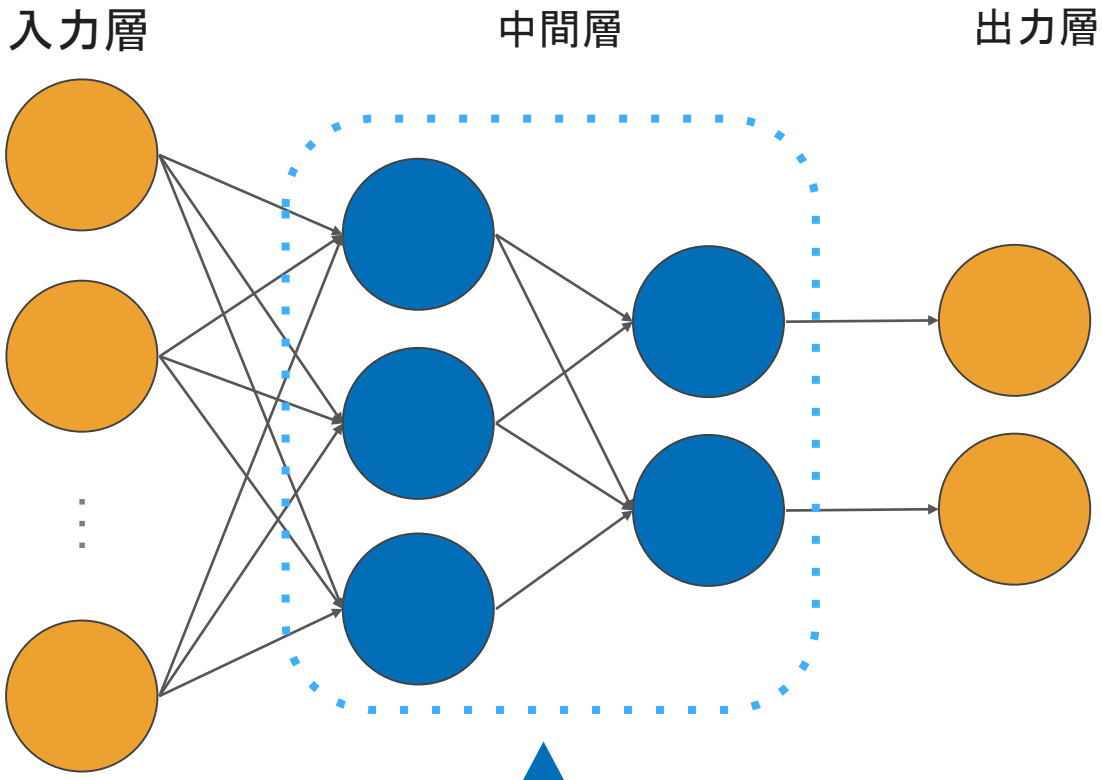
ディープラーニング体系講座 DAY 2

ノイズに対する頑健性

入力値以外の値にノイズを入れる方法

- 中間層にノイズを入れる方法
  - ドロップアウト（後述）
  - ドロップコネクト
- 出力目標へのノイズの注入
  - ラベル平滑化

中間層へのノイズの付加



中間層の重みにノイズを入れることにより汎化能力を上げることが出来る

ドロップコネクト

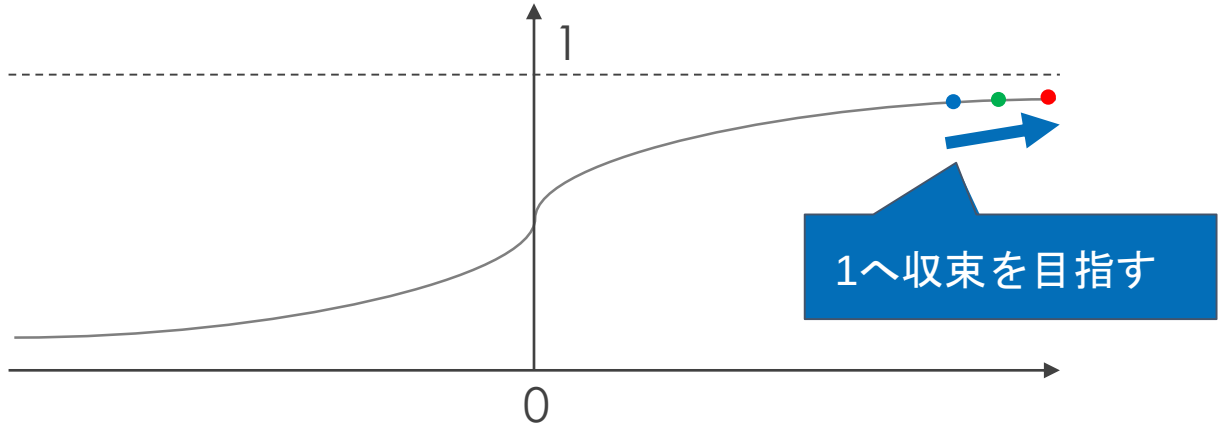
- 中間層の重みにノイズを加える方法の一つ
- 中間層の重みの一部をランダムに選んで0にする方法
- 性能面ではドロップアウトより優れている
- 乱数の値の与え方が異なると同じ性能を達成するのが困難

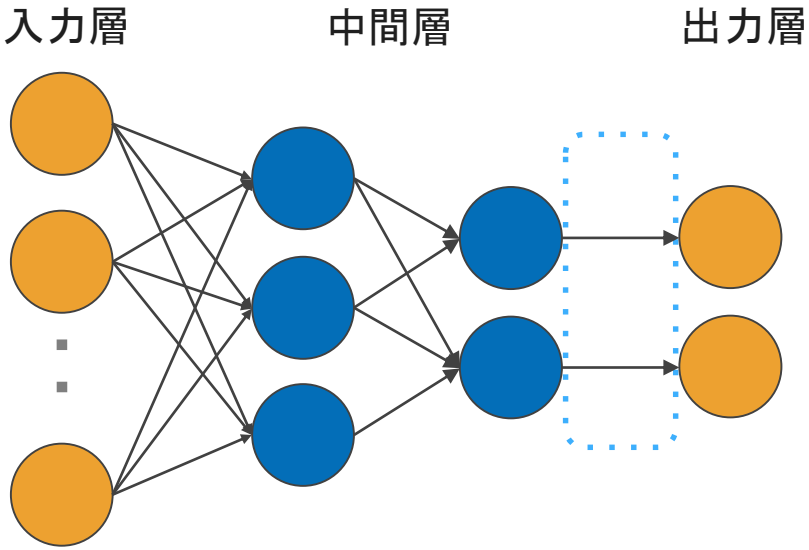
ドロップアウトのほうが用いられる

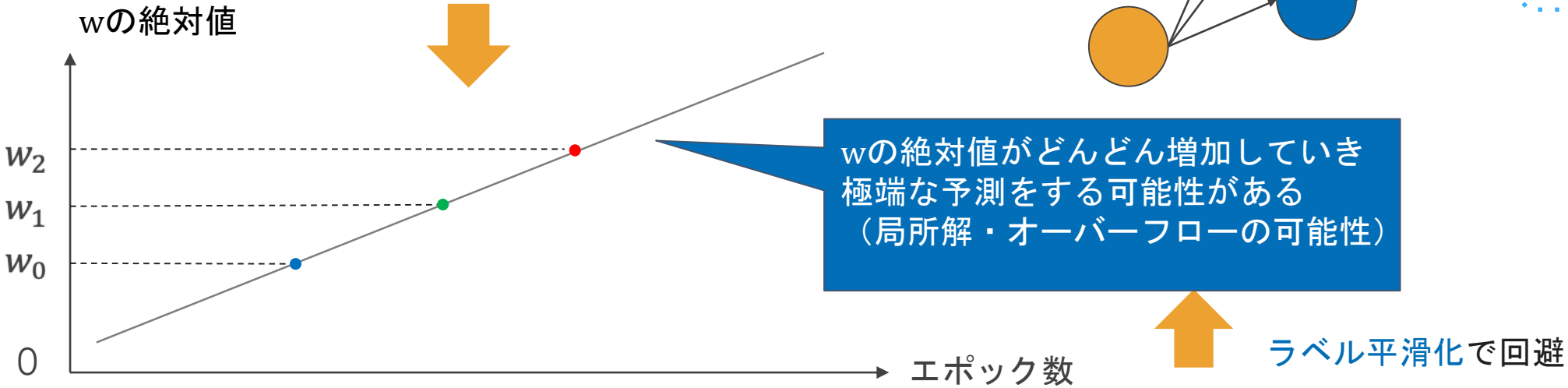
ラベル平滑化（label smoothing）

- 1980年代から知られている方法
- ソフトマックス関数の正則化に用いられる方法
- 目的関数の値を1、0ではなく、$\frac{\varepsilon}{k-1}$ 、1−εに設定する
  - k>1（出力の数）
  - ε>0（ごく小さな値）
- 目標が0、1だと重みがどんどん大きくなり極端な予想をする可能性がある
- この方法を用いると正しい分類と重みの増大を防ぐ両方の役割を果たすことが出来る

