深層学習 Chapter 5

自己符号化器

2015.07.01 @at_grandpa

@at_grandpa

・カメラ / 写真
・将棋
・テニス / ランニング
・機械学習歴 一ヶ月程度

注意

- ✓自己符号化器を1mmも知らなかった人のまとめです
- ✓直感的な解釈が多いです
- ✓実装は間に合わなかったので後日やります（ホントか？…）
- ✓飛ばしているところもあります
- ✓本の流れから脱する場合があります
- ✓間違っていたらご指摘ください

注意

- ✓自己符号化器を1mmも知らなかった人のまとめです
- ✓直感的な解釈が多いです
- ✓実装は間に合わなかったので後日やります（ホントか？...）
- ✓飛ばしているところもあります
- ✓本の流れから脱する場合があります
- ✓間違っていたらご指摘ください

「本のどの部分を言っているんだ？」と疑問に思ったら、
右下のページ数を参考にしてください

機械学習プロフェッショナルシリーズ

深層学習

Deep Learning

p55 Chapter 5

自己符号化器

の内容です

概要

自己符号化器とはなんだろう

自己符号化器とは

自己符号化器とは

以下の２つのことが行えるものです

自己符号化器とは

以下の２つのことが行えるものです

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

自己符号化器とは

以下の２つのことが行えるものです

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

② ディープネットの事前学習

・上手く学習できるような初期値を得る

自己符号化器とは

以下の２つのことが行えるものです

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

② ディープネットの事前学習

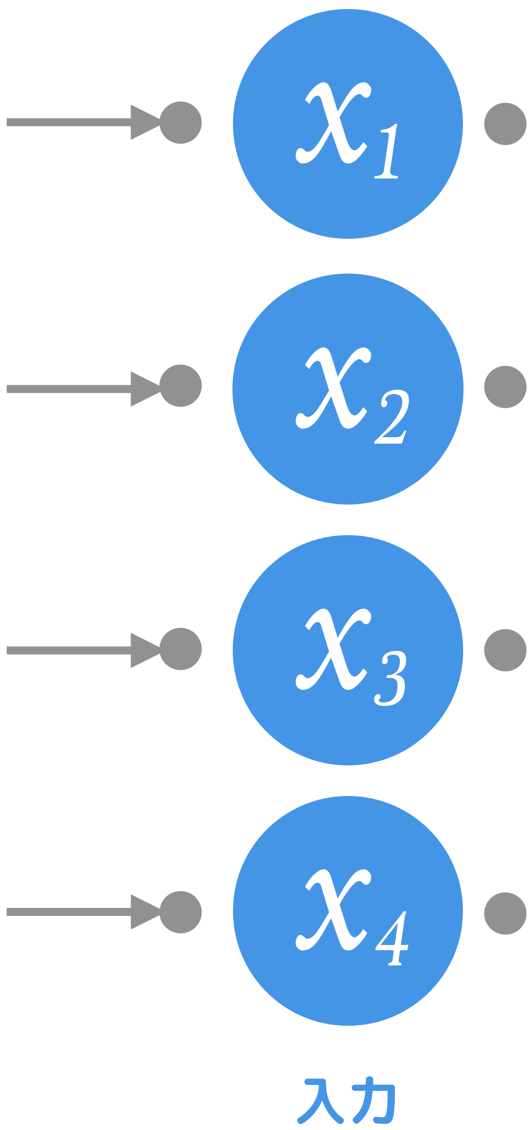
・上手く学習できるような初期値を得る

他にも使い道はありそうですが…

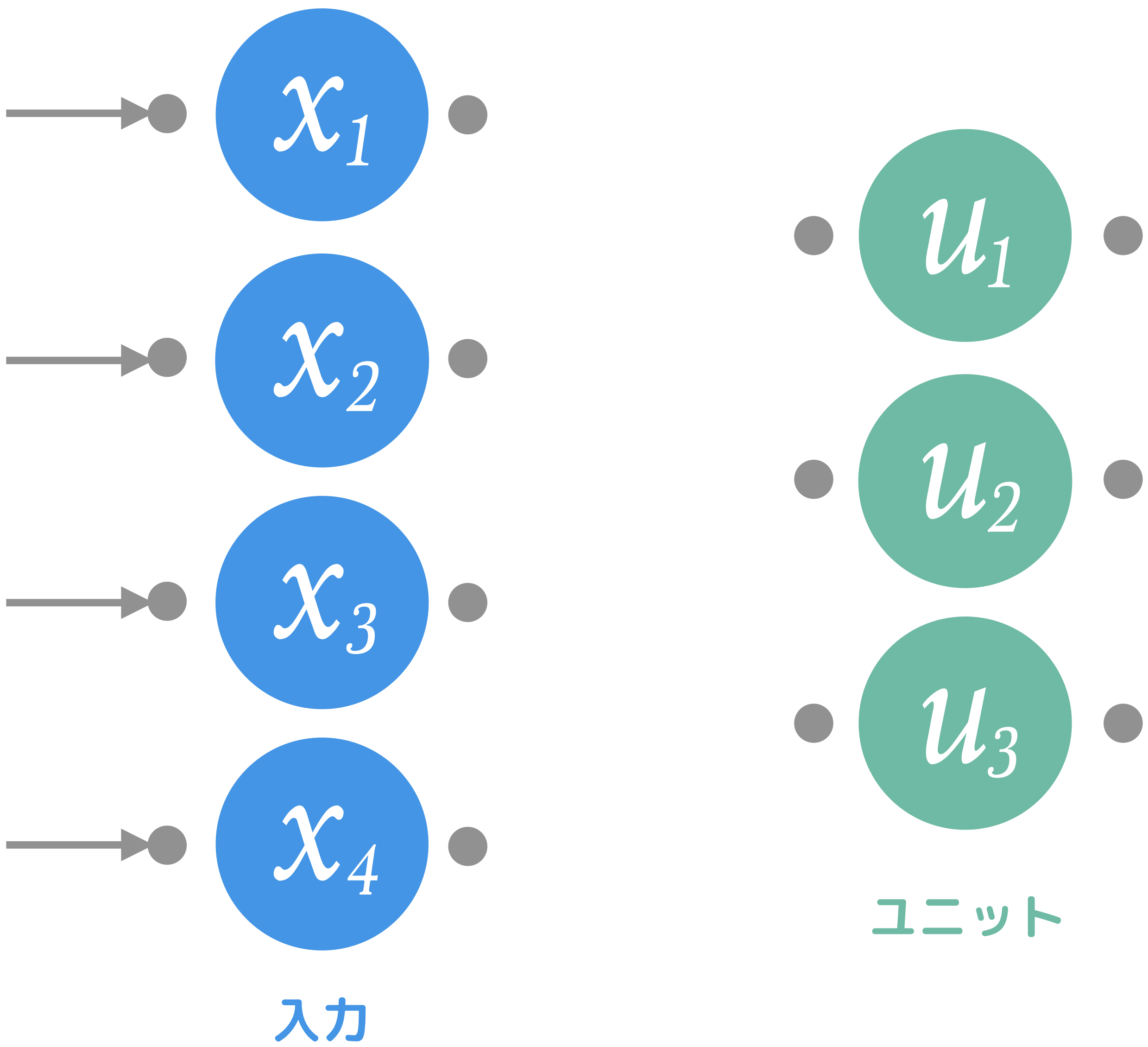
今までのニューラルネットとの違い

今まで見てきたニューラルネット

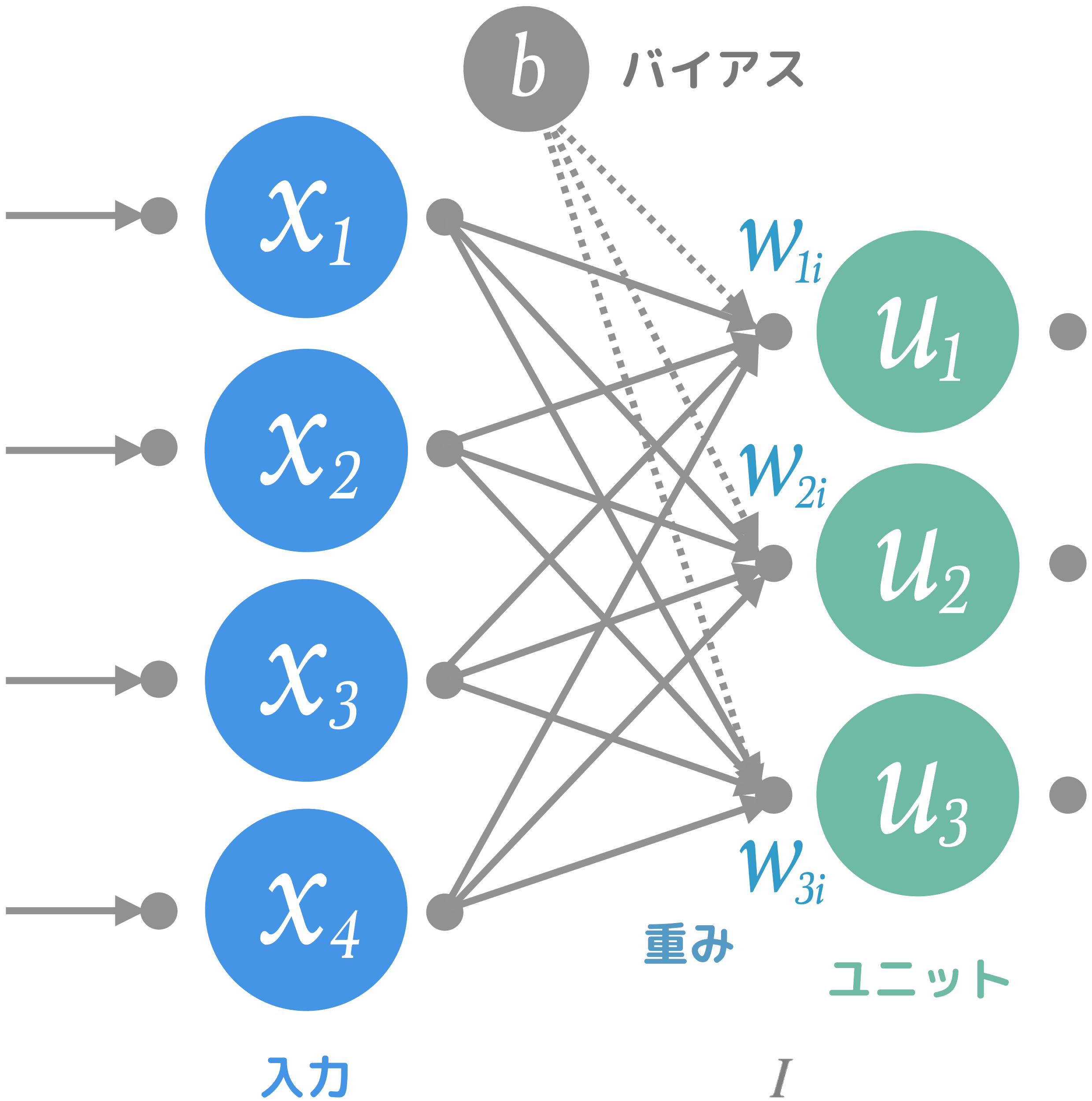
今まで見てきたニューラルネット



今まで見てきたニューラルネット

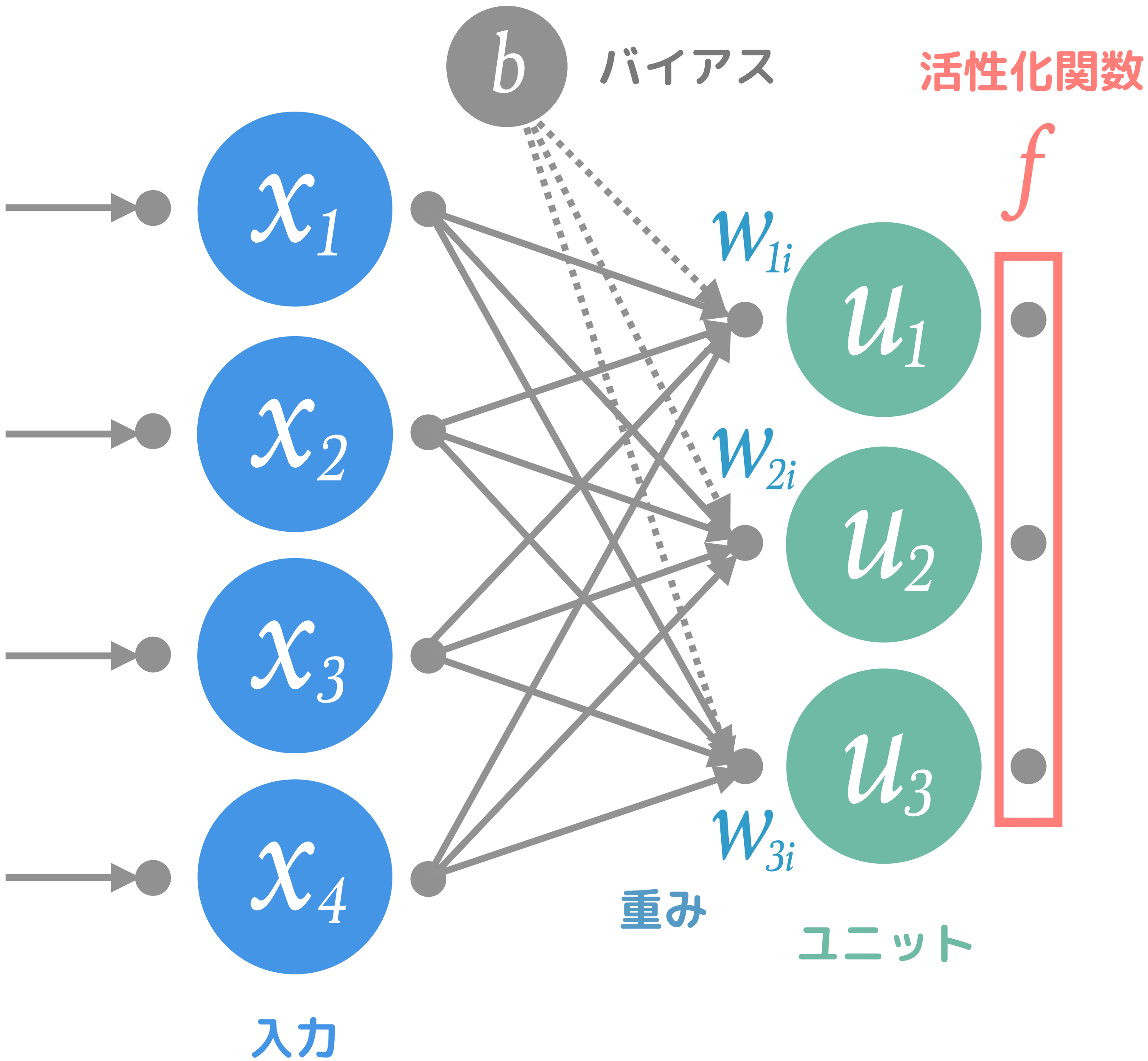


今まで見てきたニューラルネット

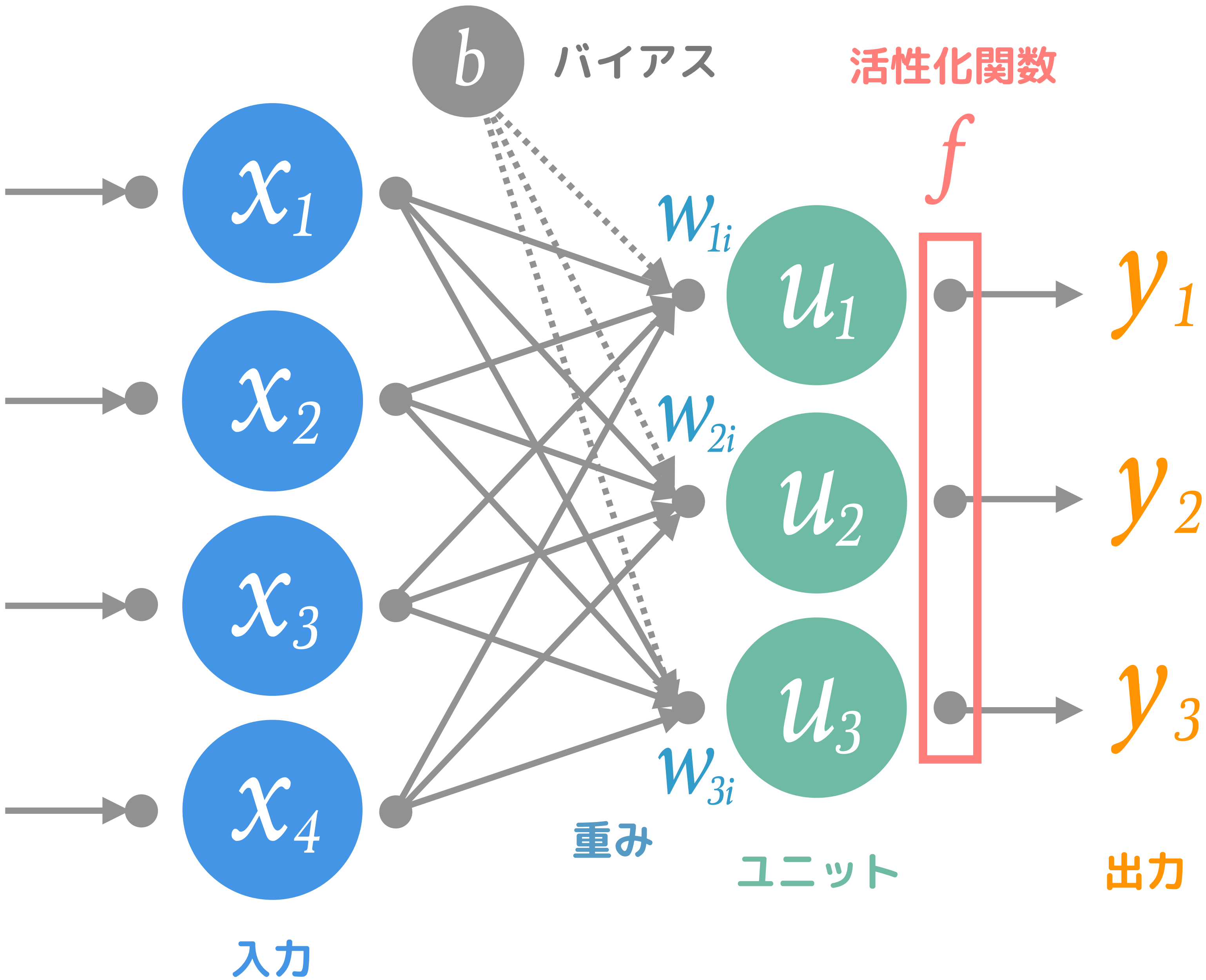


$$u_j = \sum_{i=1}^{I} W_{ji} x_i + b_j$$

今まで見てきたニューラルネット



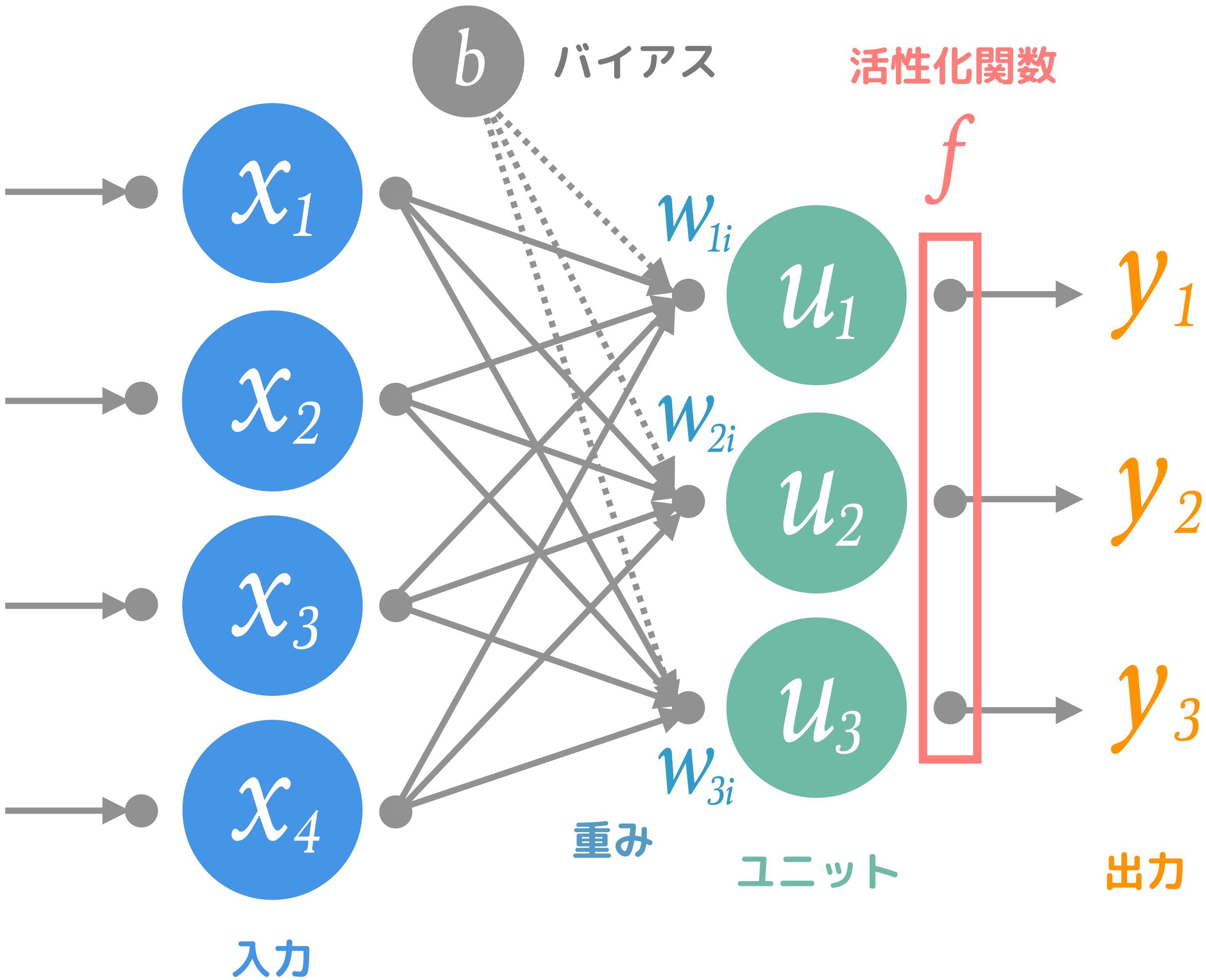
今まで見てきたニューラルネット



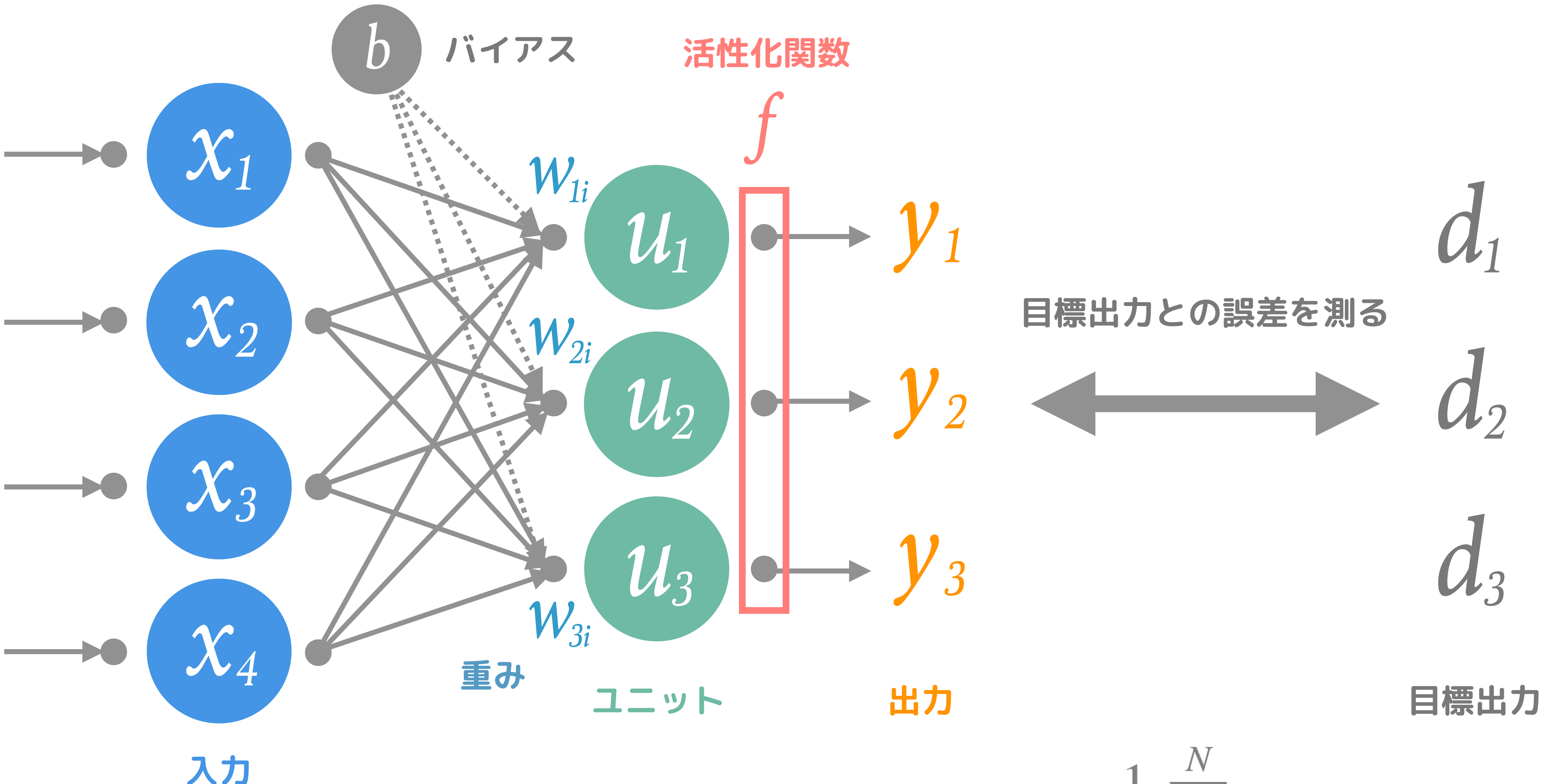
$$y_j = f(u_j)$$

本のページ p56

今まで見てきたニューラルネット



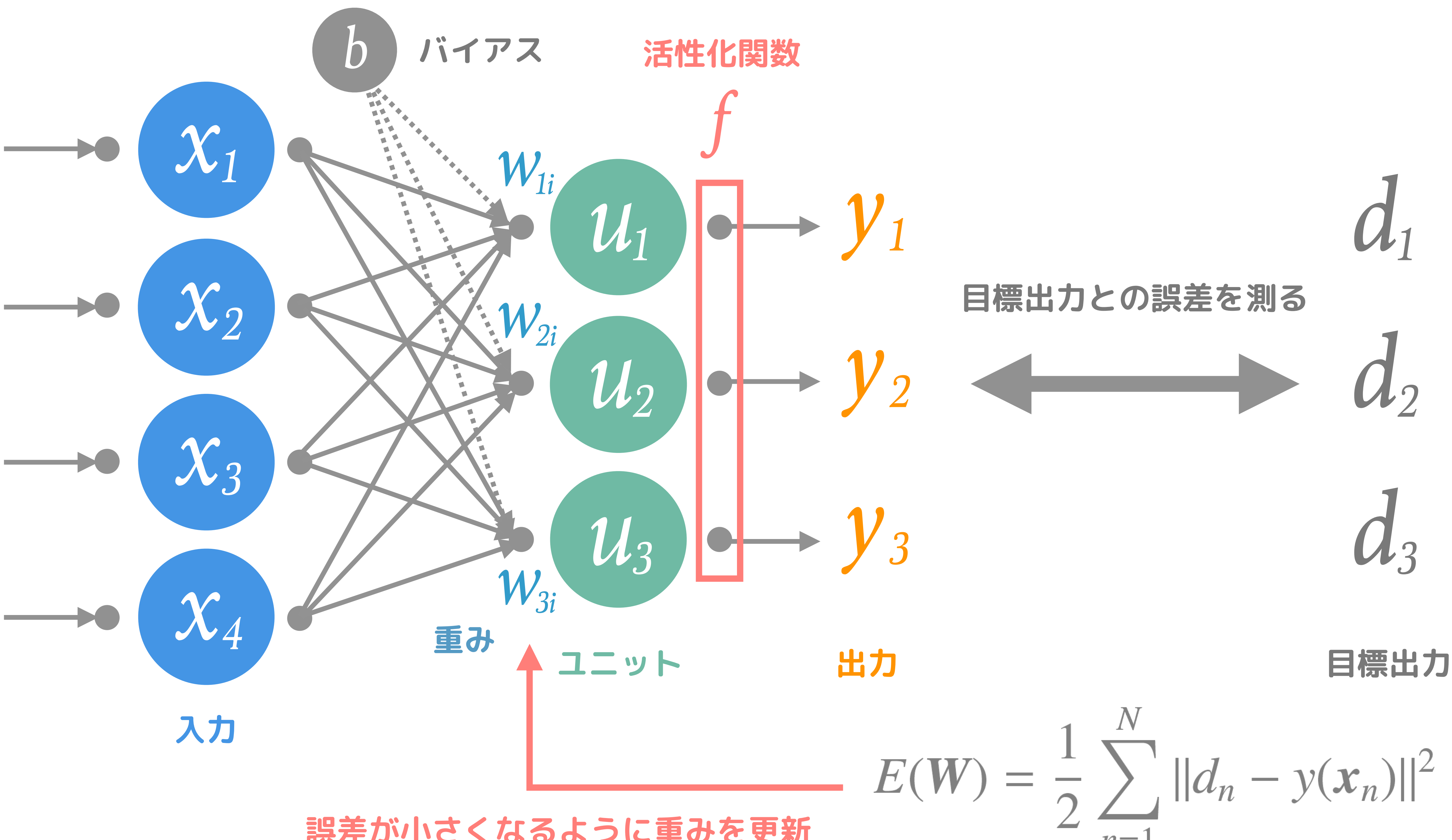
今まで見てきたニューラルネット



$$E(\boldsymbol{W}) = \frac{1}{2}\sum_{n=1}^{N} \|d_n - y(\boldsymbol{x}_n)\|^2$$

本のページ **p56**

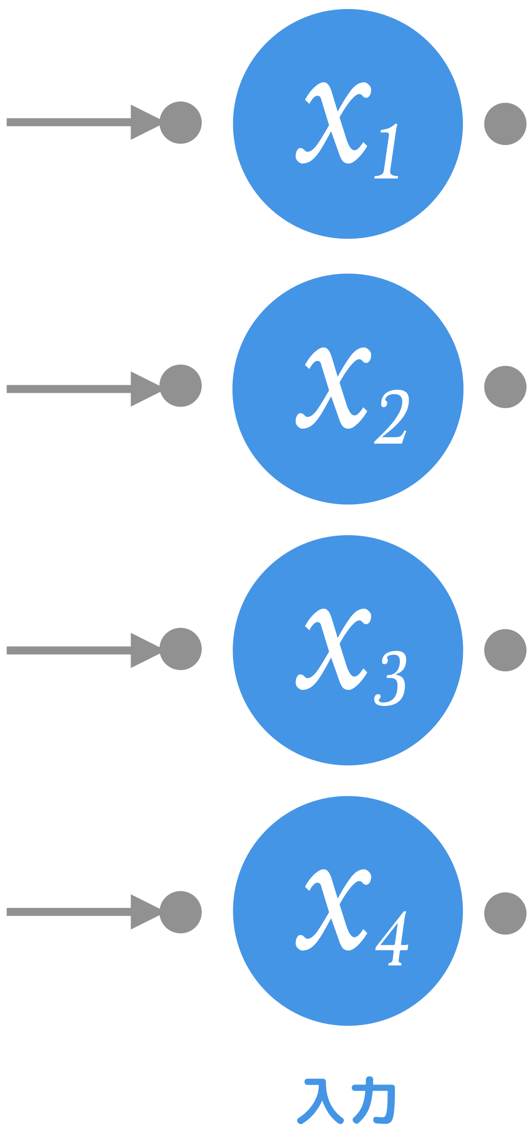
今まで見てきたニューラルネット



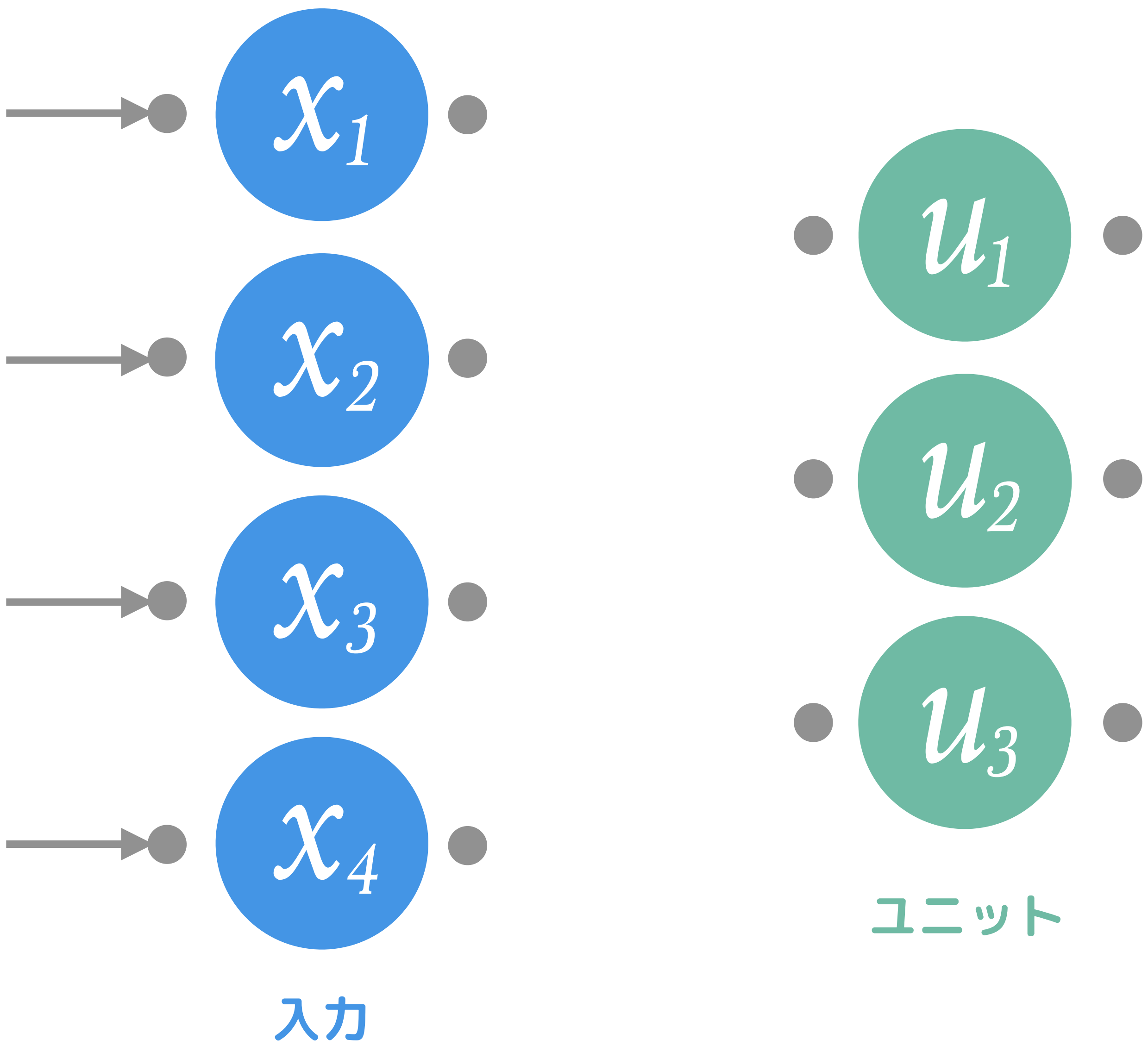
$$E(\boldsymbol{W}) = \frac{1}{2}\sum_{n=1}^{N}\|d_n - y(\boldsymbol{x}_n)\|^2$$