1と0を目標としたときの問題点







ディープラーニング体系講座 DAY 2

半教師あり学習

教師あり学習のメリットとデメリット

メリット

- 説明変数と目的変数の関係性を
  しっかり定義できる
- 問題を明確に定義できる

デメリット

- 大量のデータを作るのに時間と
  手間がかかる
- データを人為的に作る過程で
  間違いが起こる可能性が高い
- 人間のメンテナンスが必要で
  手間がかかる

教師なし学習のメリットとデメリット

メリット

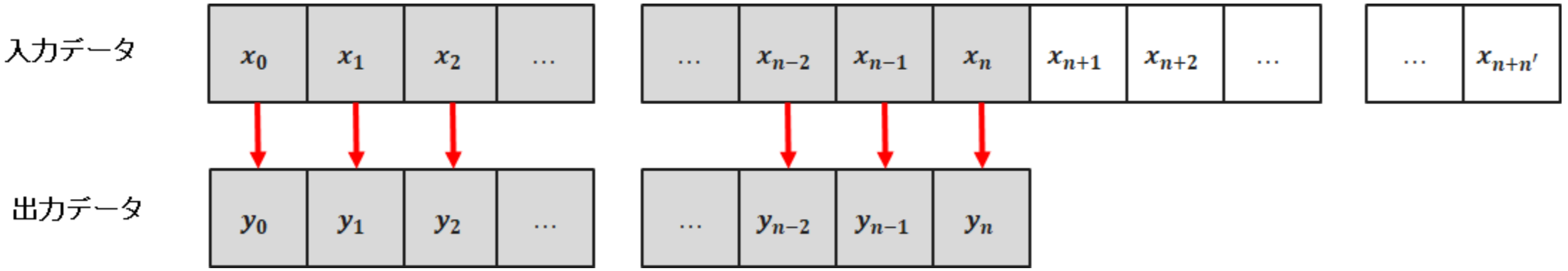
- 機械的に学習を行わせることができるため作業が楽
- メンテナンスの手間もあまりかからない

デメリット

- 問題を定義することが難しい
- 結果から意味を読み取ることが難しい

半教師あり学習とは

- 教師あり学習と教師なし学習の「良いとこどり」
- 最初の段階では教師あり学習
  - $\{(x_i, y_i)\}_{i=0}^{n}$ ・・・入出力が対になった訓練標本の組み合わせ
- 一度最初の学習が終わるとそのあとは教師なし学習に移行
  - $\{x_i\}_{i=n+1}^{n+n'}$ ・・・入力だけの訓練標本



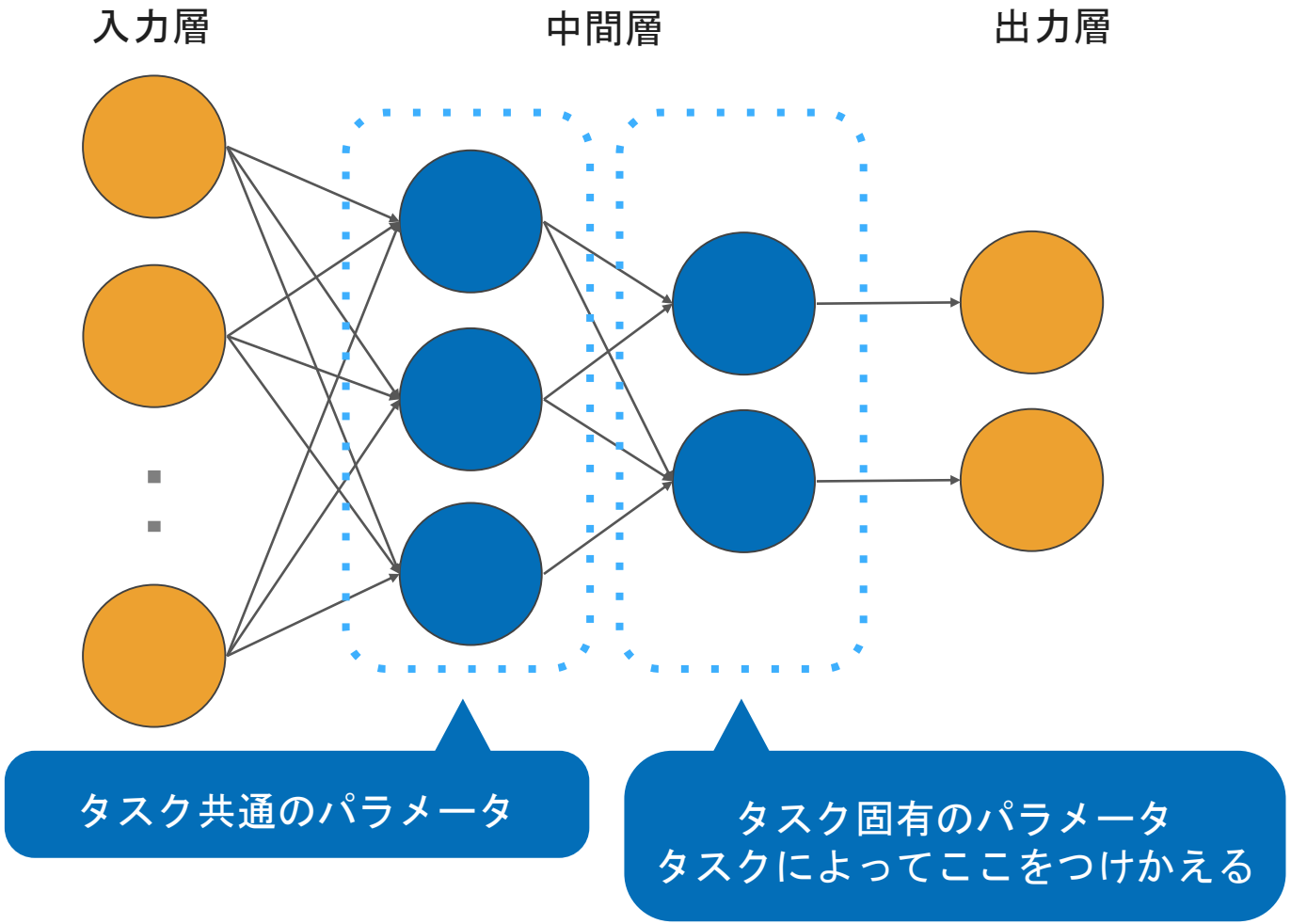
ディープラーニング体系講座 DAY 2

マルチタスク学習

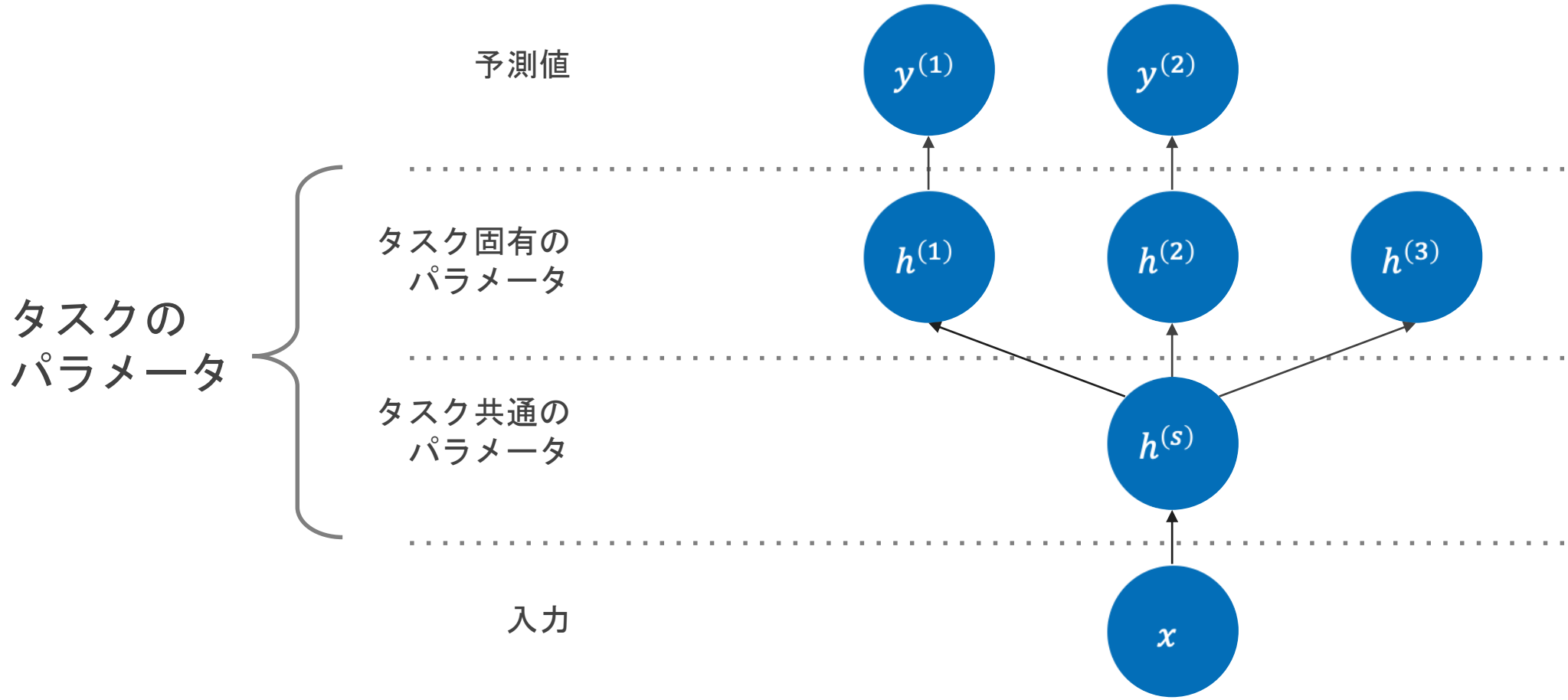
マルチタスク学習とは

- 複数の学習タスクを解く方法の一つ
- 複数のタスクの類似性を活かしながら全タスクを同時に解く
- モデルの一部がタスク間で共有される
- いくつかのタスクから生じる事例を貯めることで
  汎化性能を向上させられる

マルチタスク学習とは



マルチタスク学習のイメージ

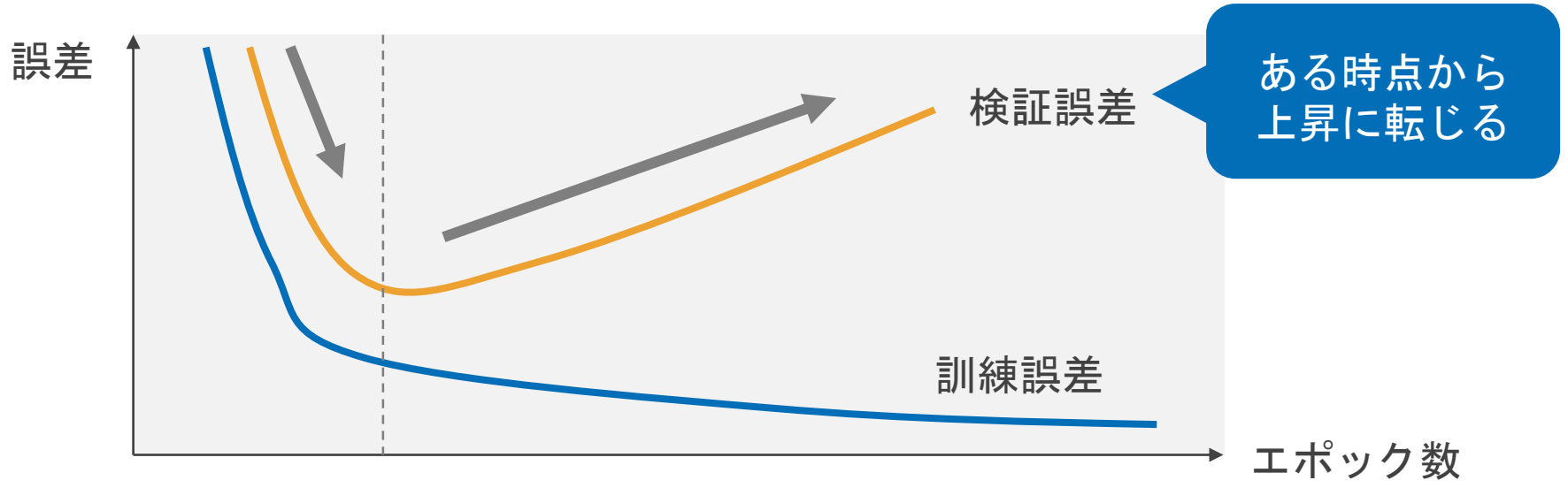


ディープラーニング体系講座 DAY 2

早期終了

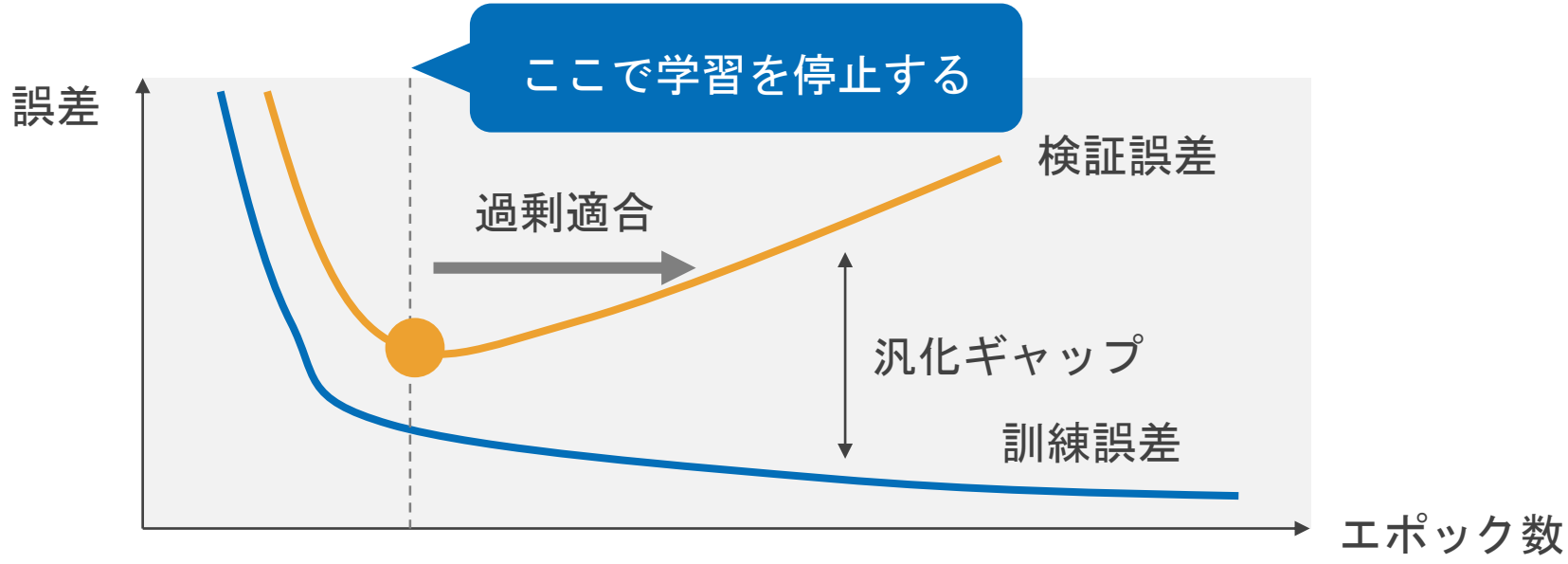
エポック数と誤差の関係

- 大きなモデルを訓練した場合以下のようなケースが発生する
- 訓練誤差 — 時間とともに減少
- 検証誤差 — 再び増加し始める
- この状態は過学習であると考えられる