自己符号化器

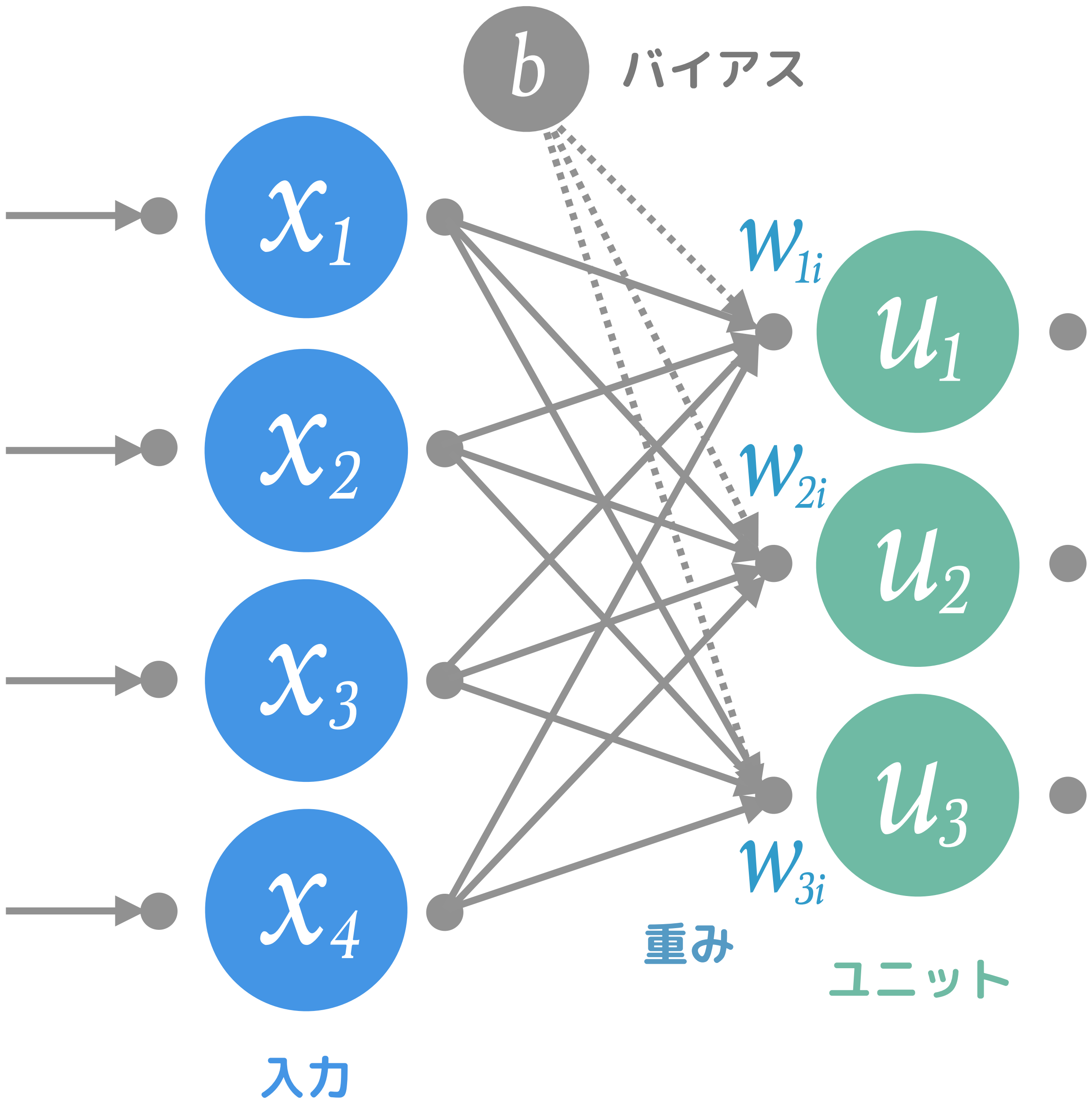
自己符号化器



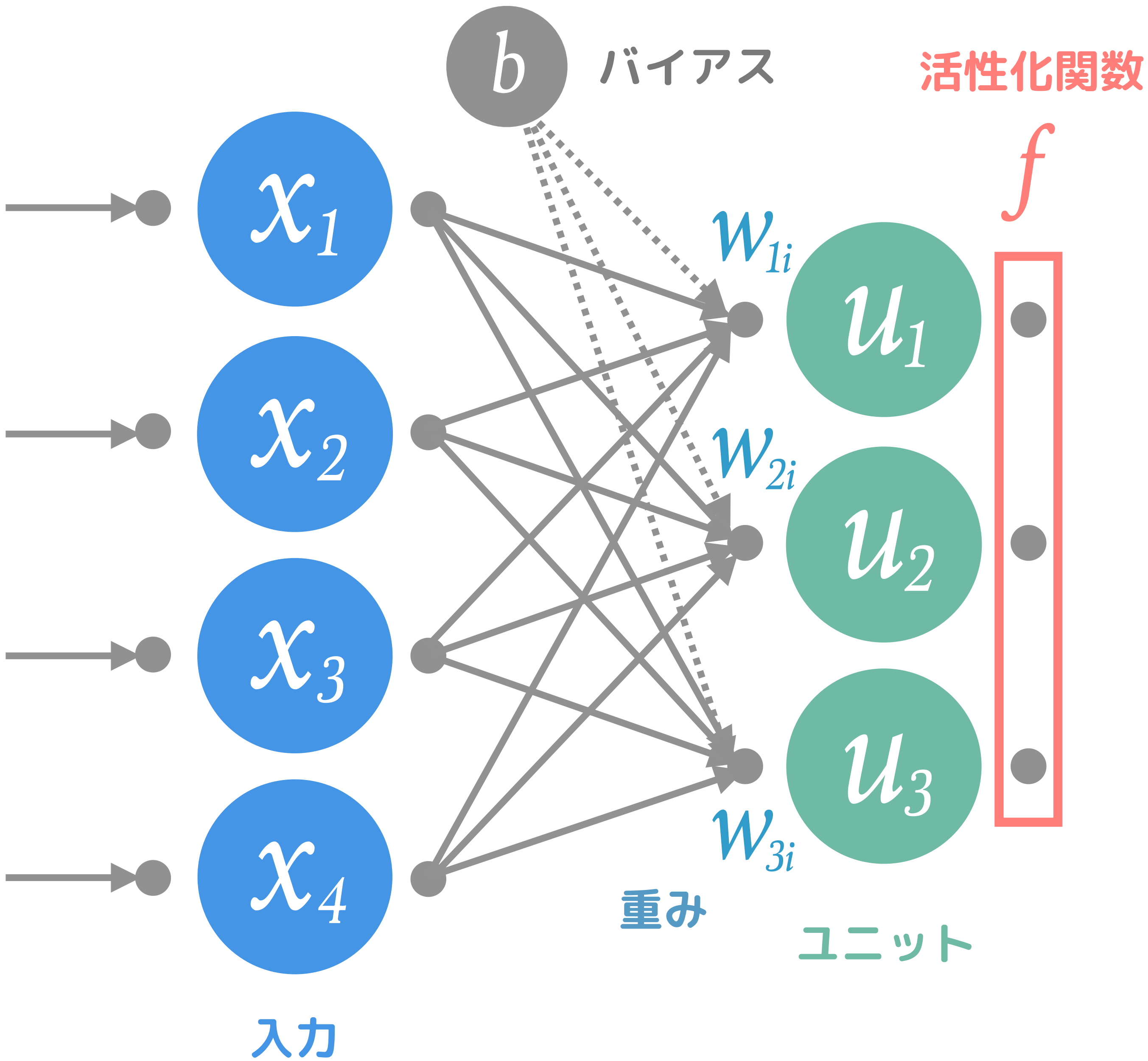
自己符号化器



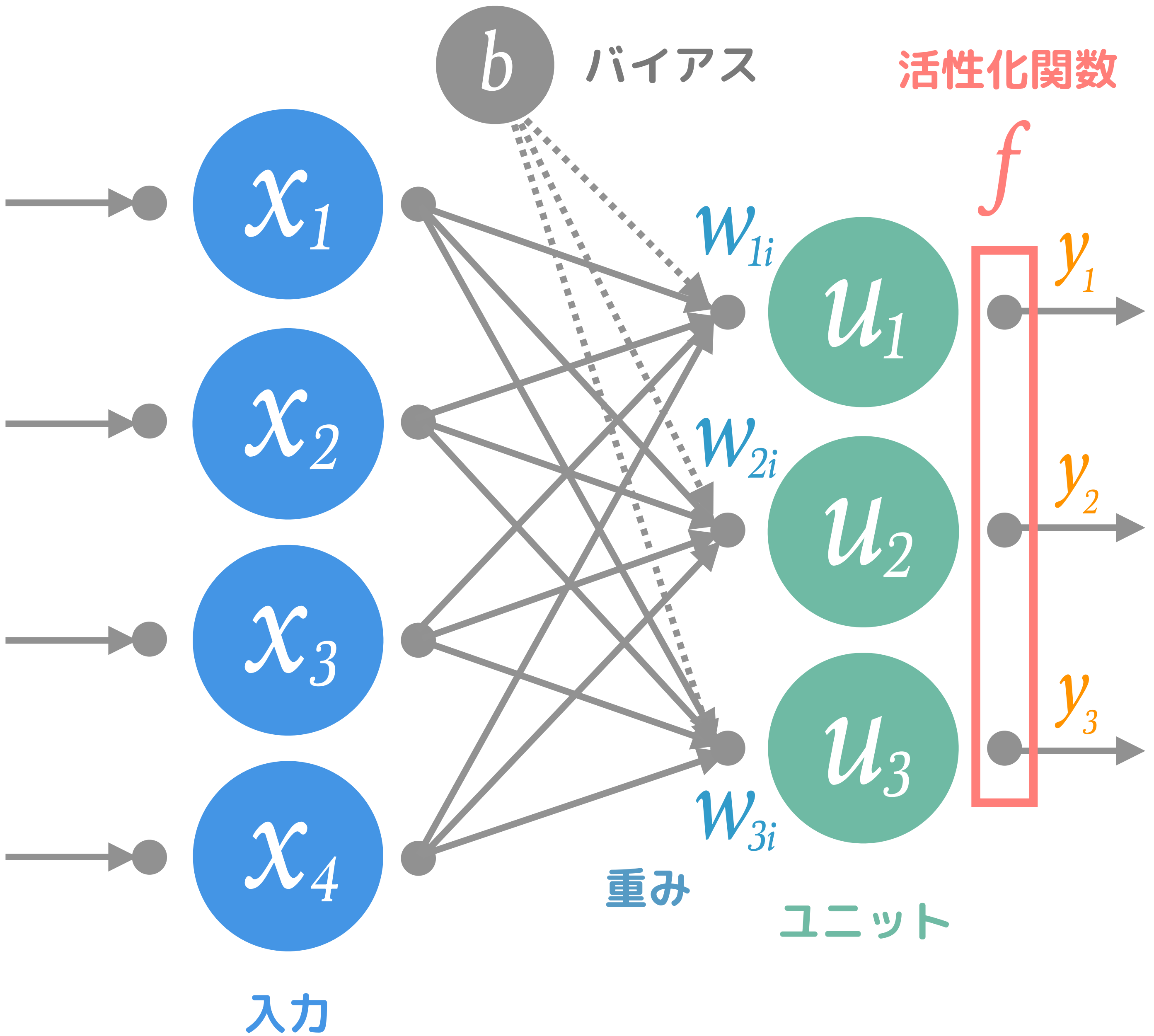
自己符号化器



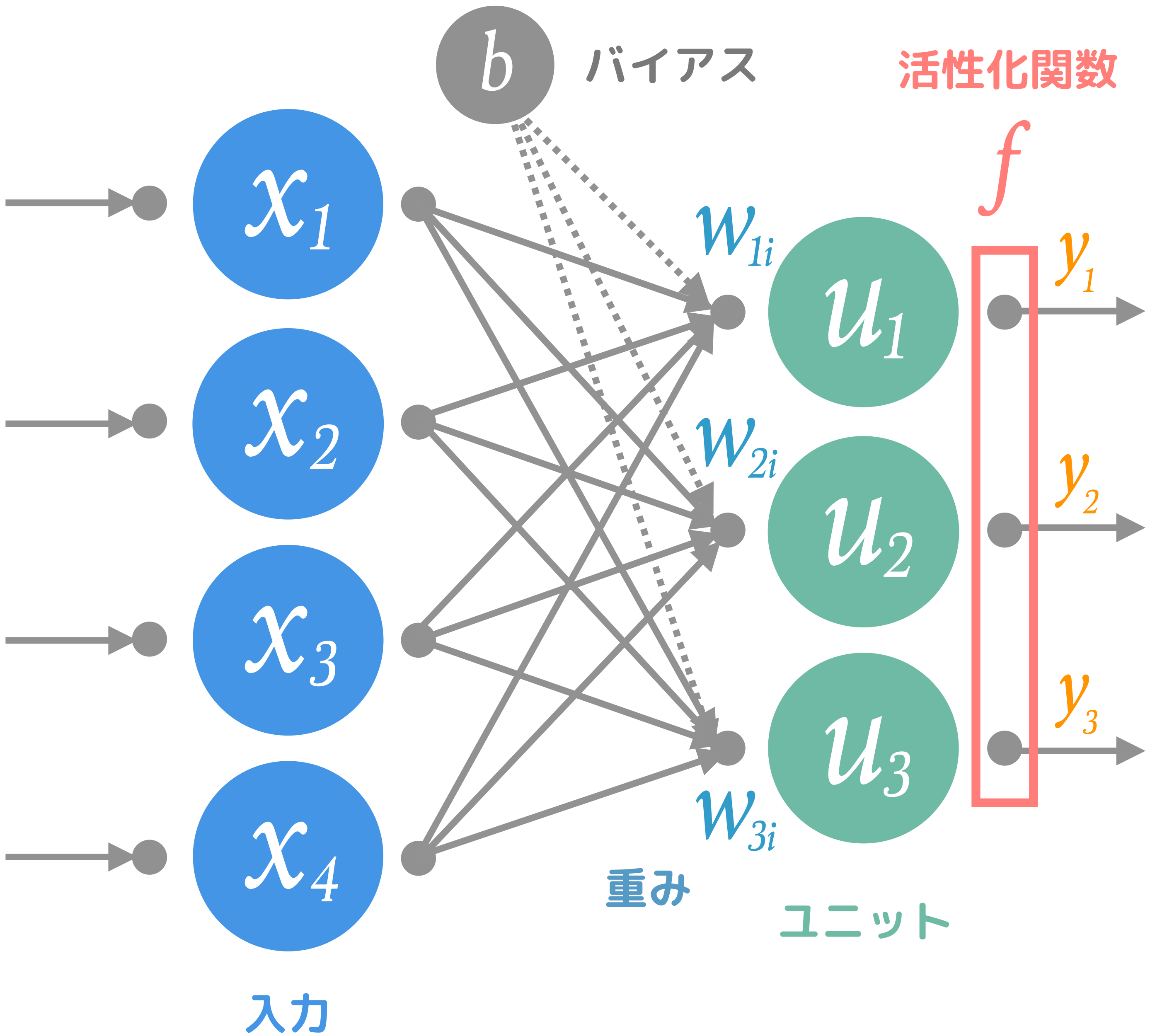
自己符号化器



自己符号化器

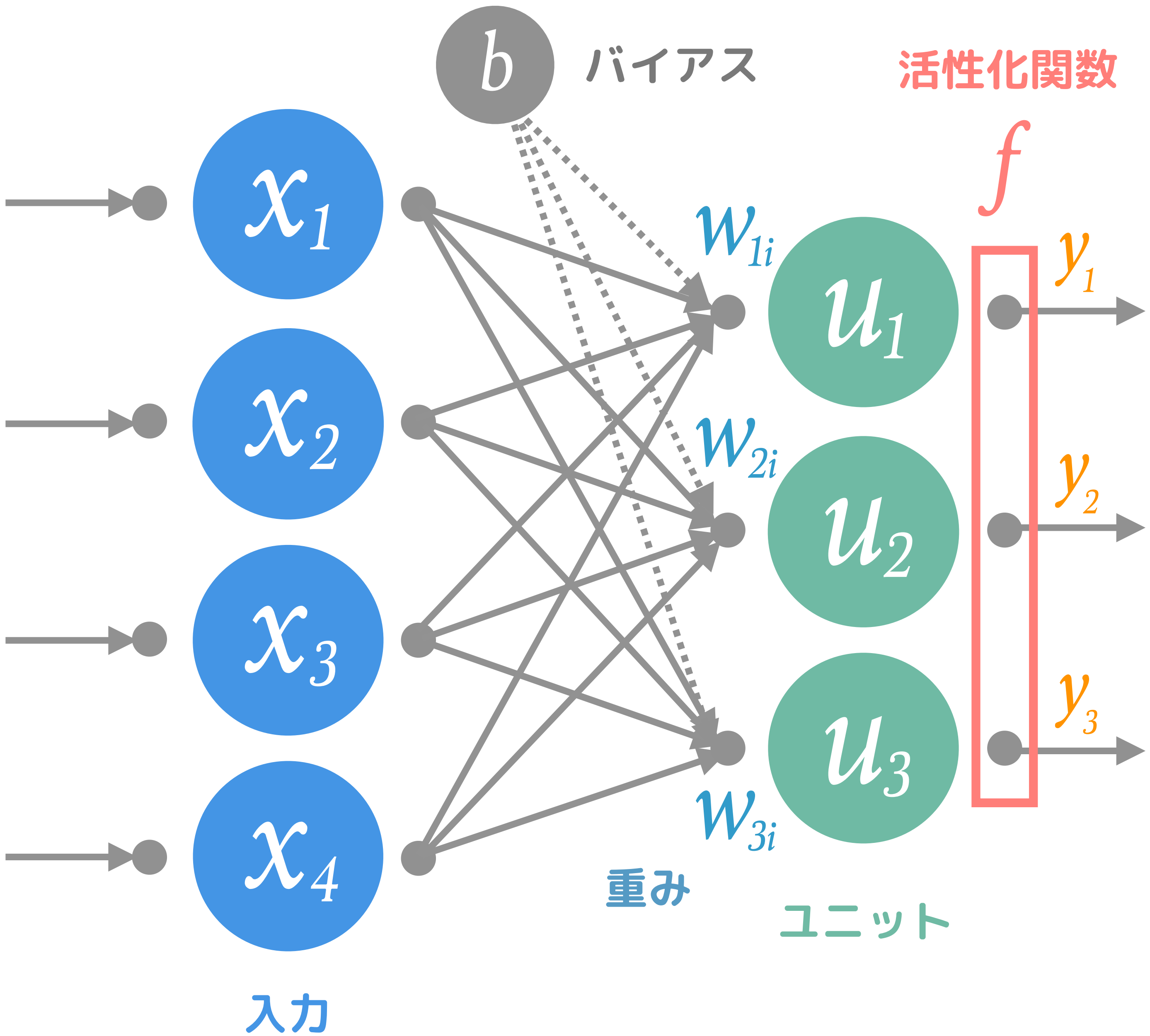


自己符号化器

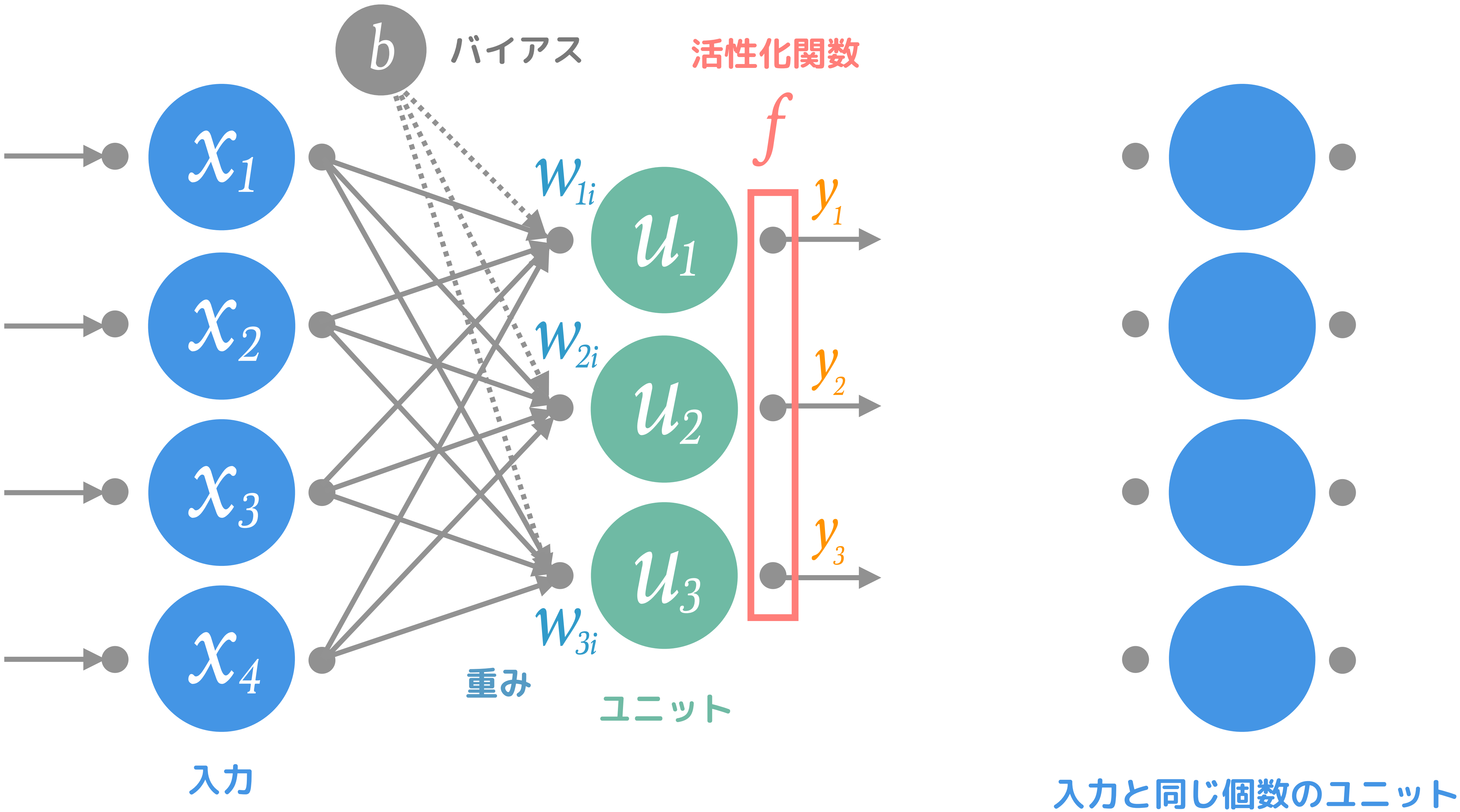


ここまでは同じ

自己符号化器

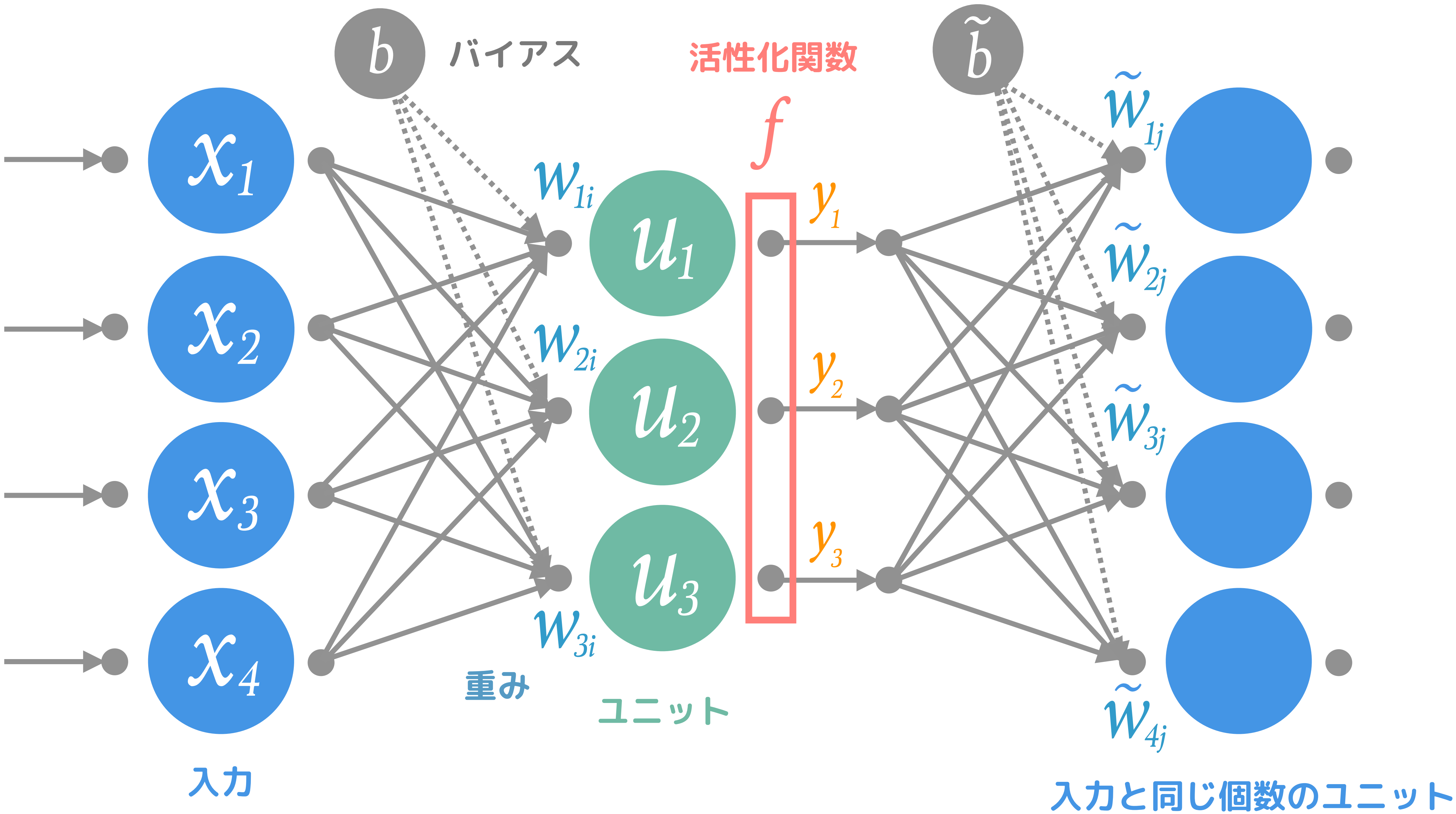


自己符号化器



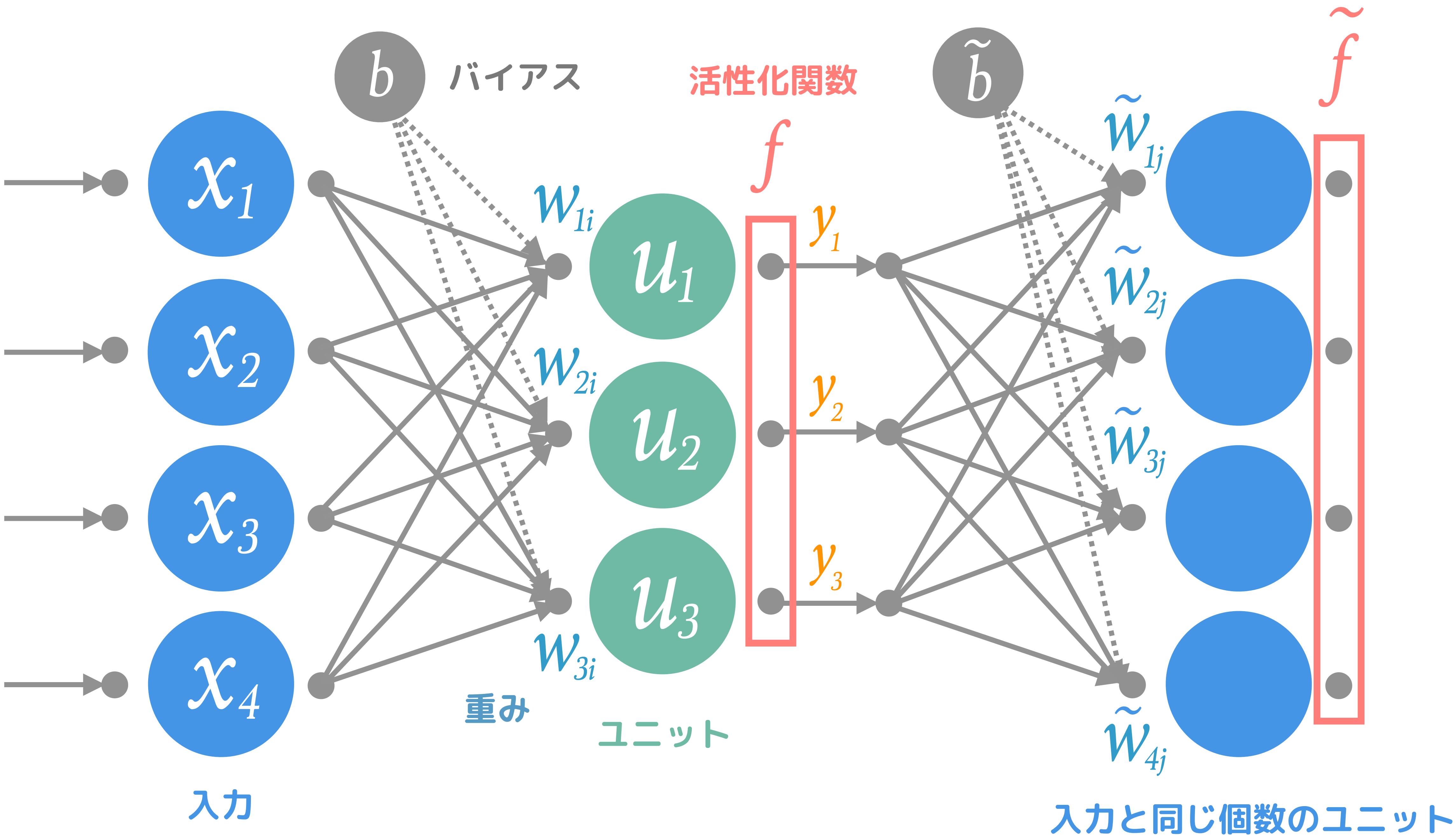
b
バイアス
活性化関数
f
x_1
x_2
x_3
x_4
w_{1i}
w_{2i}
w_{3i}
u_1
u_2
u_3
y_1
y_2
y_3
重み
ユニット
入力
入力と同じ個数のユニット