早期終了（early stopping）

- 正則化の形態の一種
- 過去のエポックで更新した重みパラメータを記憶
- 学習を続け訓練誤差が増加し始める点で学習を停止

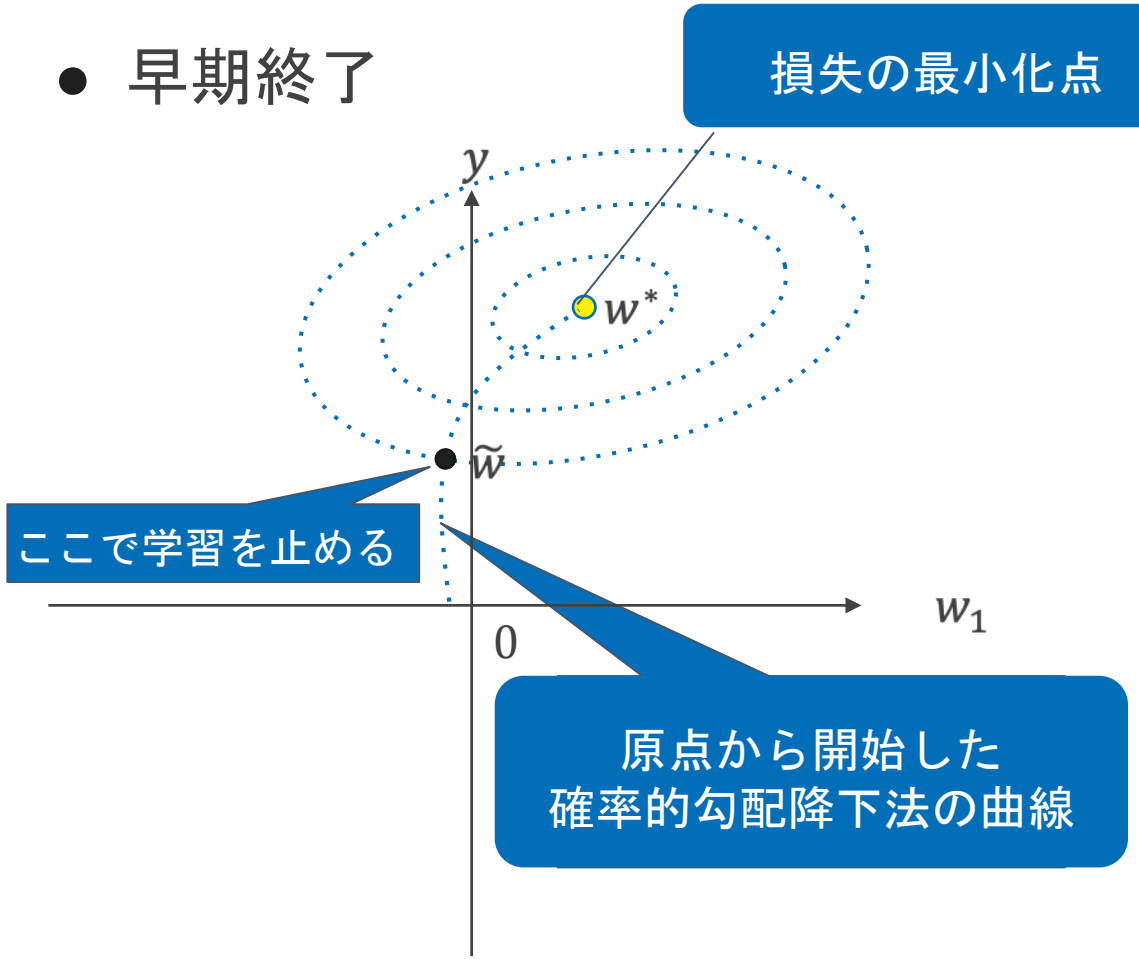


早期終了のアルゴリズム

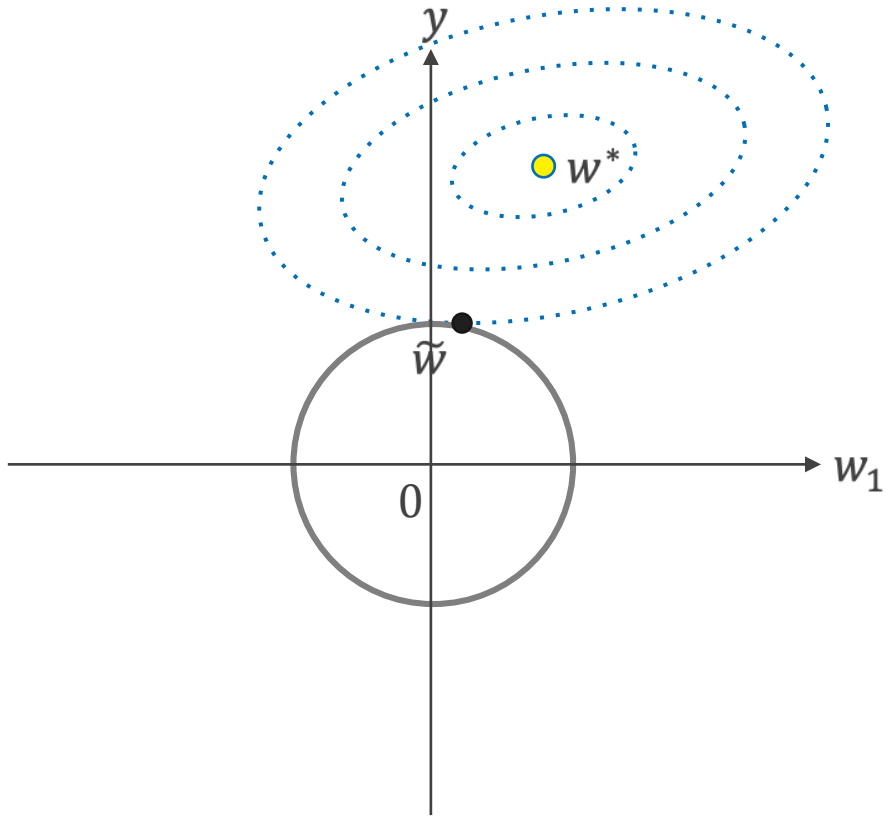
1. 周期と期間の長さを決定
   - 周期 —— 検証誤差を評価するパラメータ更新の頻度
   - 期間の長さ —— どの位のステップ数で検証誤差が連続して増加すれば学習を停止するかを決める長さ
2. 学習が始まり検証誤差の増加が期間の長さに達したら学習を停止
3. 減少から増加へ転じた時点のハイパーパラメータを最適値として採用

正則化項としての早期終了

- 早期終了



- L2ノルムによる正則化



ディープラーニング体系講座 DAY 2

パラメータ拘束とパラメータ共有

パラメータ拘束の前提

- L2正則化（または重み減衰）――0から遠ざかることに対してペナルティを与えることが出来る。
- しかしパラメータの適切な値がわからないようなケースも存在
- そのような場合、領域とモデル構造に関する知識から、近いモデルを参考にすると良い。
- そのために、互いのパラメータの近さを表現する手法が必要となる

パラメータ拘束

- 2つのタスクが非常に類似している場合、２つのモデルのパラメータが近いと想定できる
  - $\forall i$に対し、$\boldsymbol{Wi}\,(\boldsymbol{A}) \approx \boldsymbol{Wi}\,(\boldsymbol{B})$
- このとき、この情報は正則化として活用可能
- $\Omega\left(w^{(A)}, w^{(B)}\right) = \left\|w^{(A)} - w^{(B)}\right\|_2^2$ をパラメータのノルムペナルティとすることができる。（ノルムはL2以外でもよい）
- この値に上限を設けることによりパラメータに互いに制限をかけることが出来る（パラメータ拘束）

パラメータ共有

- パラメータ共有はパラメータ拘束をさらに進めたもの
- パラメータのある部分を同じにする
- 複数のモデルが同じパラメータ持つのでメモリの節約になる
- 典型的な例が、CNNの畳み込み処理でのフィルタの処理

ディープラーニング体系講座 DAY 2

スパース表現

スパース表現

- 高次元の行列・ベクトルの成分の多くを0にすること
- 重みや入力などを適度にスパースにすることにより学習が効率的に進む

- 重みをスパースにする→L1正則化
- 表現（層間で受け渡されるベクトル）をスパースにする→ReLUなど

- 表現に対してノルムペナルティを課すこともある

ディープラーニング体系講座 DAY 2

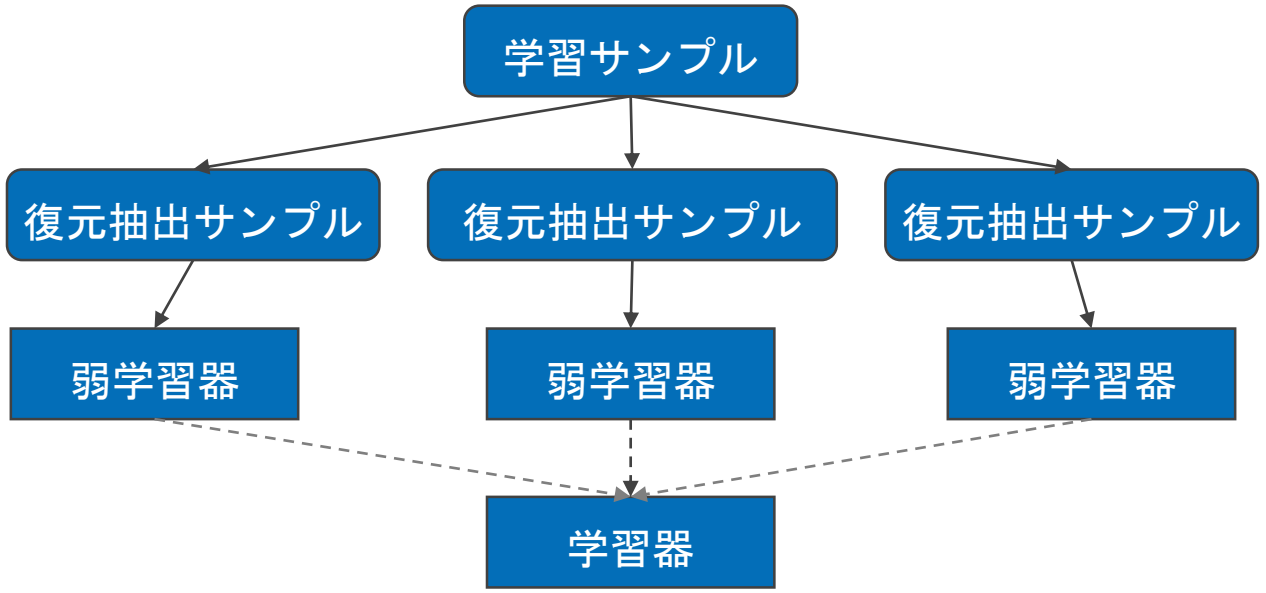
バギングや
その他のアンサンブル手法

モデル平均化とアンサンブル学習

- 複数のモデル（弱学習器）個々に学習させる
- モデルの全てからテスト事例に対する出力を投票させる
- この手法のことをモデル平均化（model averaging）と言う
- このような学習方法をアンサンブル学習という。
  - バギング
  - ブースティング

バギング（bagging）

- ブートストラップ集約（bootstrap aggregation）の略
- 複数のモデル（弱学習器）を組み合わせ汎化誤差を減少させる
- 機械学習の代表的手法としてランダムフォレストがある



バギングの仕組み

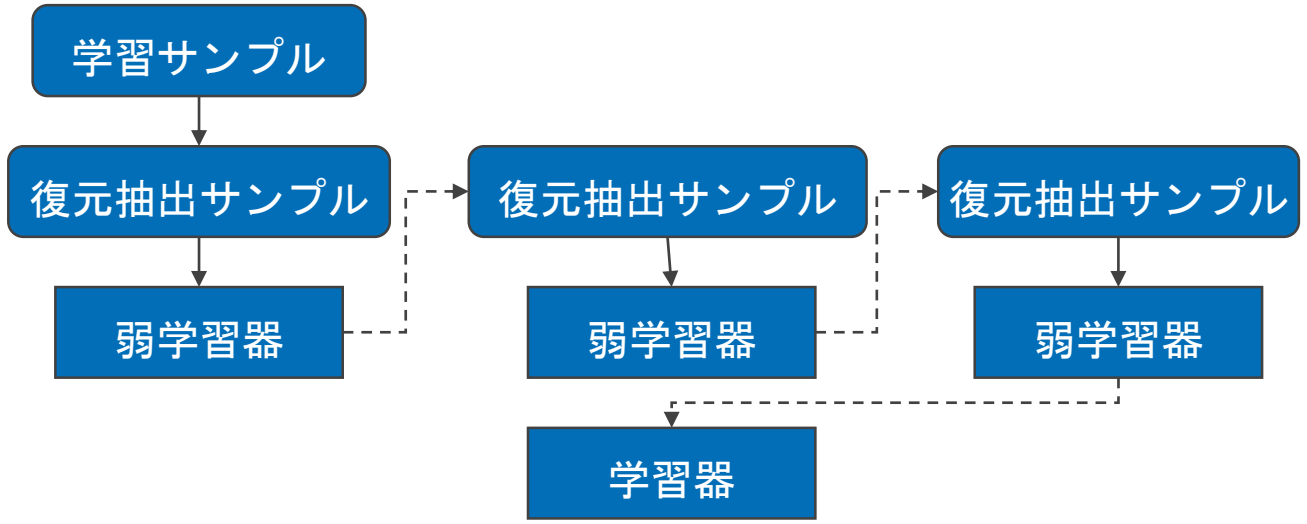
- 以下の手順をB回繰り返す
  - 学習データから、m回分割抽出をして、新しいデータセットを作る
  - その分割されたデータセットを元に、弱学習器h^iを構築
- B個の弱学習器h^iを用いて、最終的な学習結果Hを構築
  - 分類問題の場合・・・ $H(x) = arg\,max|\{i|h_i = y\}|$
  - 回帰問題の場合・・・ $H(x) = \frac{1}{B}\sum_{i=1}^{B} h_i$

ブースティング（Boosting）

- 学習データの一部を用い最後に合併させるのはバギングと共通
- 以前に使用したデータを再利用して文字通りブーストする
- 機械学習の代表的手法として**AdaBoost**がある。

ブースティングの仕組み

- サンプルに対して、確率分布に基づいて、T個に分割した弱学習器を一列に並べる
- 弱学習器の誤り率Eと重要度αを逐次計算していく
- 毎回全体のウエイトを調整



ニューラルネットワークのモデル平均化

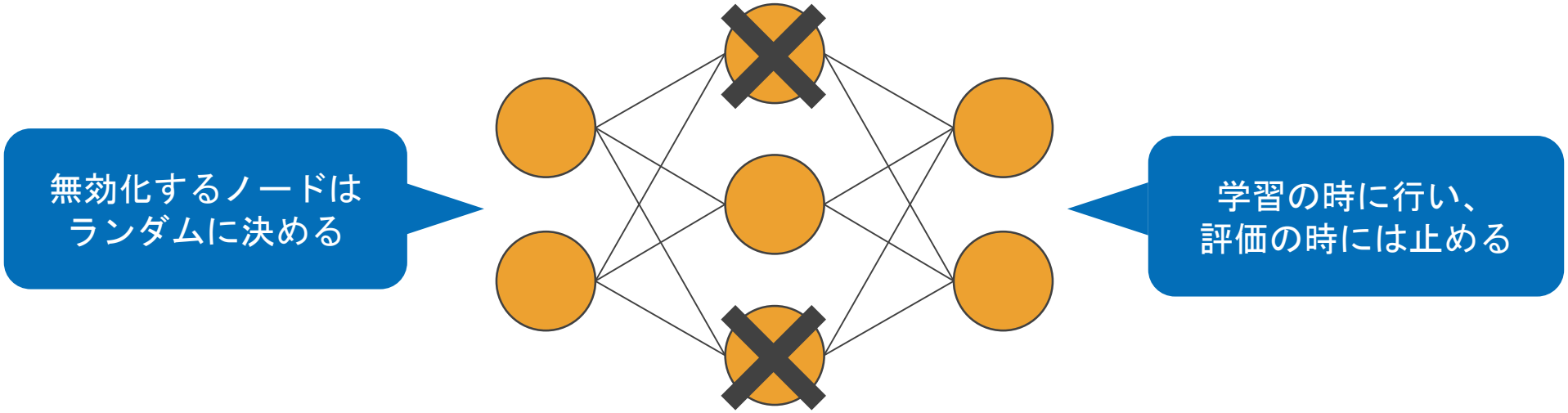
- 全てのモデルが同じデータで訓練されていても十分に幅広い多様な解の点に到達する
- アンサンブルを構成する個々のモデルに部分的に独立な誤差を生成できる
  - ランダムな初期化による差
  - ミニバッチのランダムな選択による差
  - ハイパーパラメータの違いによる差
  - ニューラルネットワークの非決定的論的な実装結果による差