自己符号化器



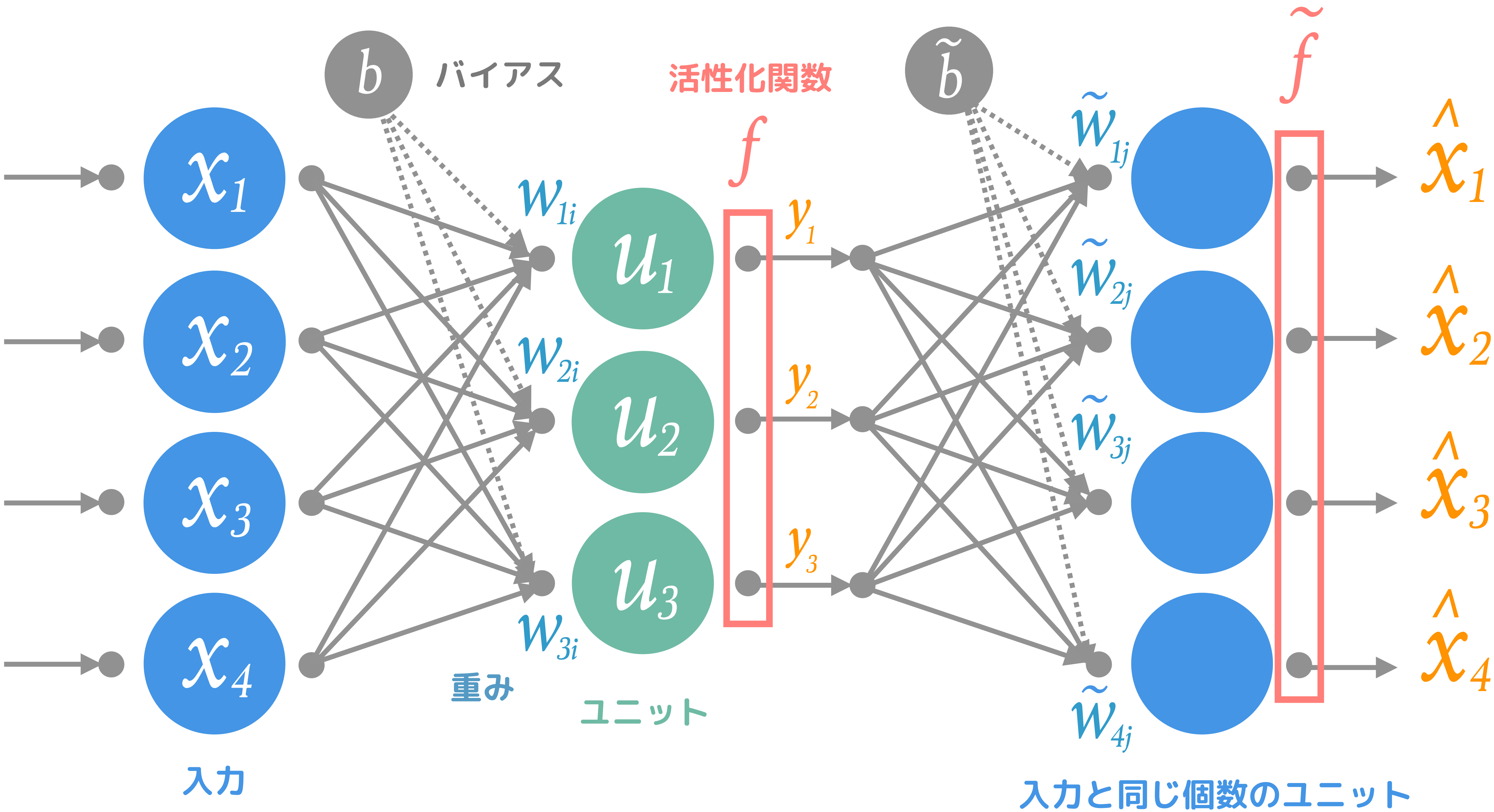
b
バイアス
活性化関数
f
$\tilde{b}$
x_1
x_2
x_3
x_4
w_{1i}
w_{2i}
w_{3i}
u_1
u_2
u_3
y_1
y_2
y_3
$\tilde{w}_{1j}$
$\tilde{w}_{2j}$
$\tilde{w}_{3j}$
$\tilde{w}_{4j}$
重み
ユニット
入力
入力と同じ個数のユニット

自己符号化器



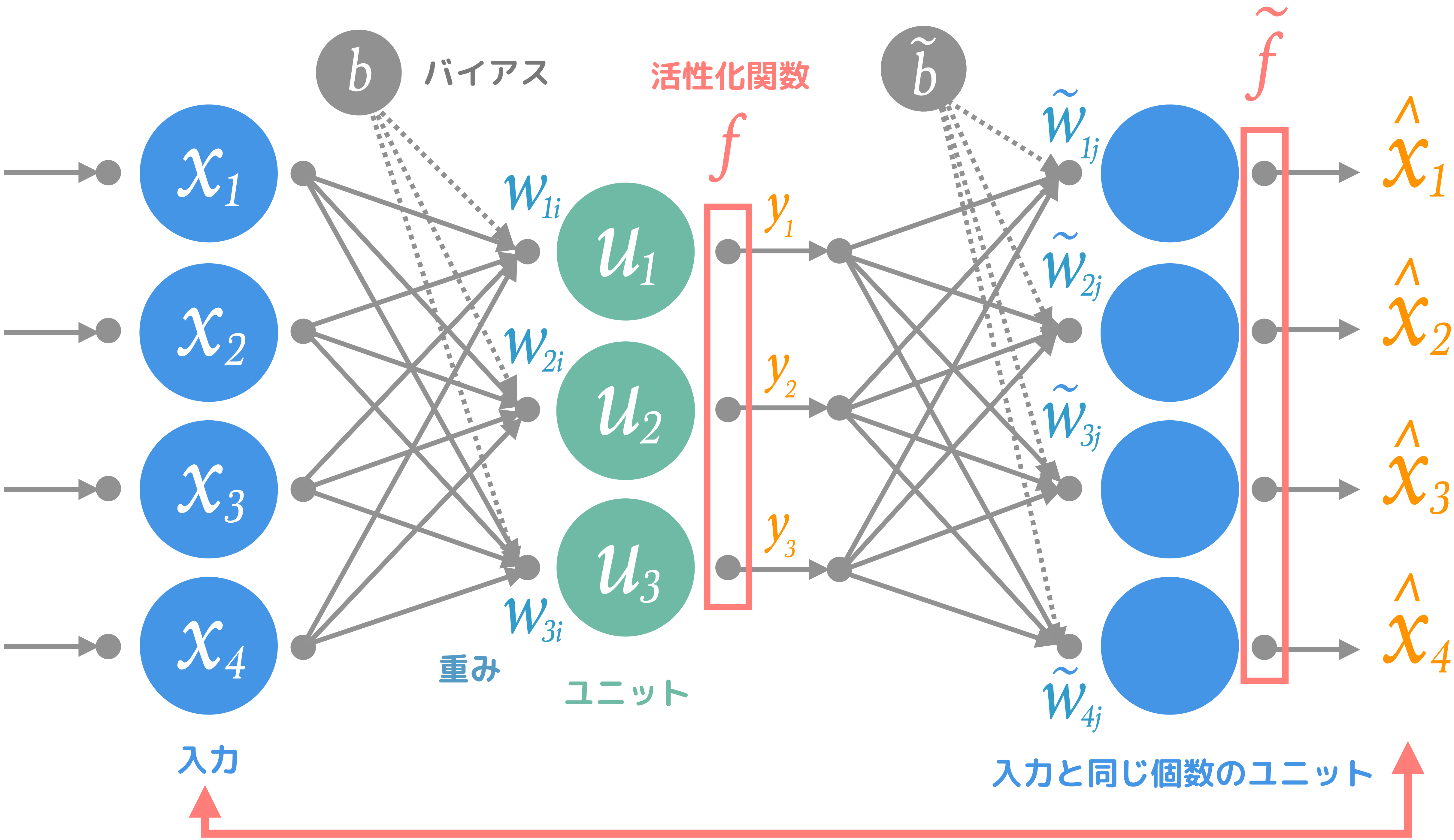
b
バイアス
活性化関数
f
$\tilde{b}$
$\tilde{f}$
x_1
x_2
x_3
x_4
w_{1i}
w_{2i}
w_{3i}
u_1
u_2
u_3
y_1
y_2
y_3
$\tilde{w}_{1j}$
$\tilde{w}_{2j}$
$\tilde{w}_{3j}$
$\tilde{w}_{4j}$
重み
ユニット
入力
入力と同じ個数のユニット

自己符号化器



本のページ p56

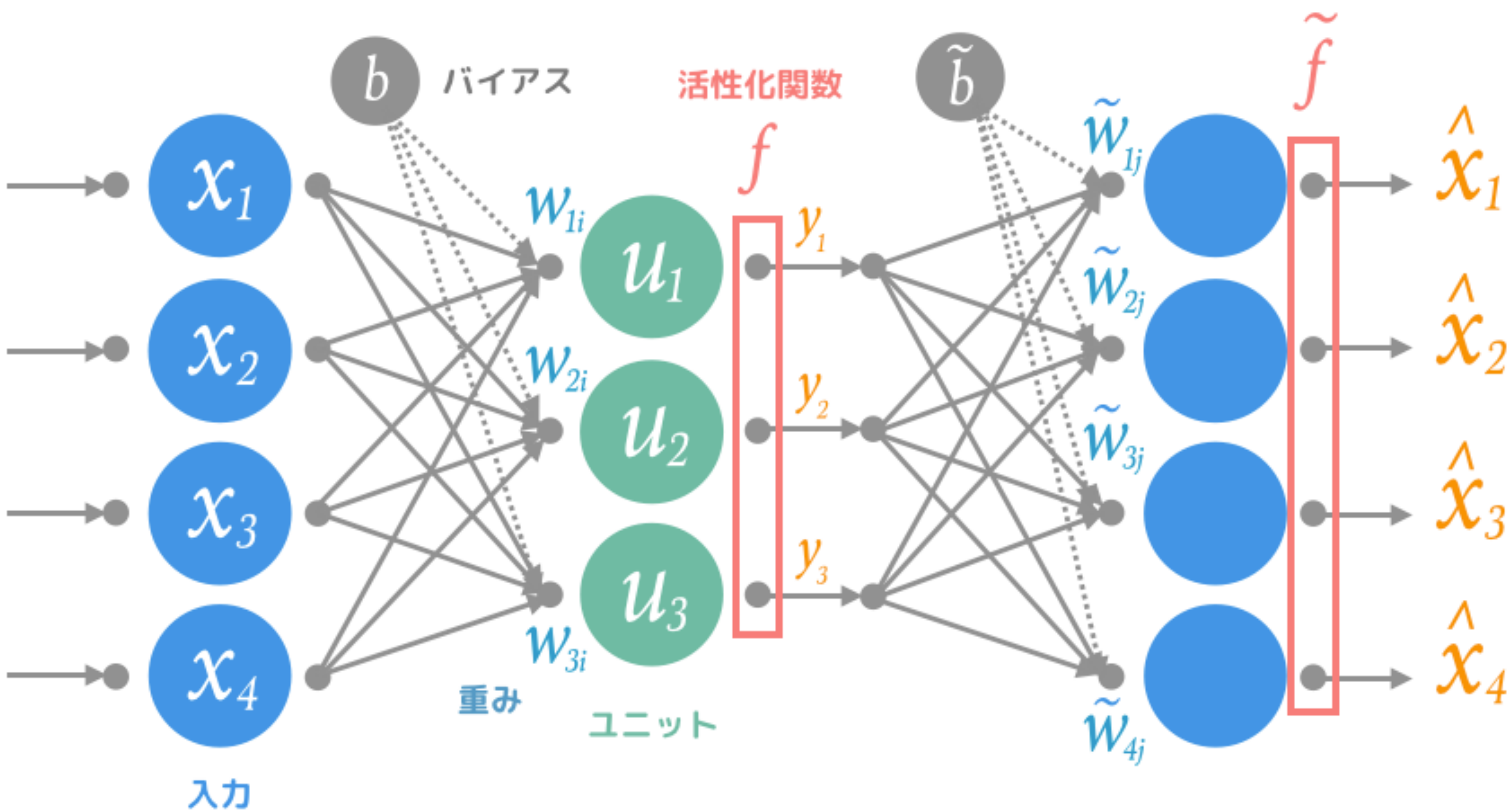
自己符号化器



入力と出力の誤差が最小になるよう学習する！

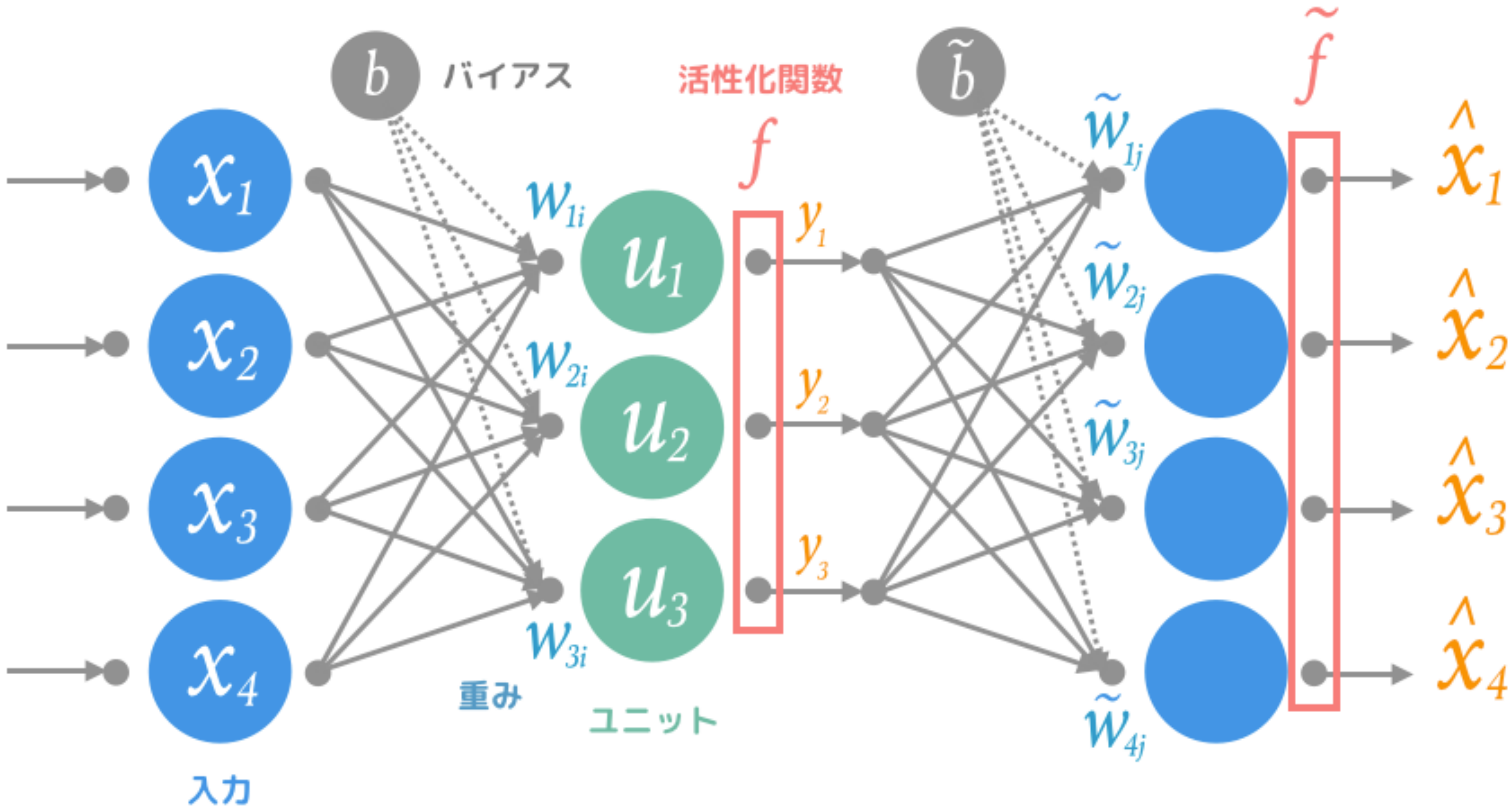
本のページ p56

自己符号化器



b
バイアス
活性化関数
$\tilde{b}$
$\tilde{f}$
x_1
x_2
x_3
x_4
w_{1i}
w_{2i}
w_{3i}
u_1
u_2
u_3
f
y_1
y_2
y_3
$\tilde{w}_{1j}$
$\tilde{w}_{2j}$
$\tilde{w}_{3j}$
$\tilde{w}_{4j}$
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
重み
ユニット
入力

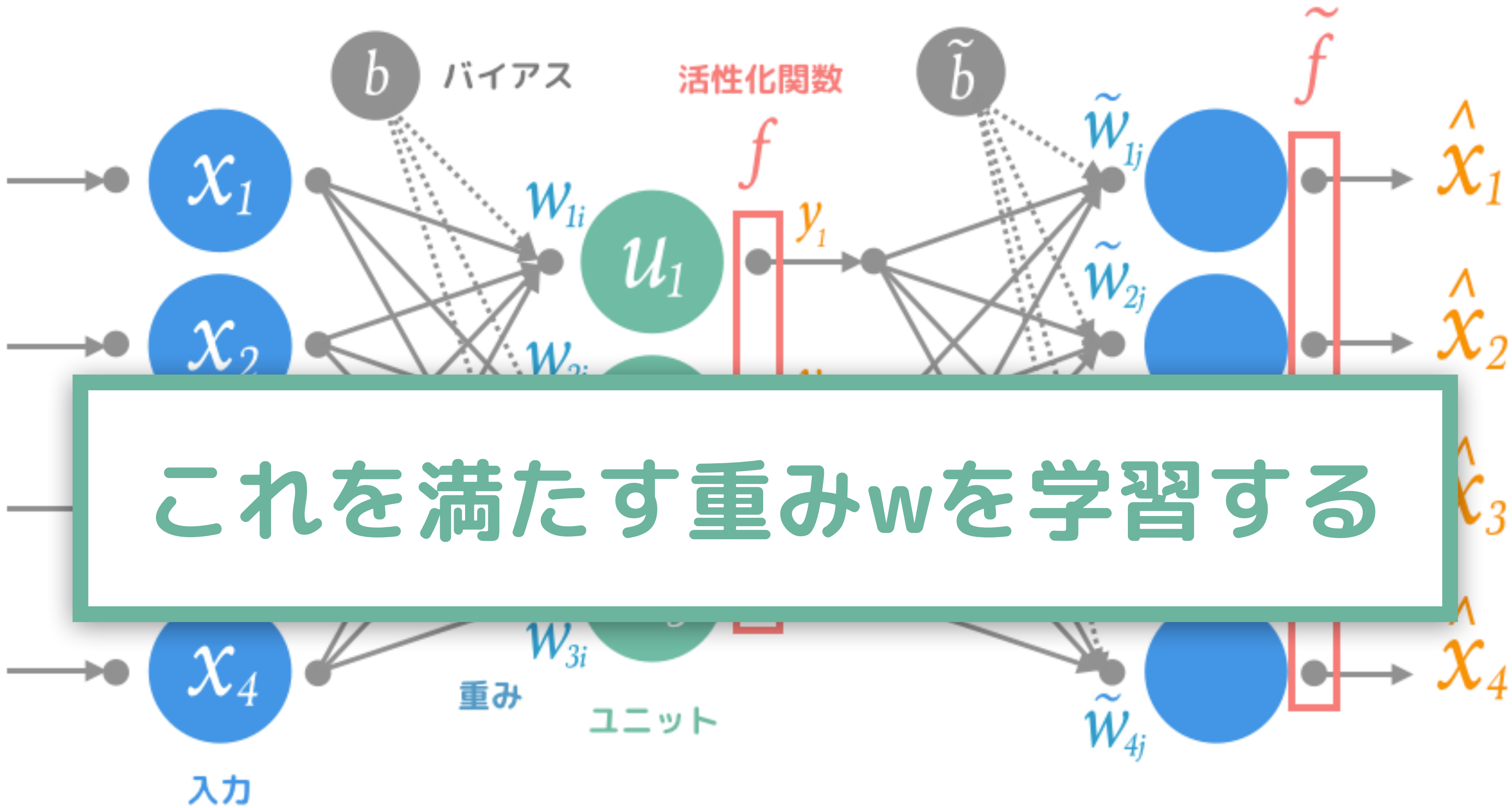
自己符号化器



入力 → 符号 → 元に戻す

符号化　　　復号化

自己符号化器



これを満たす重みwを学習する

入力 → 符号 → 元に戻す

符号化　　復号化

なんでこんなことするの?