ディープラーニング体系講座 DAY 2

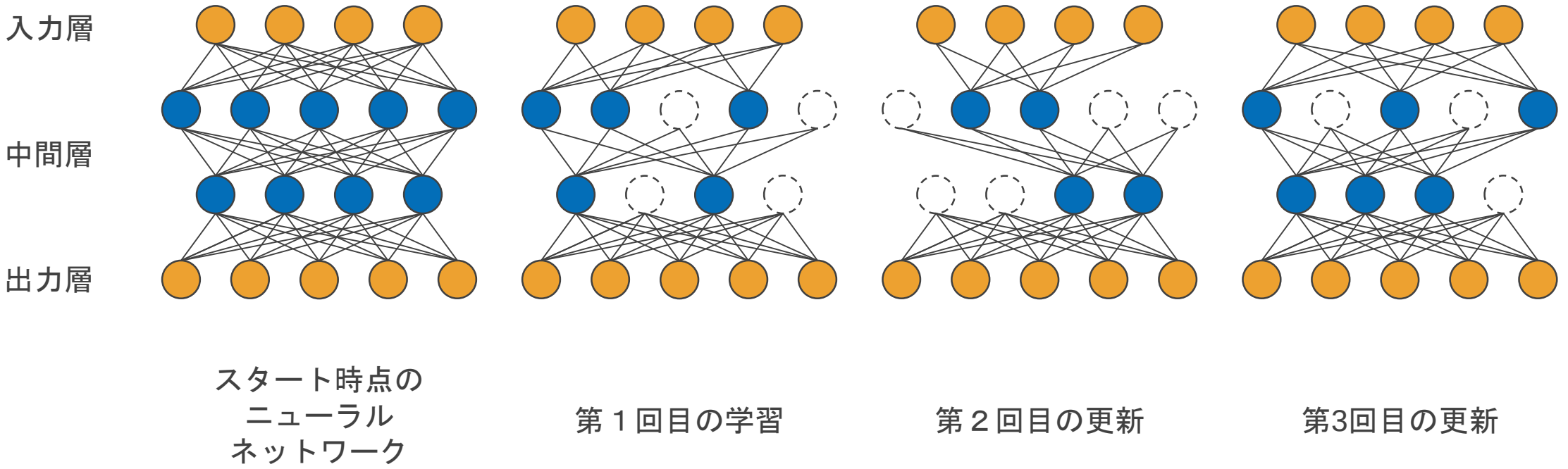
ドロップアウト

ドロップアウト

- ニューラルネットの正則化の手法の一つ
- CNNの全結合層等に施す処理
- ノードのうちのいくつかを無効にして学習を行う
- 一般的に50%前後を指定すると有効とされる