元に戻して何がおいしいの？

「5.2 ネットワークの設計」は
手法の説明なので飛ばします

自己符号化器のおいしい部分

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

② ディープネットの事前学習

・上手く学習できるような初期値を得る

自己符号化器のおいしい部分

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

今はここに注目！

② ディープネットの事前学習

・上手く学習できるような初期値を得る

本のページ p58

特徴の獲得

データを良く表す「特徴」って何？

本のページ　p58

特徴の獲得

データを良く表す「特徴」って何？

同じデータを
より少ない次元で表したとき、
その少ない次元の基底ベクトル

特徴の獲得

データを良く表す「特徴」って何？

同じデータを
より少ない次元で表したとき、
その少ない次元の基底ベクトル

わかりづらい…

特徴の獲得

デー … 何？

ひと言でいうと

次元を落とす

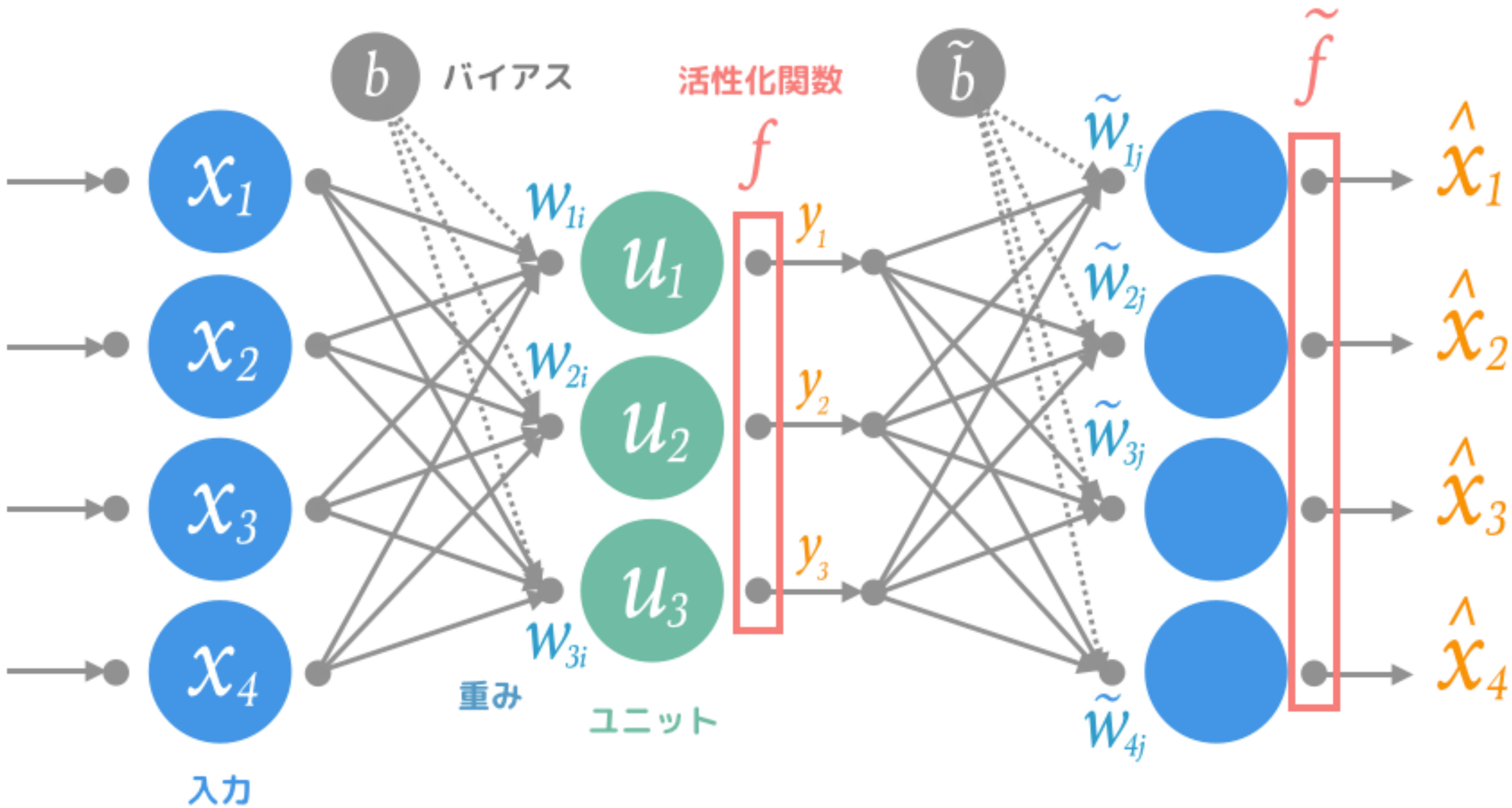
ということ

より … き、

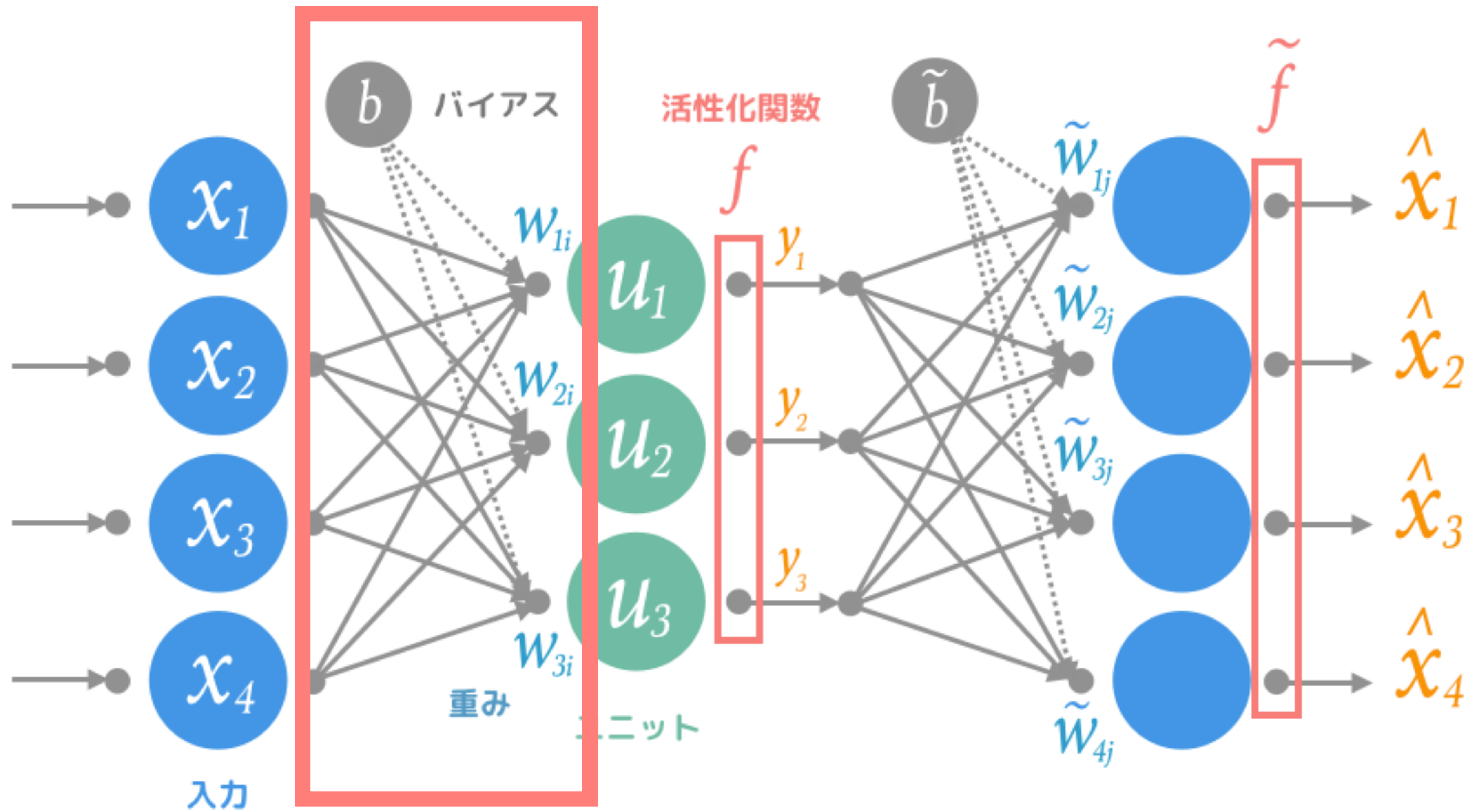
その少ない次元の基底ベクトル

わかりづらい…

特徴とはどの部分か



特徴とはどの部分か

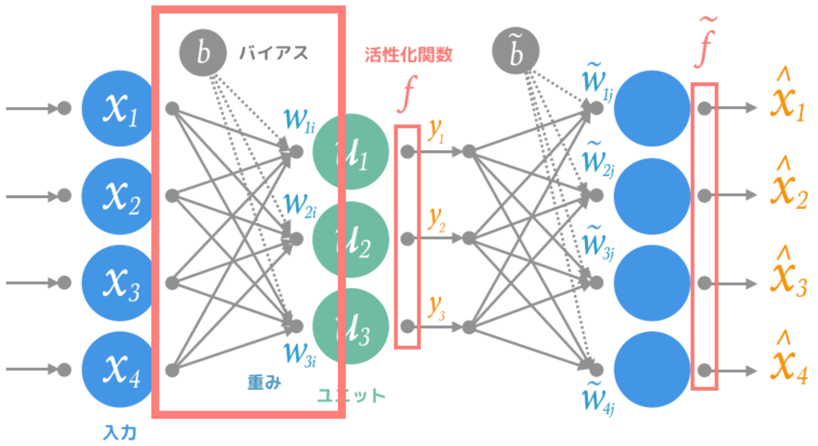


この「重み」と「バイアス」を「特徴」という

自己符号化器が学習によって決定する部分

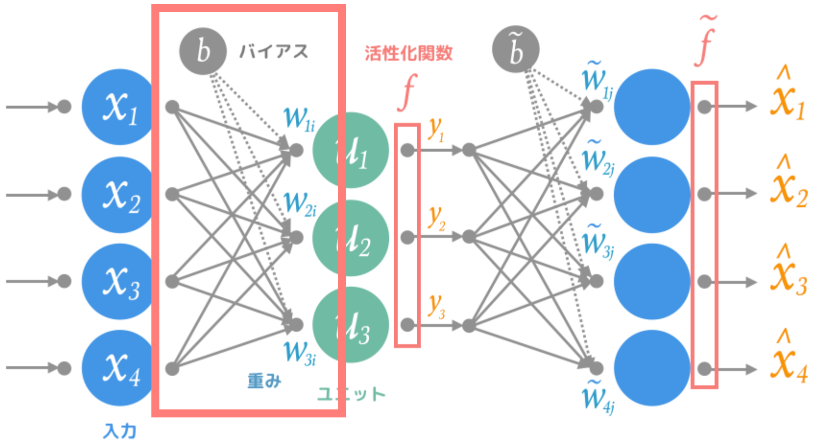
特徴とは

なぜこれが特徴なのか



特徴とは

なぜこれが特徴なのか →

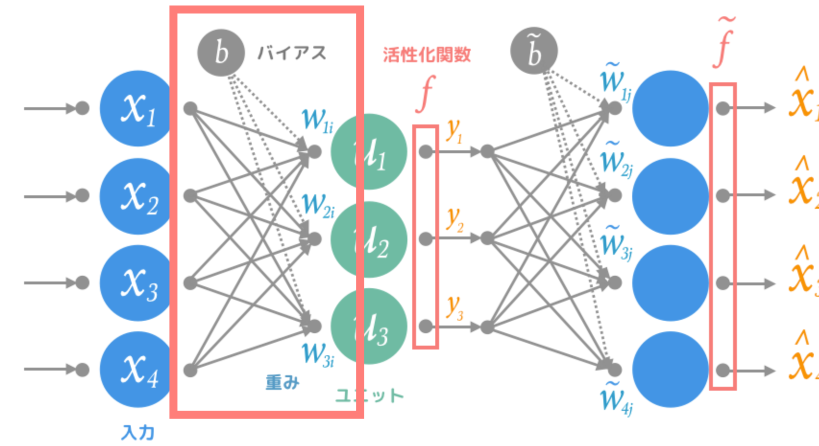


引用（p58中段） → 音読しましょう

yの計算は、重み行列Wとxの積から始まりますが、
これはWの各行ベクトルとxの内積の計算であって、
そこでは**入力xにWの各行ベクトル表す成分が**
どれだけ含まれているかを取り出していると言えます。

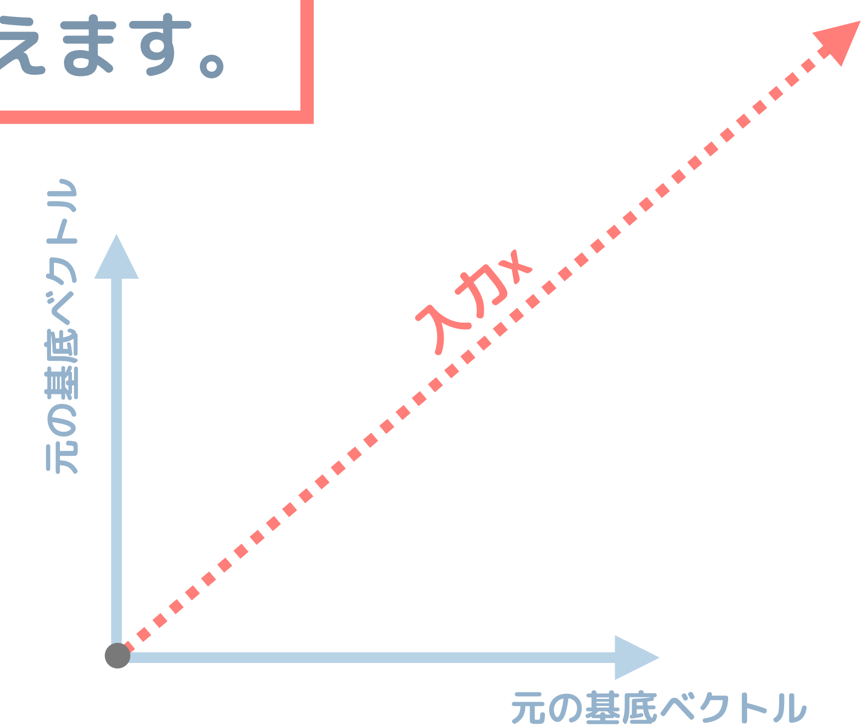
特徴とは

なぜこれが特徴なのか →



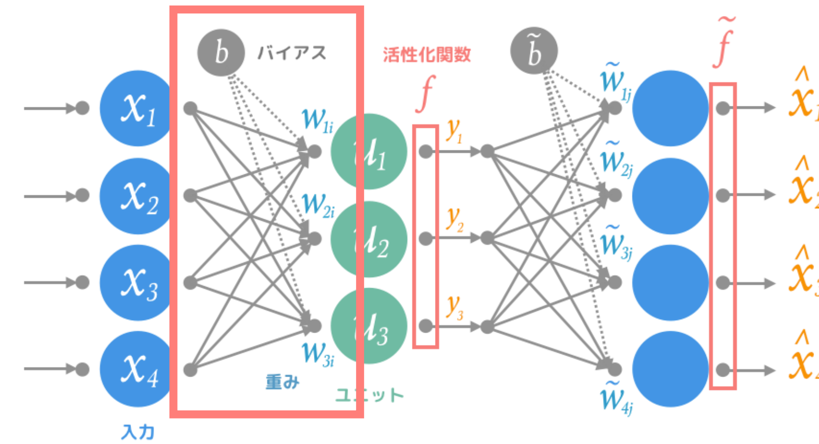
引用（p58中段） → 音読しましょう

yの計算は、重み行列Wとxの積から始まりますが、
これはWの各行ベクトルとxの内積の計算であって、
そこでは**入力xにWの各行ベクトル表す成分がどれだけ含まれているか**を取り出していると言えます。



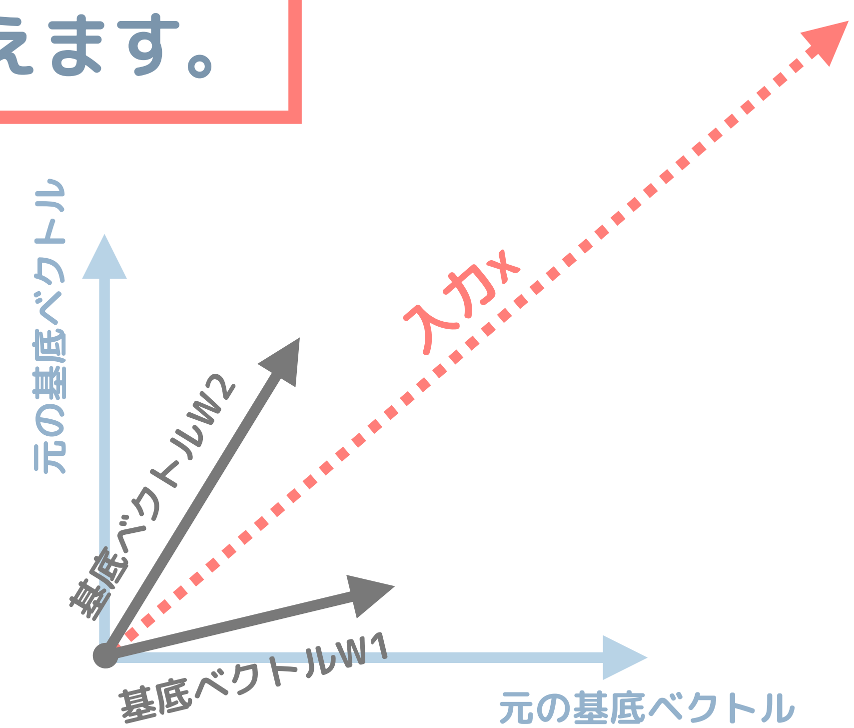
特徴とは

なぜこれが特徴なのか →



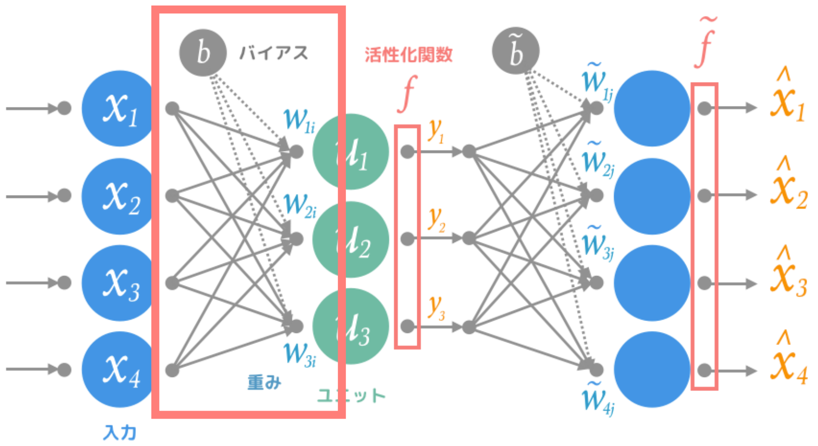
引用（p58中段）→ 音読しましょう

yの計算は、重み行列Wとxの積から始まりますが、
これはWの各行ベクトルとxの内積の計算であって、
そこでは**入力xにWの各行ベクトル表す成分がどれだけ含まれているか**を取り出していると言えます。



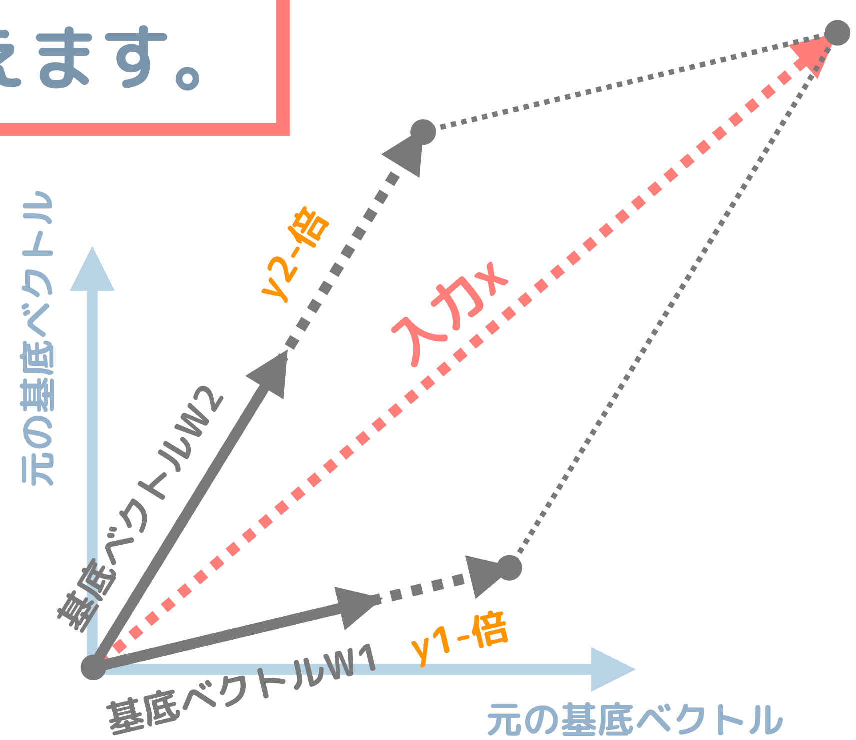
特徴とは

なぜこれが特徴なのか



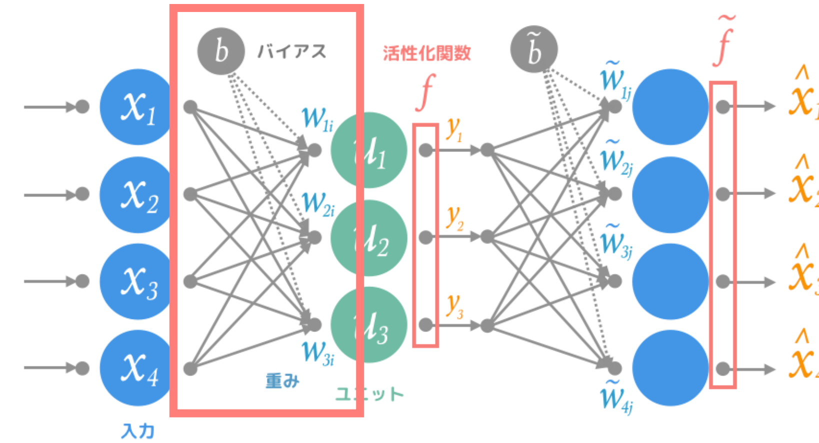
引用（p58中段）→ 音読しましょう

yの計算は、重み行列Wとxの積から始まりますが、
これはWの各行ベクトルとxの内積の計算であって、
そこでは入力xにWの各行ベクトル表す成分が
どれだけ含まれているかを取り出していると言えます。