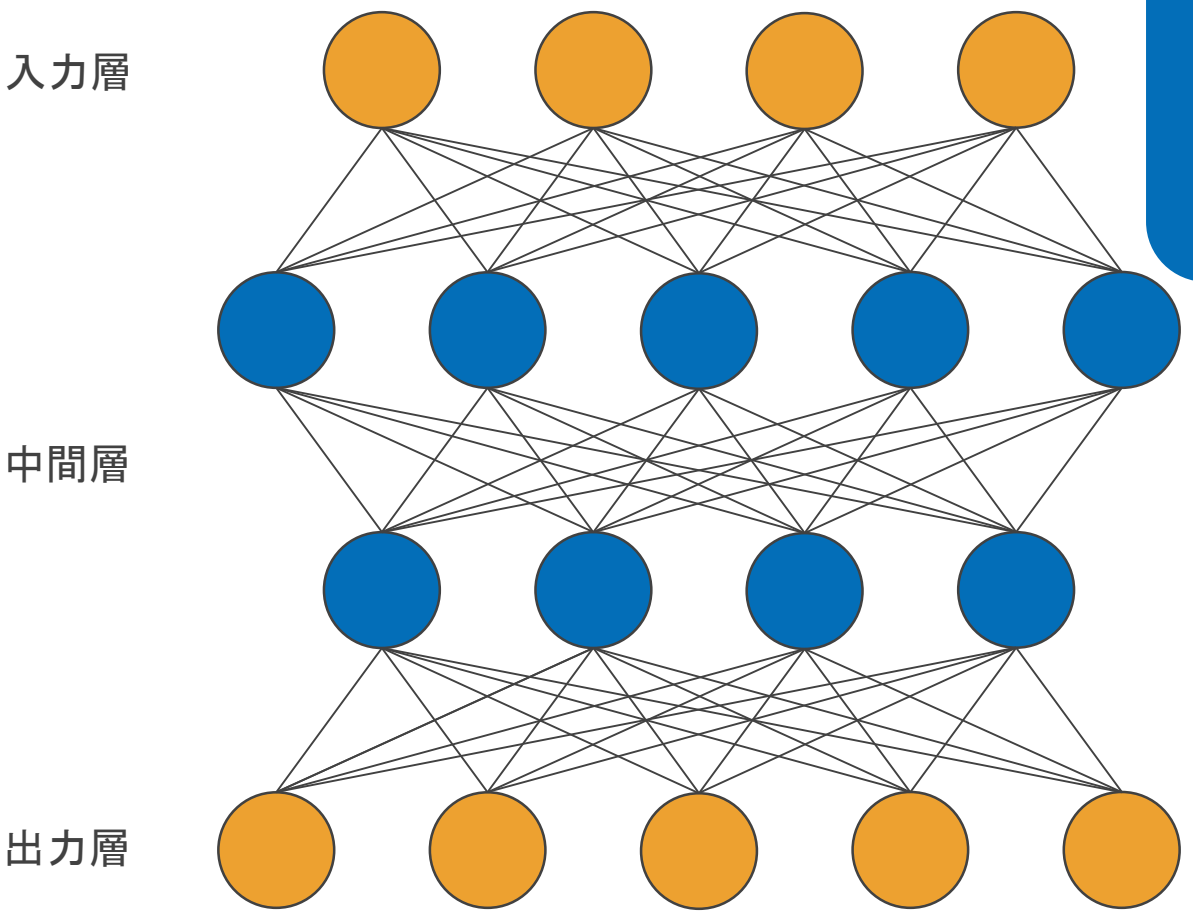


ドロップアウトの学習処理のイメージ



部分ネットワークのアンサンブル学習として考えることができる

ドロップアウトとネットワークの評価



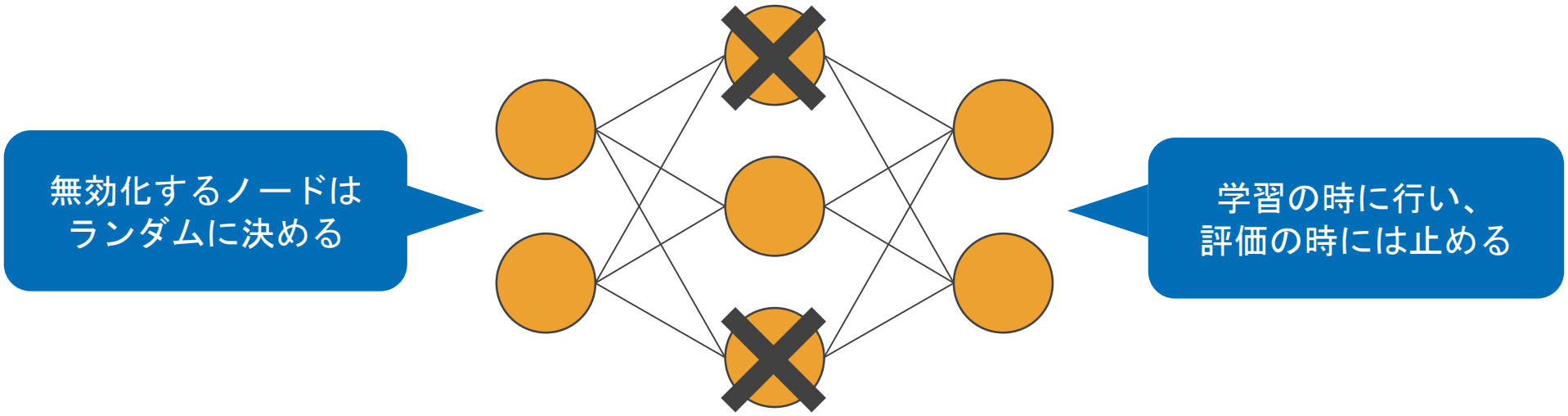
出力は
ドロップアウトの割合×
重みづけ結合値

（ライブラリによっては、訓練時に
ドロップアウト率の逆数で補正する
ものもある）

評価には
全てのネットワーク使う

ドロップアウト

- 非常に実用性が高い
- 計算量が少ない
- ほぼ全てのモデルに対して適用することが容易
- 訓練データが少なすぎるときはうまくいかない傾向にある



ディープラーニング体系講座 DAY 2

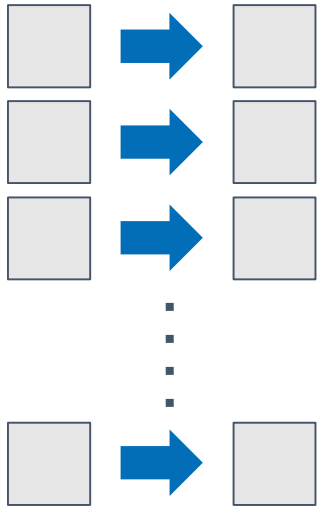
アクセラレータ

GPU

- **GraphicsProcessingUnit**と呼ばれ、元々は**画像処理を高速に行う装置**として発展
- ベクトル型プロセッサという機構をもち、単純な計算を**並列処理で高速化**できる
- 近年、**深層学習の演算資源として活用**されるようになる
- このように、GPUを画像処理以外の目的に利用する技術を**GPGPU**と呼ぶ

CPU

複雑な命令を
逐次処理（直列）で実現



GPU

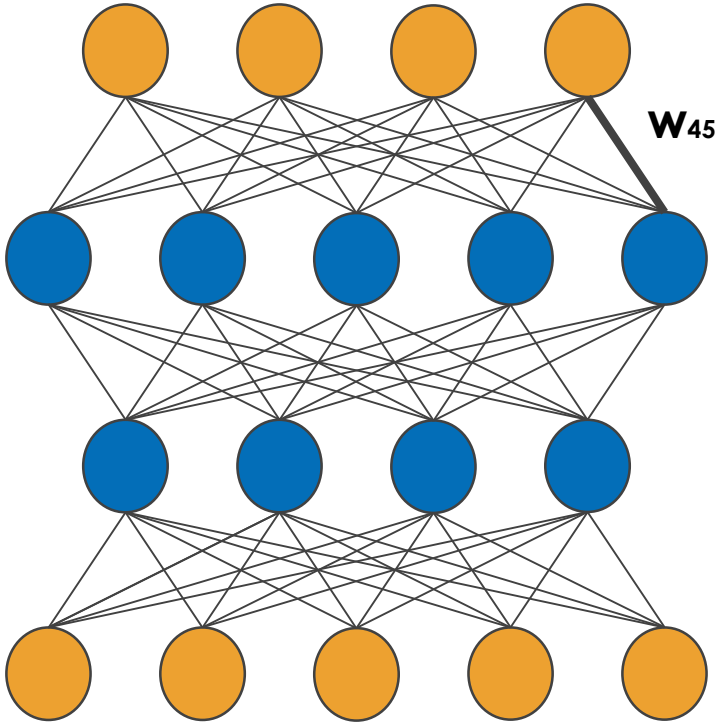
単純な命令を
並列処理によって高速化

ディープラーニング体系講座 DAY 2

軽量化技術

量子化

- 浮動小数点で表されるネットワークの**パラメータ等を低bitで表現**（近似）
- パラメータだけを量子化すれば、ほぼ精度を落とさずにモデル圧縮が可能



w_{45} = 0.12492940522023…（10進数）

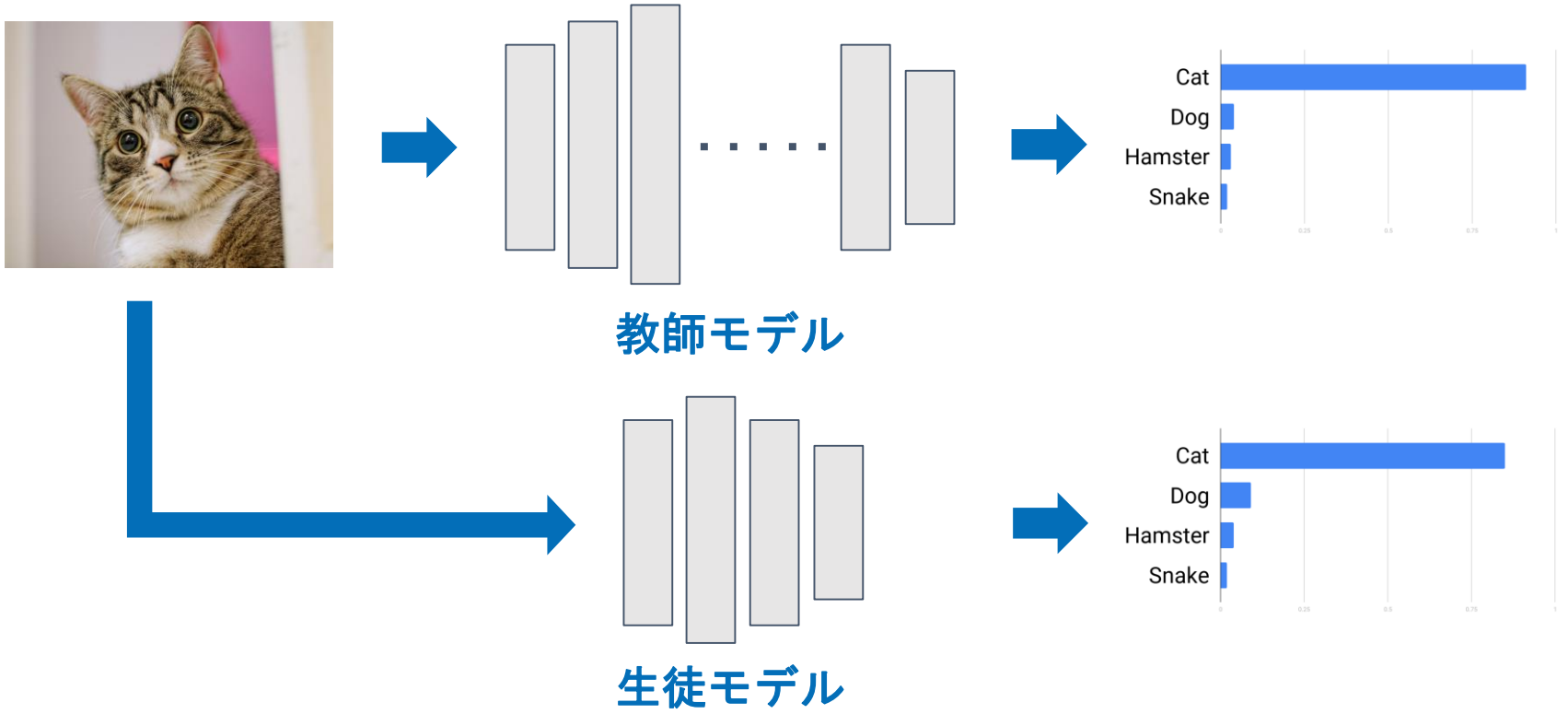
0000 0000 . 0001 1111 1111 1011 0101 1111
（2進数・32bit）