特徴とは

なぜこれが特徴なのか

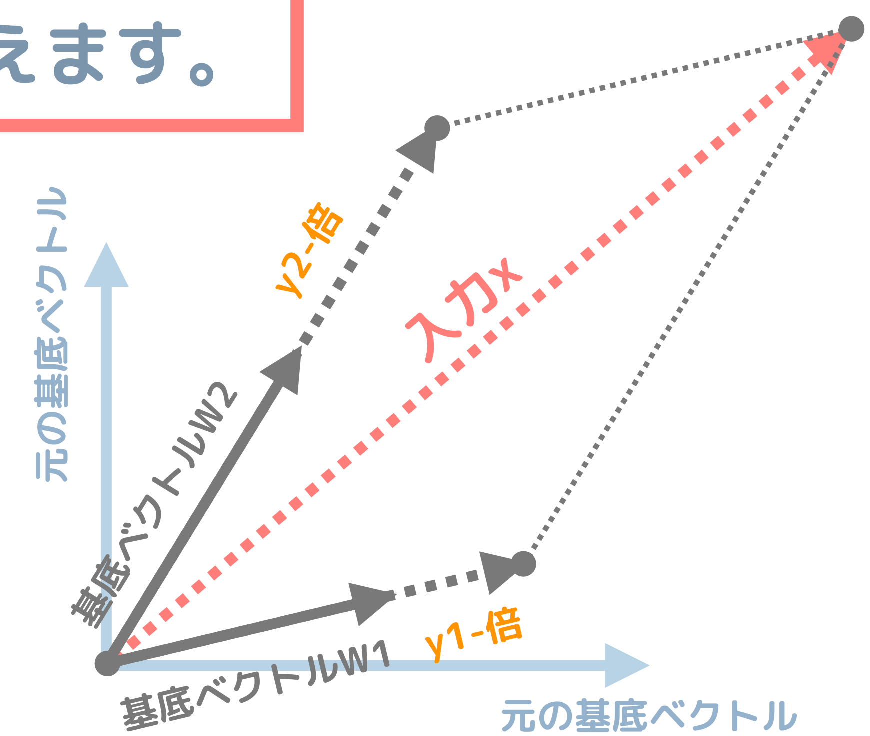


引用（p58中段）→ 音読しましょう

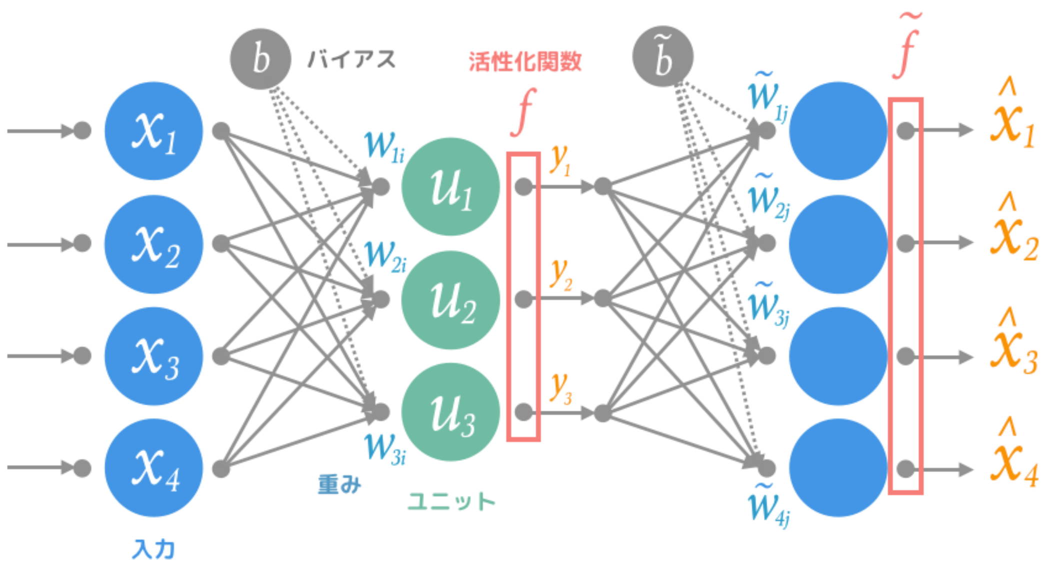
yの計算は、重み行列Wとxの積から始まりますが、
これはWの各行ベクトルとxの内積の計算であって、
そこでは入力xにWの各行ベクトル表す成分が
どれだけ含まれているかを取り出していると言えます。