0000 0000 . 0001 1111（2進数・16bit）

w_{45} = 0.12109375（10進数）

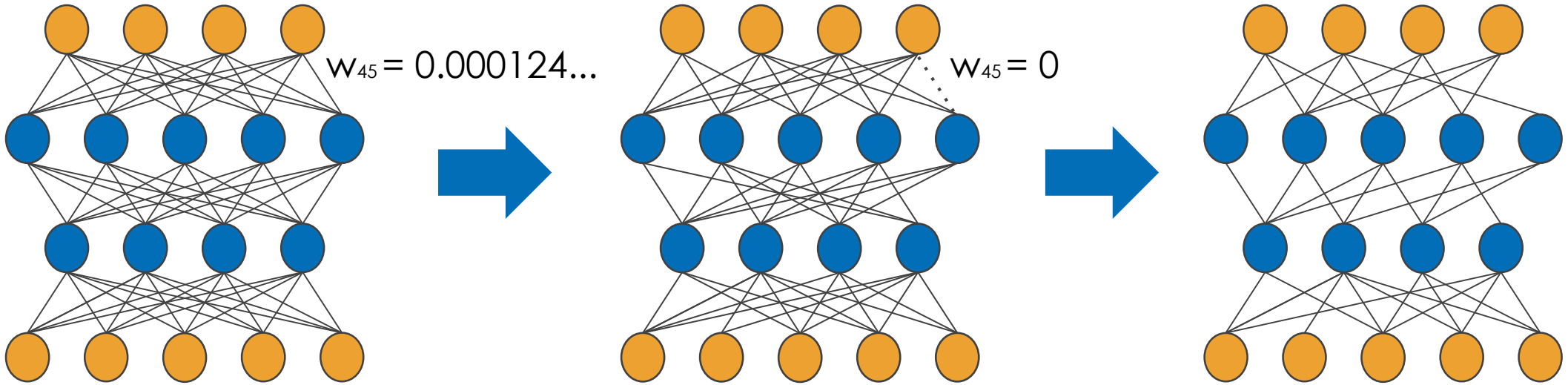
蒸留

- 訓練された**教師モデルの入出力を用いて**、より軽量なモデル（生徒モデル）で学習
- 精度が落ちる反面、軽量なモデルで計算資源を大幅に削減できる



プルーニング

- Pruning（枝刈り）の名の通り、**寄与の小さい重みを0にする（データの流れを切る）**
- 学習を繰り返すたびに枝刈りを実行し、精度を保ったままモデルを軽量化させる



ディープラーニング体系講座 DAY 2

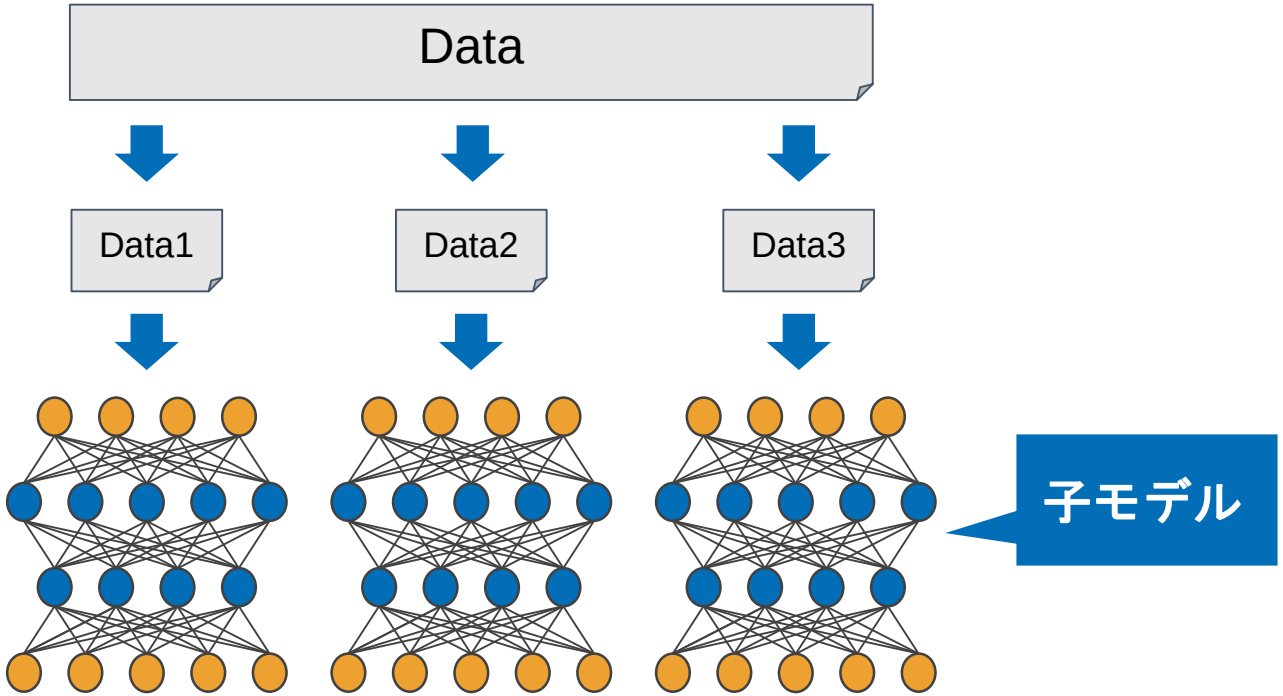
分散処理

モデル並列とデータ並列

データ並列

データを分割してそれぞれ勾配計算

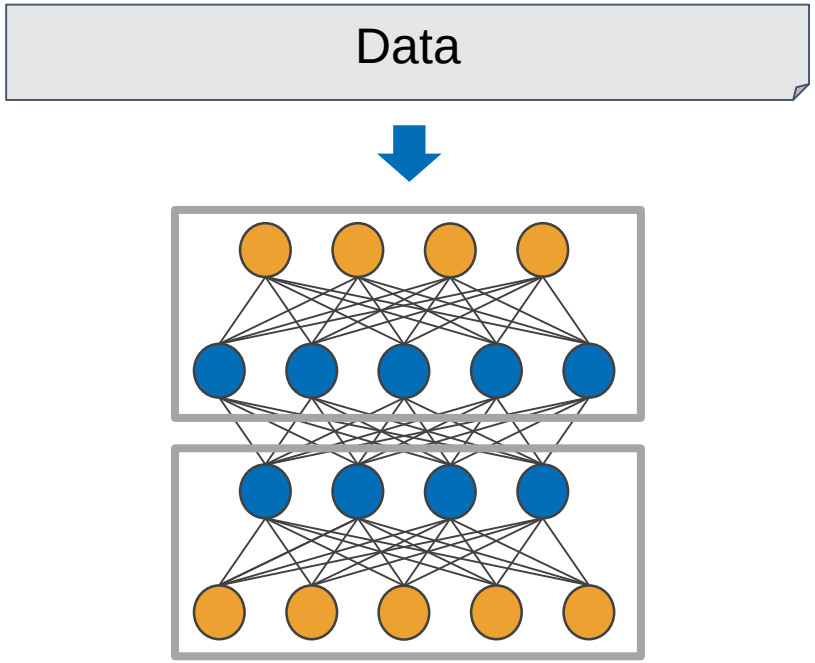
同期型：全GPUの計算が完了してからパラメータ更新
非同期型：1つ以上の計算が完了したらその時点で更新



モデル並列

モデルを分割して勾配計算

モデルによって最適な分割の仕方は異なる



アンケートのお願い/SNSなど

今日の講座をより良くするため、以下のURLからアンケートの協力をお願いしております。

https://seminar.to-kei.net/qt/

LINE@ イベントや講座の優先告知、事務的な質問等をすぐに返します。
https://avilen.co.jp/contact/

twitter 世の中のAIニュースをビジネス視点で紹介するアカウント
https://twitter.com/toukei_net

Thank you