元の基底ベクトルの空間ではなく、
Wの行ベクトルを基底ベクトルとした空間でxを見て、
その空間での各成分が y である

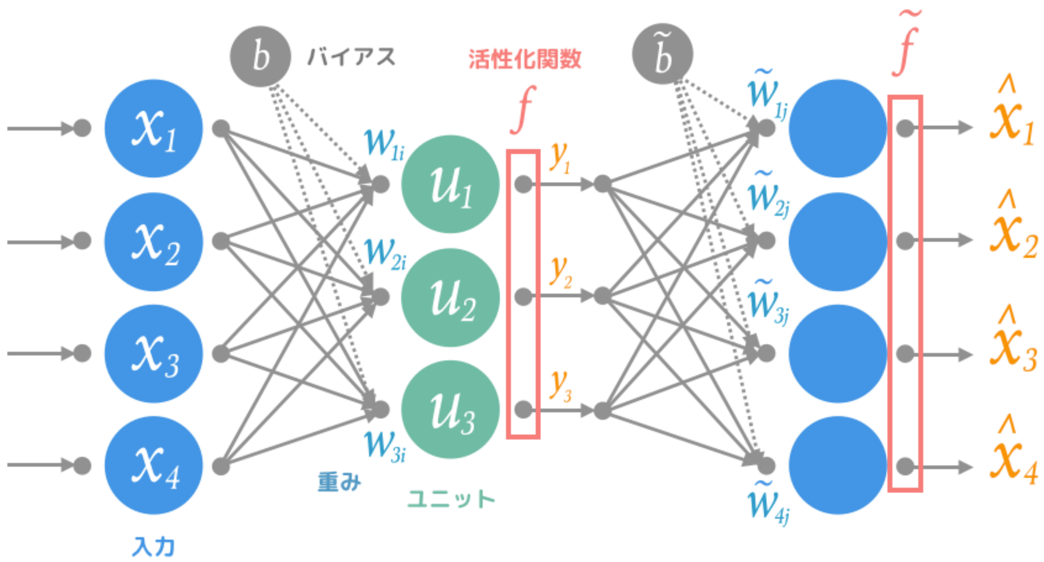
符号化することで見る空間が変わる



次元を落とす

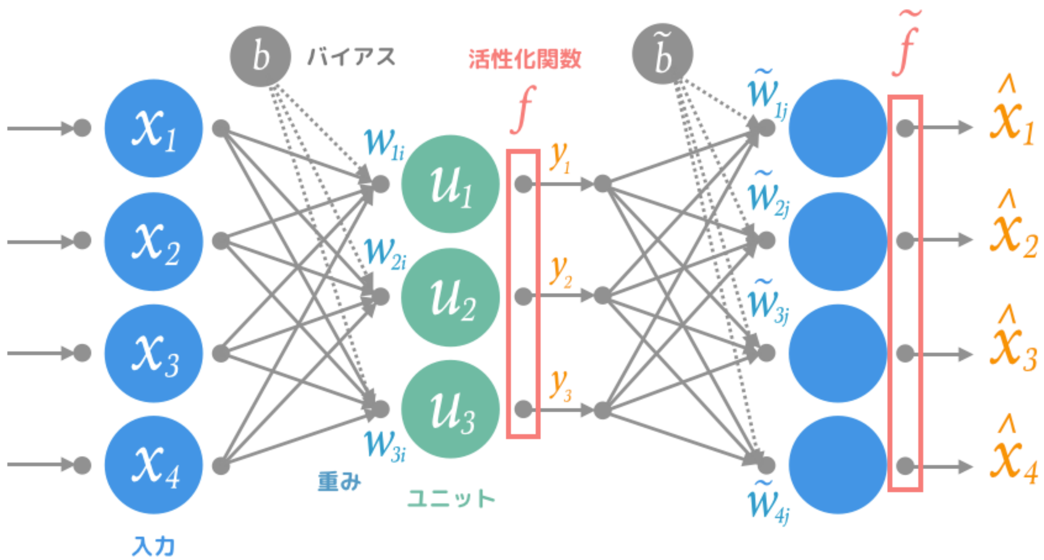


次元を落とす



さらに注目するのは

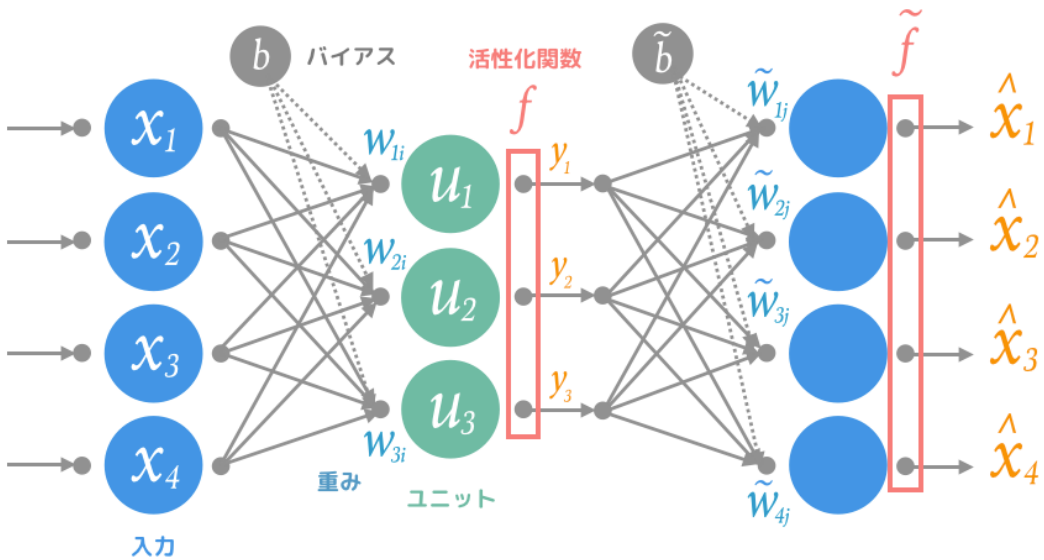
次元を落とす



さらに注目するのは

次元の数 = ユニットの数

次元を落とす

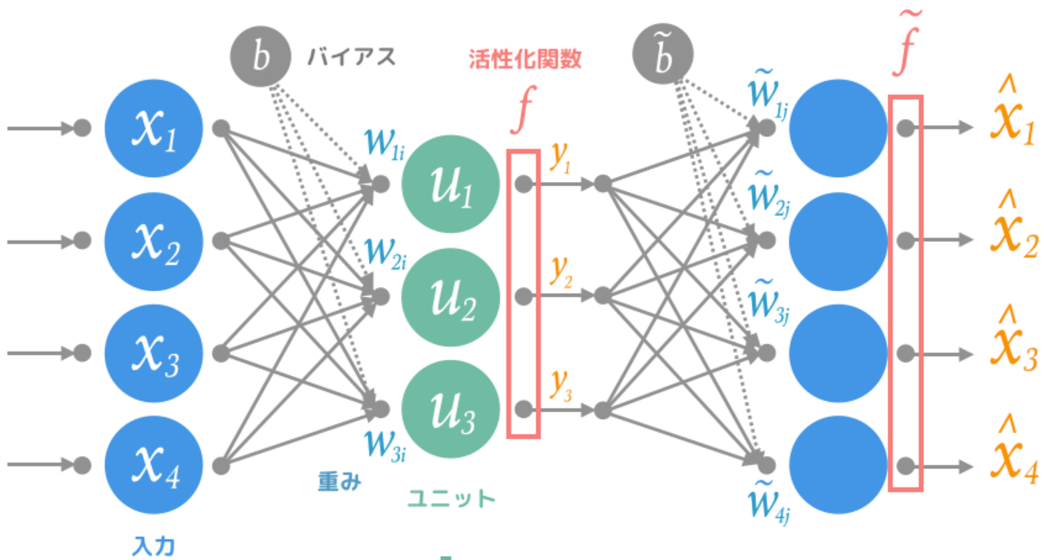


さらに注目するのは

次元の数 = ユニットの数

入力は４次元

次元を落とす



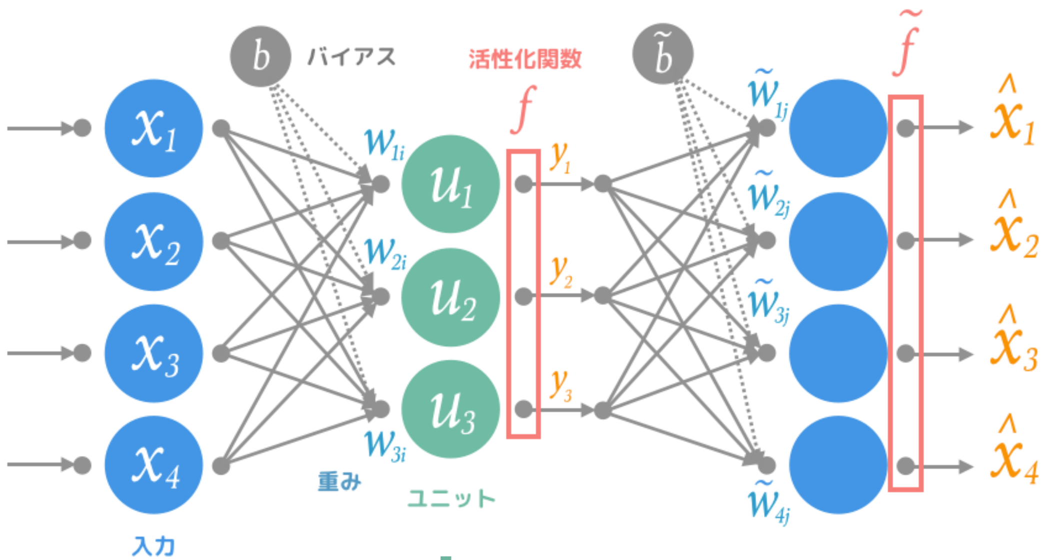
さらに注目するのは

次元の数 = ユニットの数

符号化したあとは３次元

入力は４次元

次元を落とす



さらに注目するのは

次元の数 = ユニットの数

符号化したあとは３次元

入力は４次元

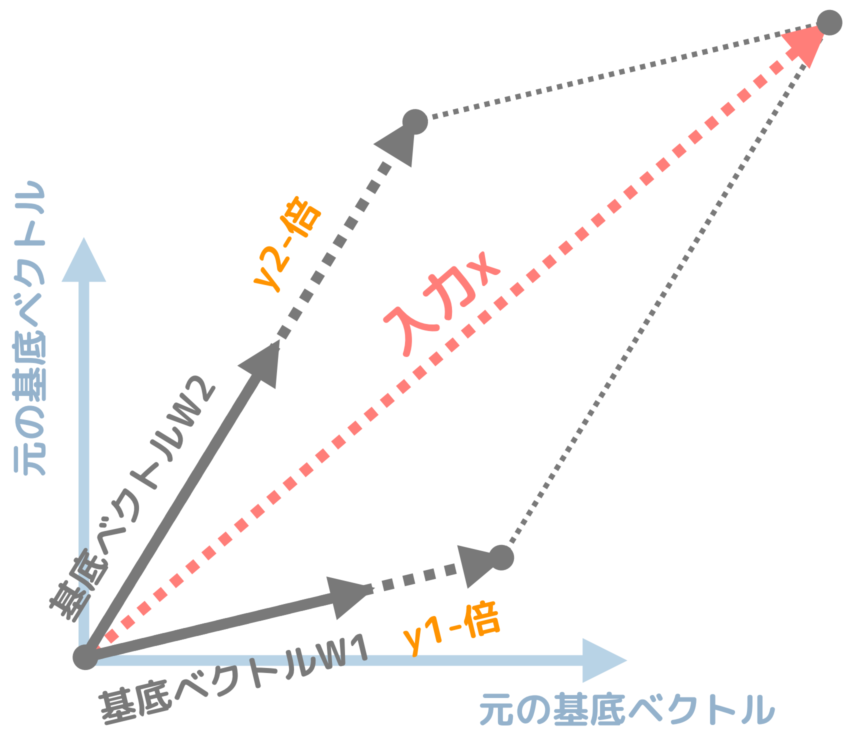
符号化した空間では
データ表現の次元が減っている！

本のページ p58

軽くまとめ

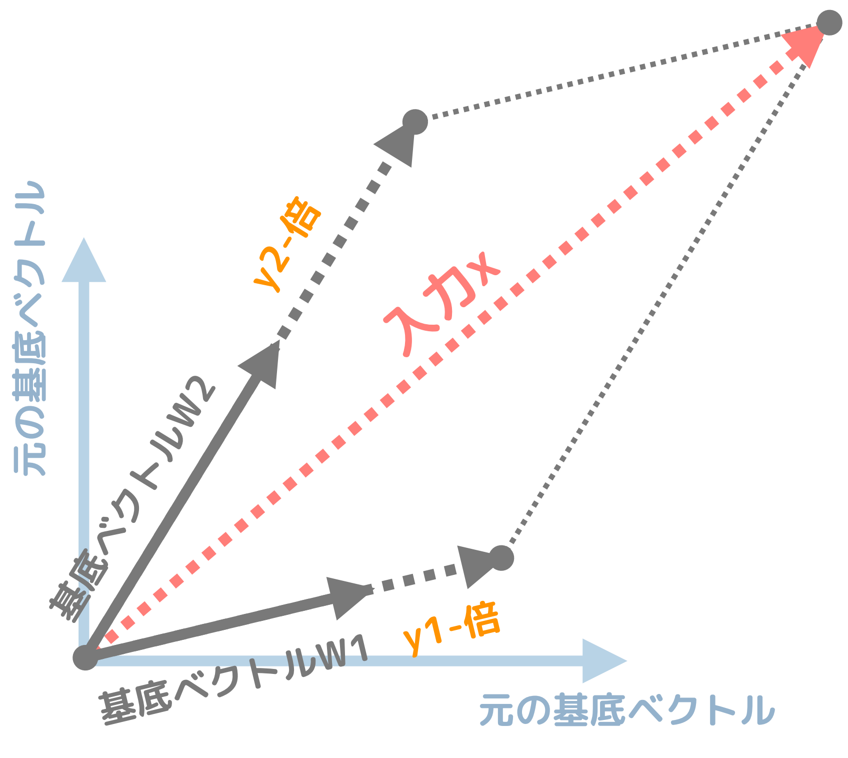
軽くまとめ

・自己符号化器の学習で得られた重み行列Wを特徴という



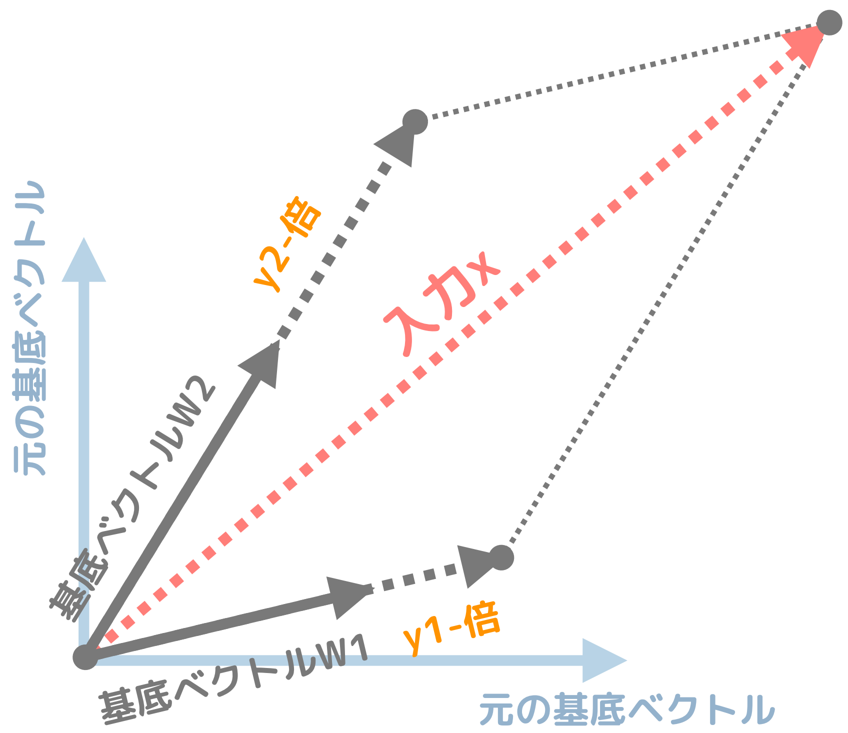
軽くまとめ

- 自己符号化器の学習で得られた重み行列Wを特徴という
- 特徴とは、新しい空間の基底ベクトルである



軽くまとめ

- 自己符号化器の学習で得られた重み行列Wを特徴という
- 特徴とは、新しい空間の基底ベクトルである
- その新しい空間は、元の空間よりも次元が少ない



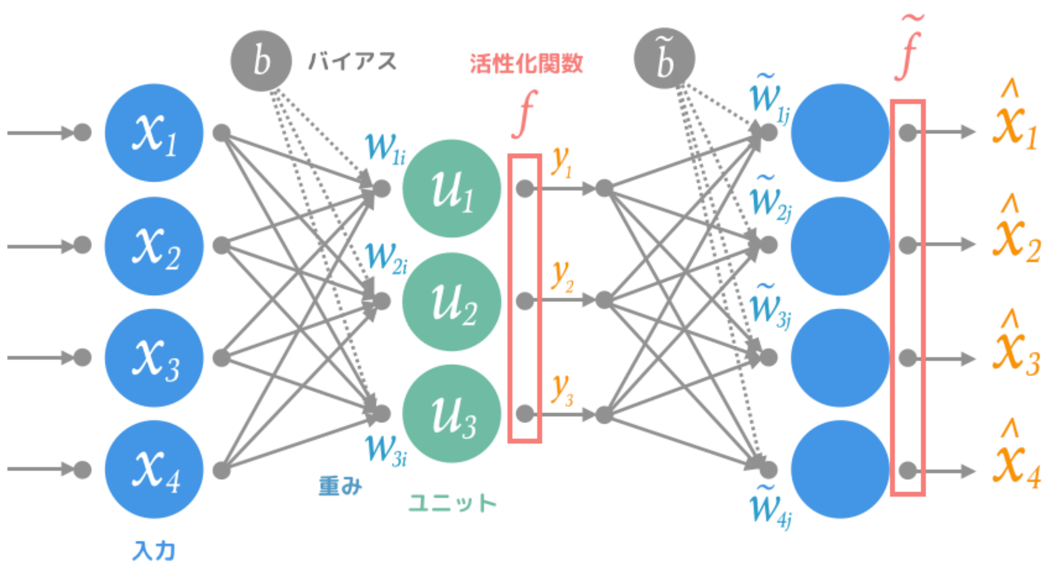
あるデータを、
より少ない次元で表せるのは、
非常に有用なこと

どうしてそんなことできるの？

なぜ次元を落とした空間での表現が可能なのか

本のページ p57

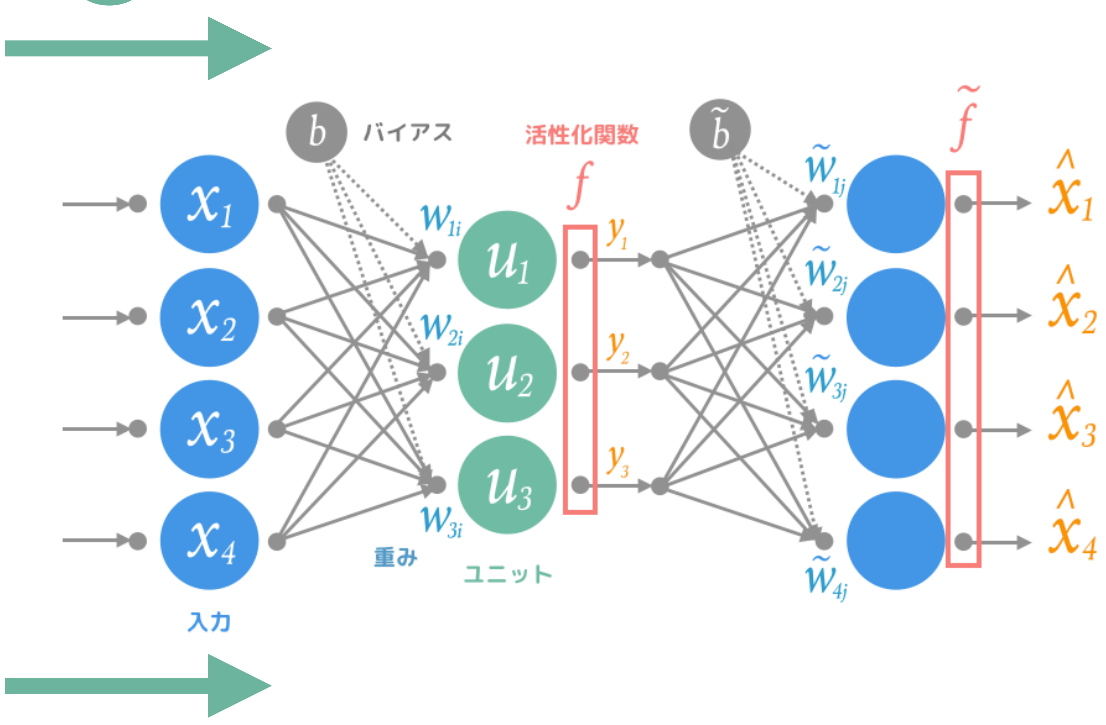
直感的な独自解釈



本のページ p**

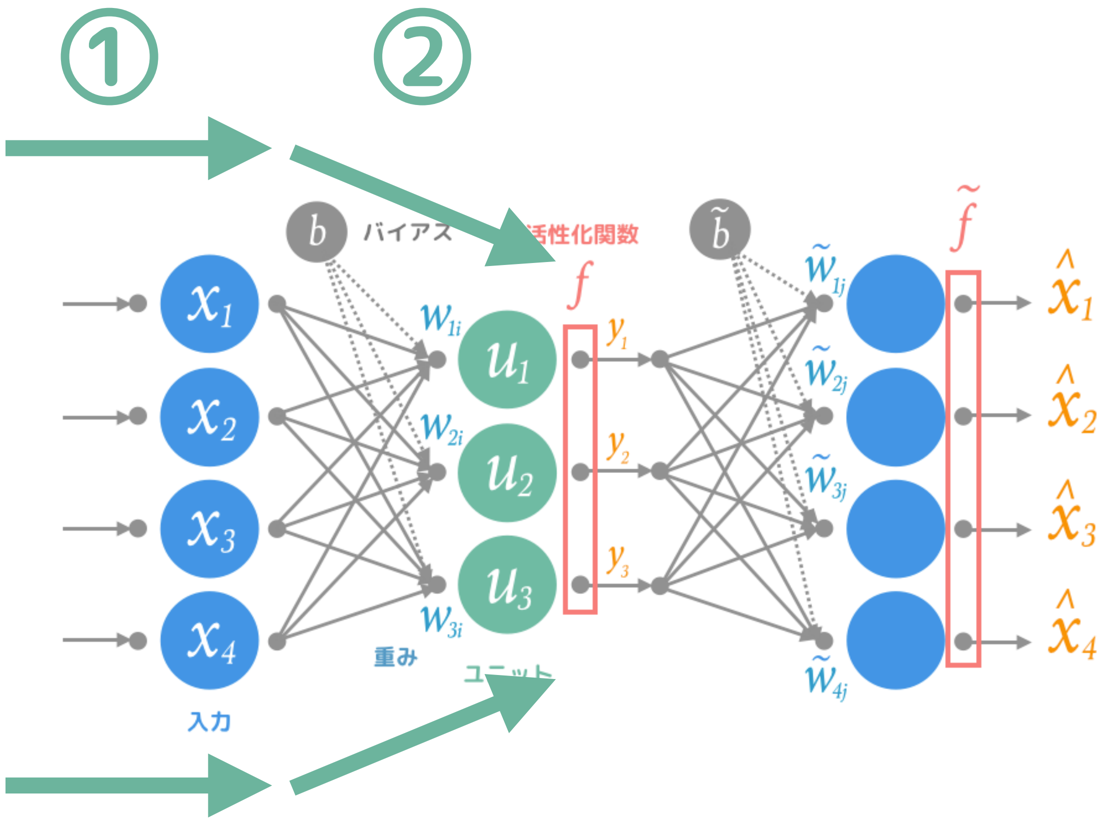
直感的な独自解釈

①



① 高次元データを入力

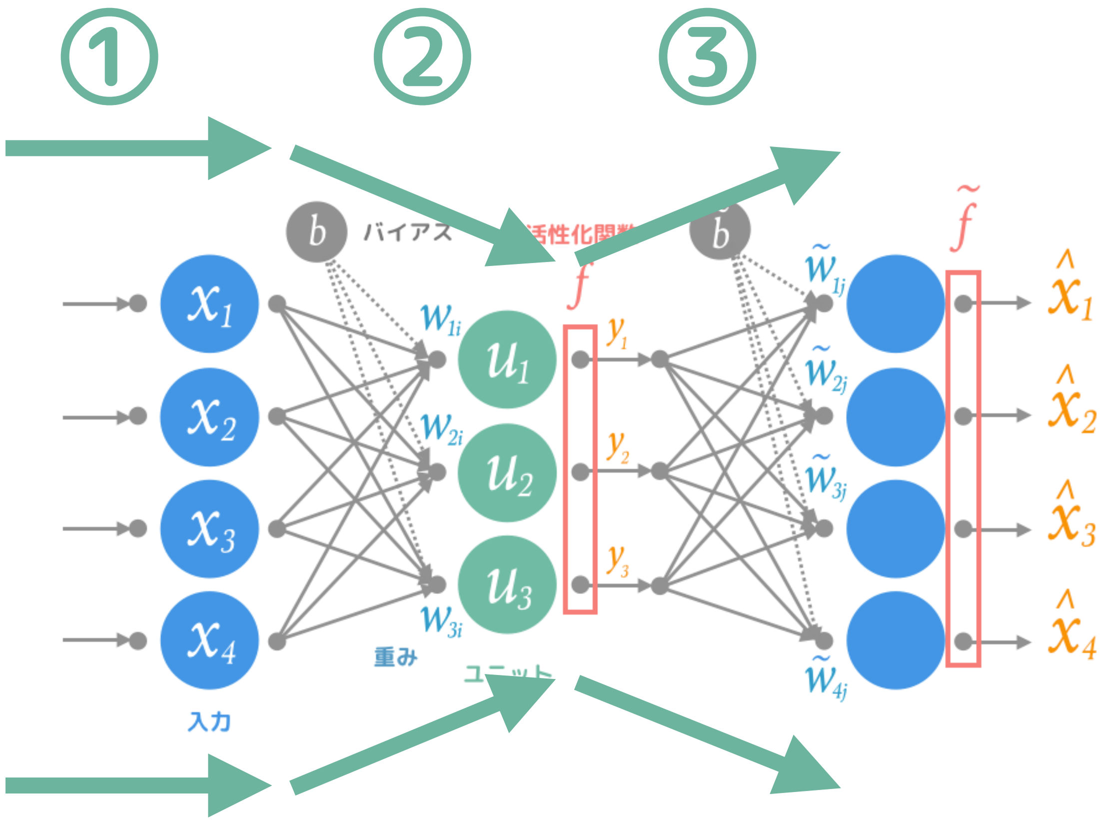
直感的な独自解釈



① 高次元データを入力

② 強制的に低次元空間に変換（符号化）

直感的な独自解釈

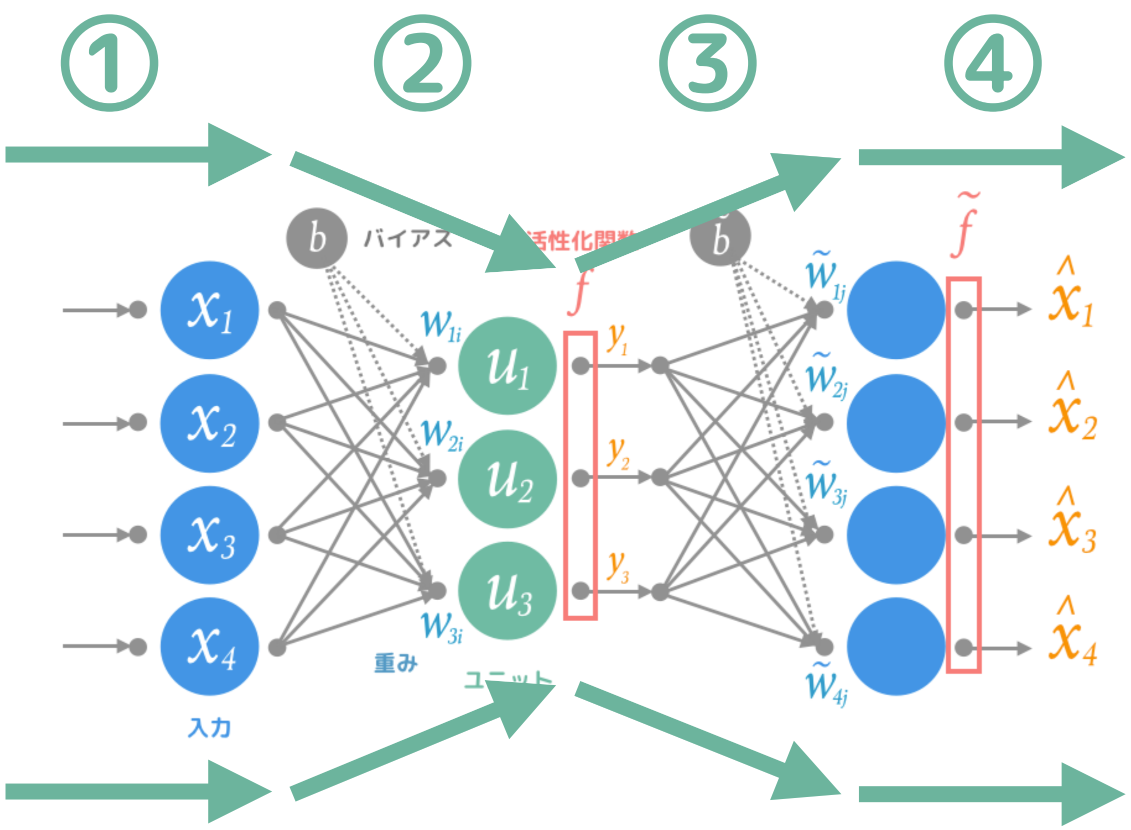


① 高次元データを入力

② 強制的に低次元空間に変換（符号化）

③ 元の高次元に戻す

直感的な独自解釈



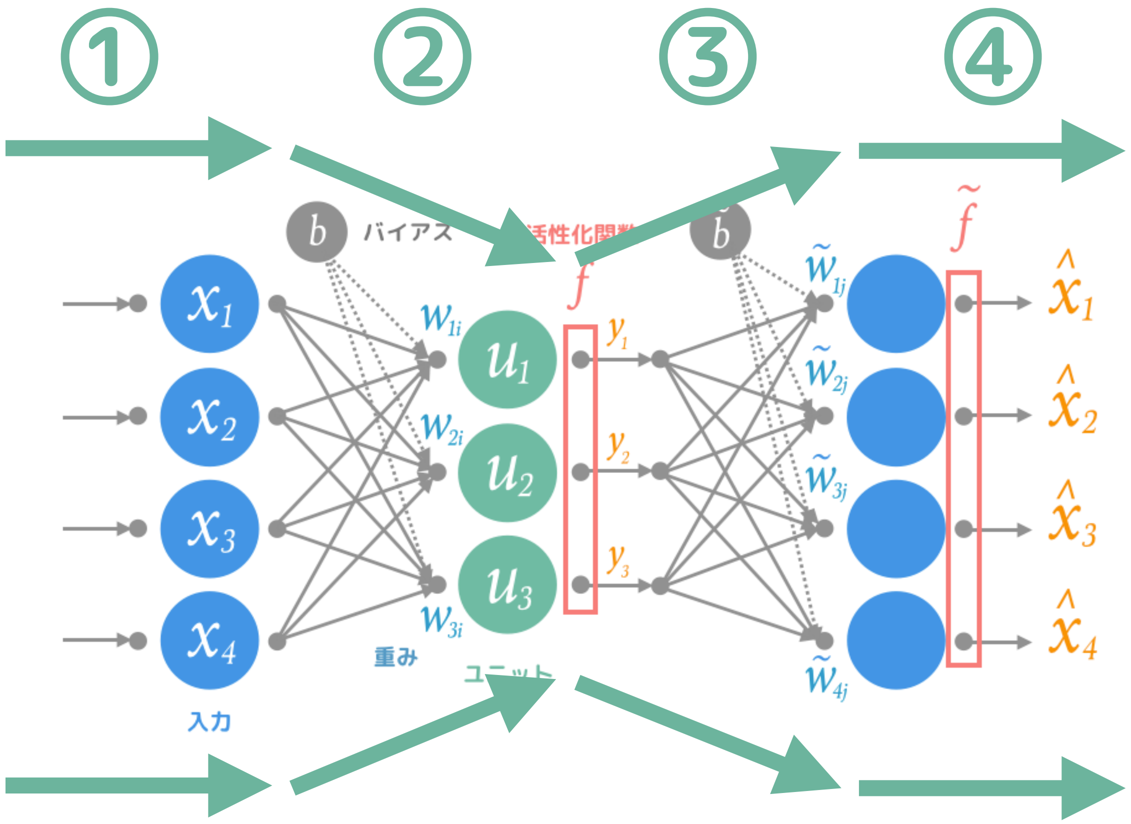
① 高次元データを入力

② 強制的に低次元空間に変換（符号化）

③ 元の高次元に戻す

④ 同じ入力に復号

直感的な独自解釈



① 高次元データを入力

② 強制的に低次元空間に変換（符号化）

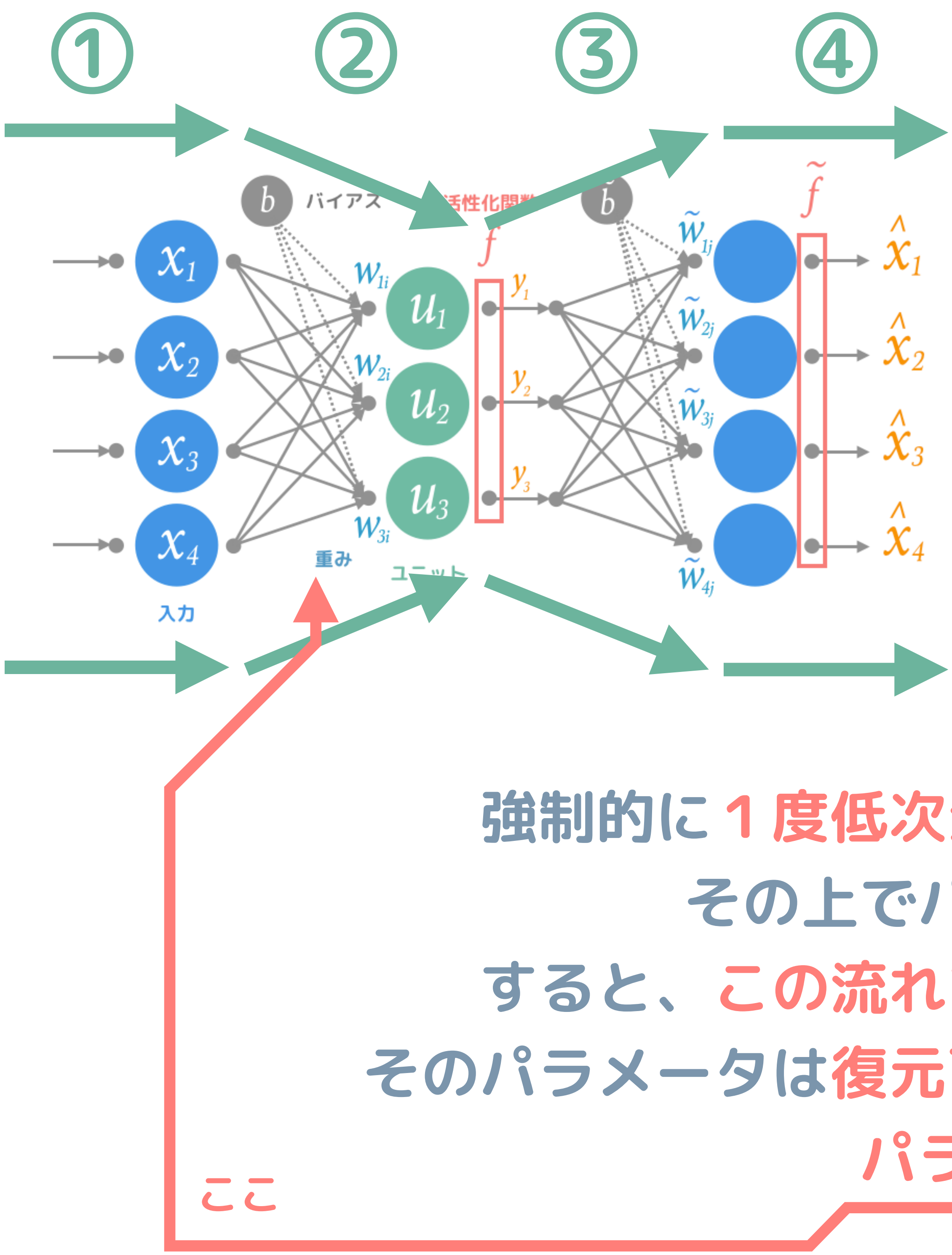
③ 元の高次元に戻す

④ 同じ入力に復号

強制的に１度低次元データに落とす層を通過させ、
その上でパラメータを学習させせる。
すると、この流れを満たすパラメータが得られる。
そのパラメータは復元可能な低次元データへ変換するための
パラメータを与える。

本のページ p**

直感的な独自解釈



① 高次元データを入力

② 強制的に低次元空間に変換（符号化）

③ 元の高次元に戻す

④ 同じ入力に復号

強制的に１度低次元データに落とす層を通過させ、
その上でパラメータを学習させる。
すると、この流れを満たすパラメータが得られる。
そのパラメータは復元可能な低次元データへ変換するための
パラメータを与える。

本のページ　p**

これは「主成分分析」と違うのか？

似たようなこと聞いたことあるけど…

はい、同じです

主成分分析との関係

主成分分析との関係

- 学習の結果得られる重み行列Wは、
  主成分分析で得られる結果と実質的に同じ（p61中段）
  - 重み行列Wは主成分分析で得られる固有ベクトルが並んだ形

主成分分析との関係

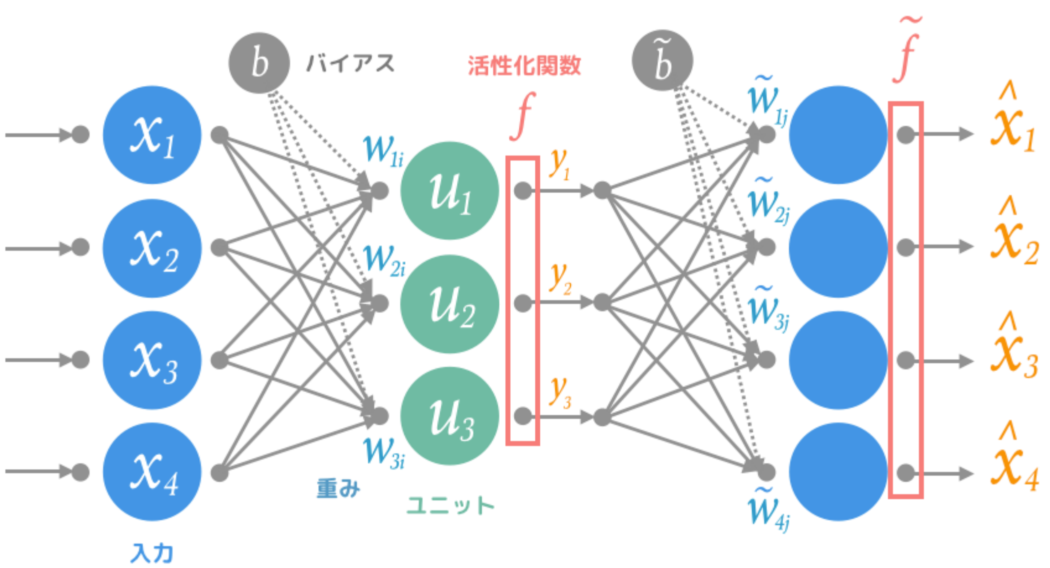
- 学習の結果得られる重み行列Wは、
  主成分分析で得られる結果と実質的に同じ（p61中段）
    - 重み行列Wは主成分分析で得られる固有ベクトルが並んだ形

- 中間ユニット数が入力ユニット数よりも少ない場合に限る！
    - 中間ユニットの数が多い場合は期待した結果が得られない

自己符号化器（まとめ）

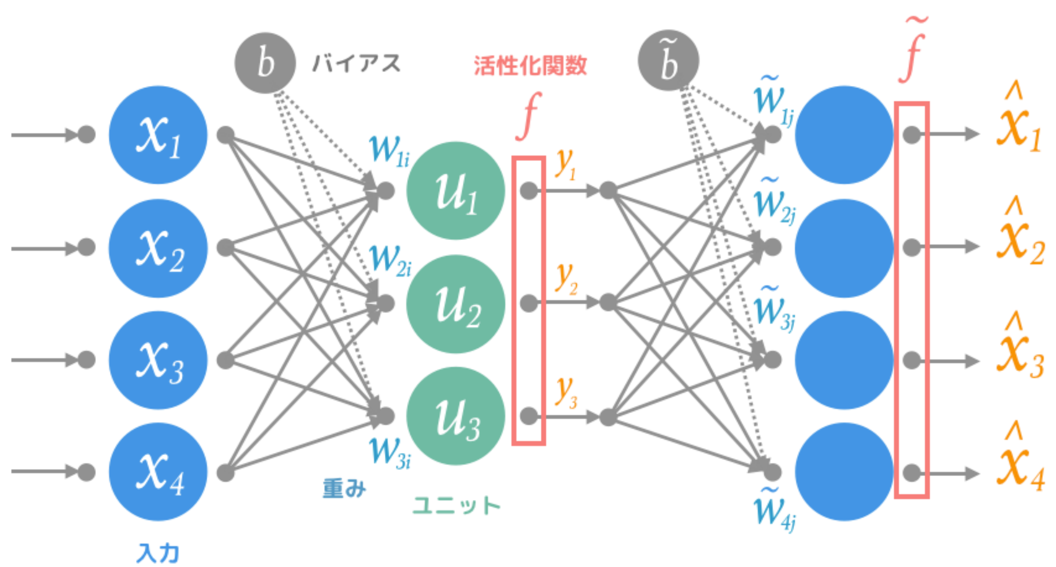
自己符号化器（まとめ）

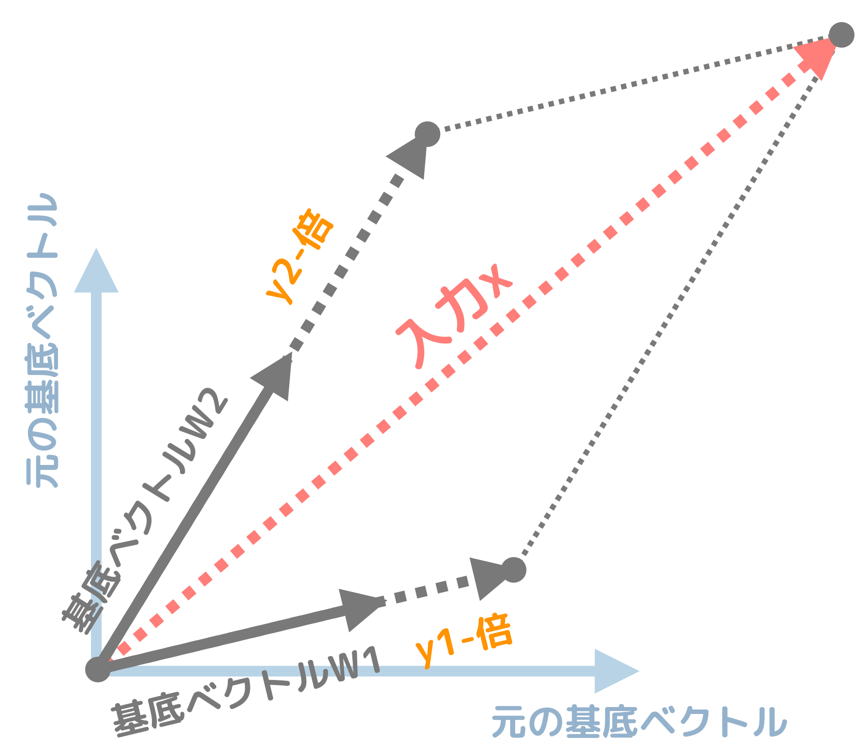
・自己符号化器の一つの機能 → **特徴を得る**



自己符号化器（まとめ）

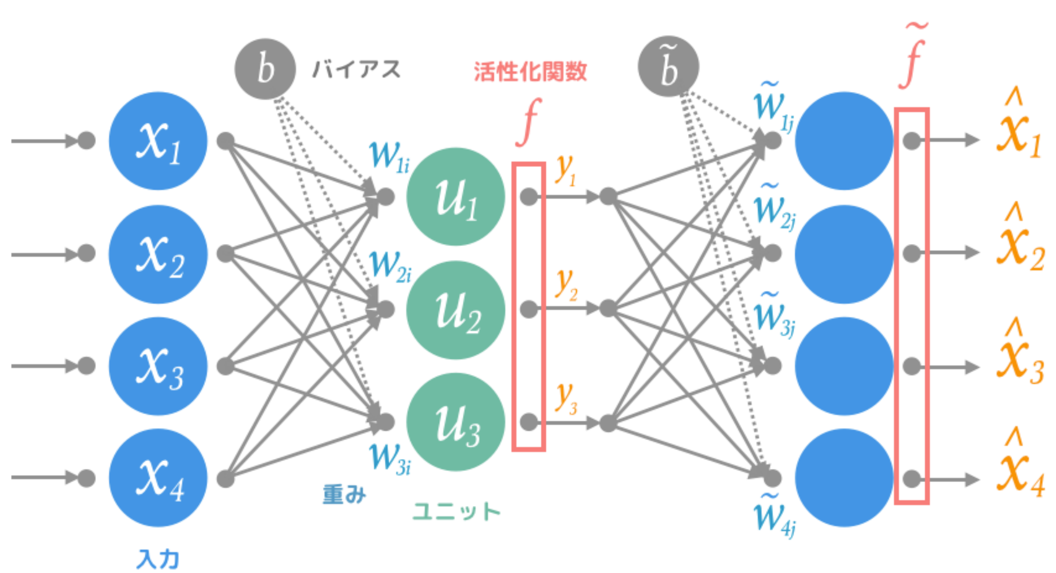
- 自己符号化器の一つの機能 → 特徴を得る
- 特徴とは、少ない次元でデータを表現できる空間の基底ベクトルのこと

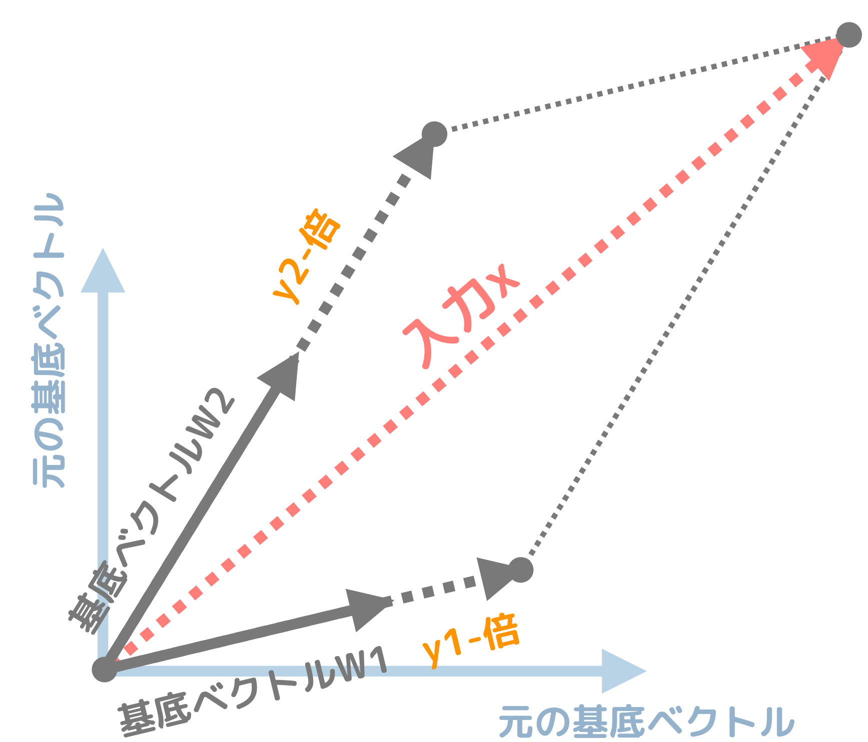




自己符号化器（まとめ）

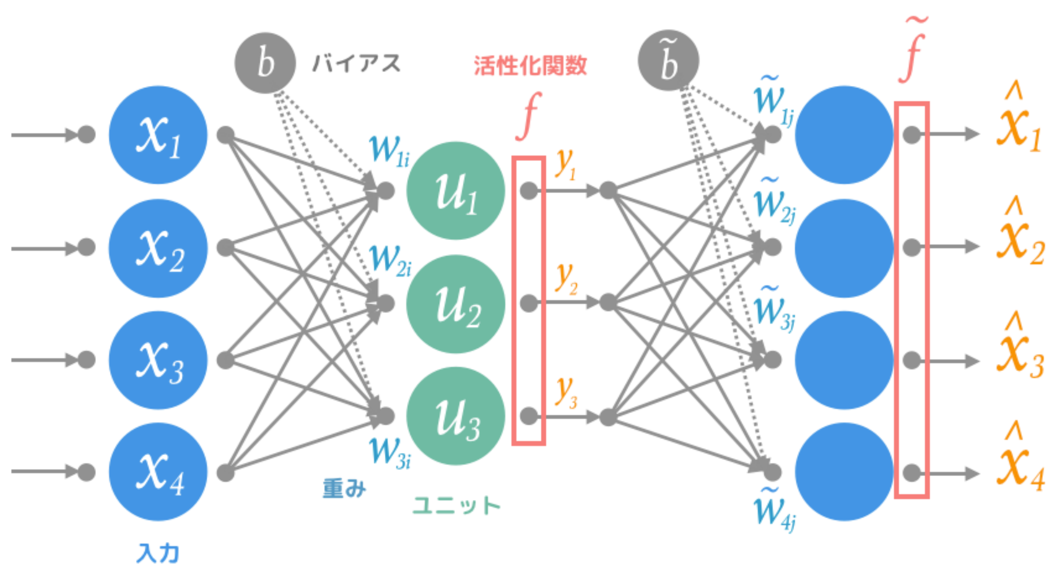
- 自己符号化器の一つの機能 → 特徴を得る
- 特徴とは、少ない次元でデータを表現できる空間の基底ベクトルのこと
- 強制的に次元が落とされる環境で学習することで次元を落とすためのパラメータ（特徴）が得られる

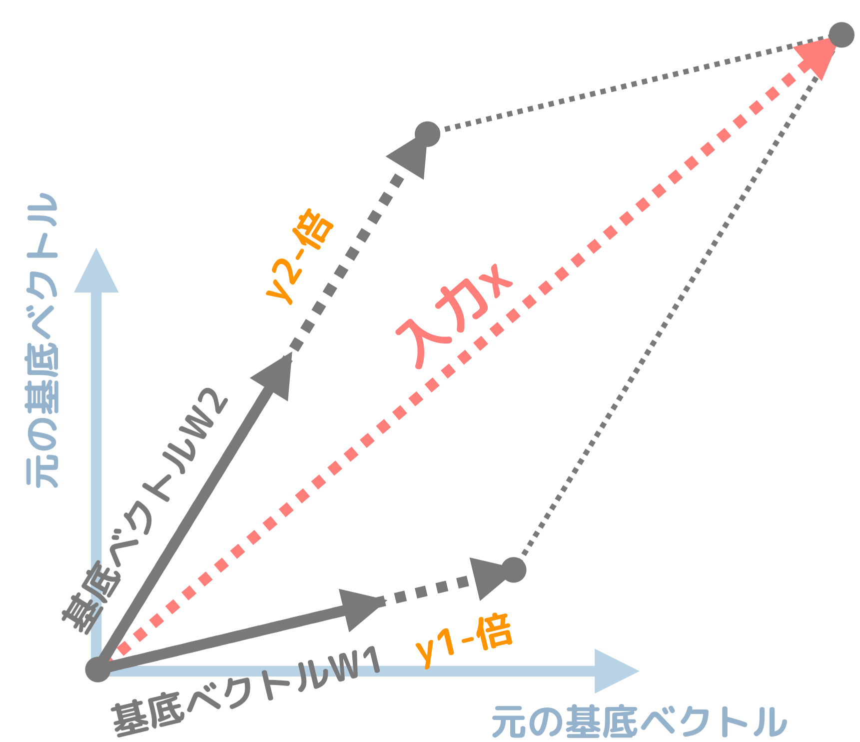




自己符号化器（まとめ）

- 自己符号化器の一つの機能 → 特徴を得る
- 特徴とは、少ない次元でデータを表現できる空間の基底ベクトルのこと
- 強制的に次元が落とされる環境で学習することで次元を落とすためのパラメータ（特徴）が得られる
- 実質的には主成分分析と同じ





本のページ p**

スパース正則化

もう少し自己符号化器をレベルアップさせせます

スパース正則化とは

スパース正則化とは

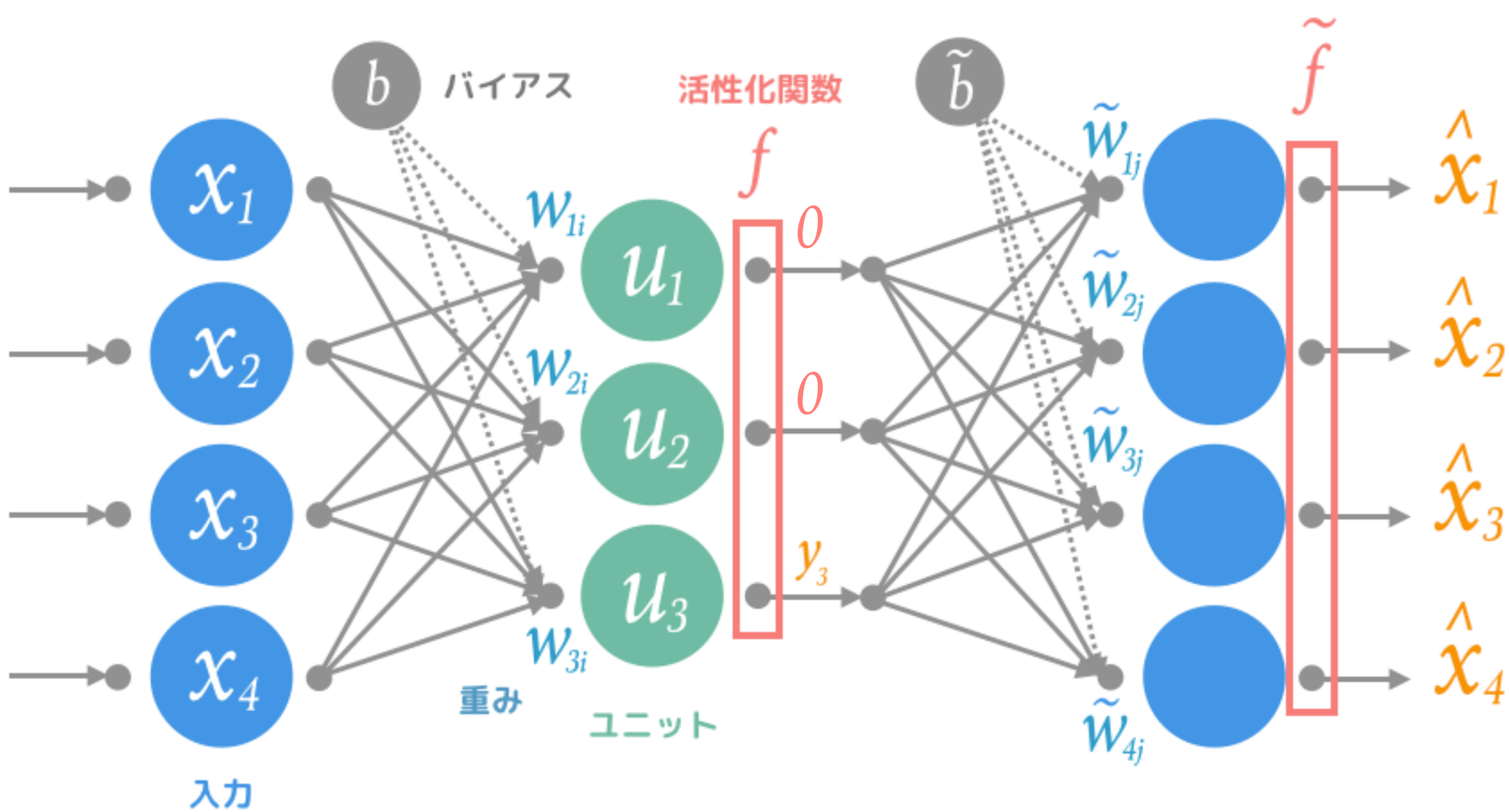
いくつかの中間ユニットの出力をゼロにする

そうなるように学習するようテコ入れする

スパース正則化とは

いくつかの中間ユニットの出力をゼロにする

そうなるように学習するようテコ入れする



何が利点？

何が利点？

符号化後の空間において多数の成分がゼロである利点

何が利点？

符号化後の空間において多数の成分がゼロである利点

・符号化後のデータを使えば計算が簡単
・符号化することでデータ圧縮が可能

何が利点？

符号化後の空間において多数の成分がゼロである利点

・符号化後のデータを使えば計算が簡単
・符号化することでデータ圧縮が可能

ニューラルネットに特化した利点

何が利点？

符号化後の空間において多数の成分がゼロである利点

- 符号化後のデータを使えば計算が簡単
- 符号化することでデータ圧縮が可能

ニューラルネットに特化した利点

- 中間層のユニット数が多くても大丈夫
  - 入力層のユニット数より多くても意味のあるデータが取れる
  - これを「過完備な表現」という

本のページ p62

どうやって学習するの？

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(\boldsymbol{w}) \equiv E(\boldsymbol{w}) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(\boldsymbol{w}) \equiv E(\boldsymbol{w}) + \boxed{\underline{\beta} \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

正則化の強さ

パラメータとして設定する

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(\mathbf{w}) \equiv E(\mathbf{w}) + \boxed{\underline{\beta} \sum_{j=1}^{D_y} \mathrm{KL}\left(\underline{\rho} \| \hat{\rho}_j\right)}$$

平均活性度の目標値

パラメータとして設定する

正則化の強さ

パラメータとして設定する

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(\boldsymbol{w}) \equiv E(\boldsymbol{w}) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

正則化の強さ

パラメータとして設定する

平均活性度の目標値

パラメータとして設定する

ユニット j の平均活性度

全訓練データの活性度の平均

$$\hat{\rho}_j = \frac{1}{N} \sum_{n=1}^{N} y_j(\boldsymbol{x}_n)$$

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(\boldsymbol{w}) \equiv E(\boldsymbol{w}) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

どうやって学習するの？

誤差関数に **正則化項** を加えて最小化する

$$\tilde{E}(w) \equiv E(w) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

学習手順

① 小さな ρ を設定する → 平均活性度をゼロに向かわせるため
② 中間層の各ユニットの平均活性度が ρ に近づく
③ その中で誤差 E が最小になるように w が決定される
④ 活性化するユニットはどれでも良い

正則化項について

$$\sum_{j=1}^{D_y} \underbrace{\mathrm{KL}\,(\,\rho \| \,\hat{\rho}_j)}_{\text{カルバック・ライブラー・ダイバージェンス}}$$

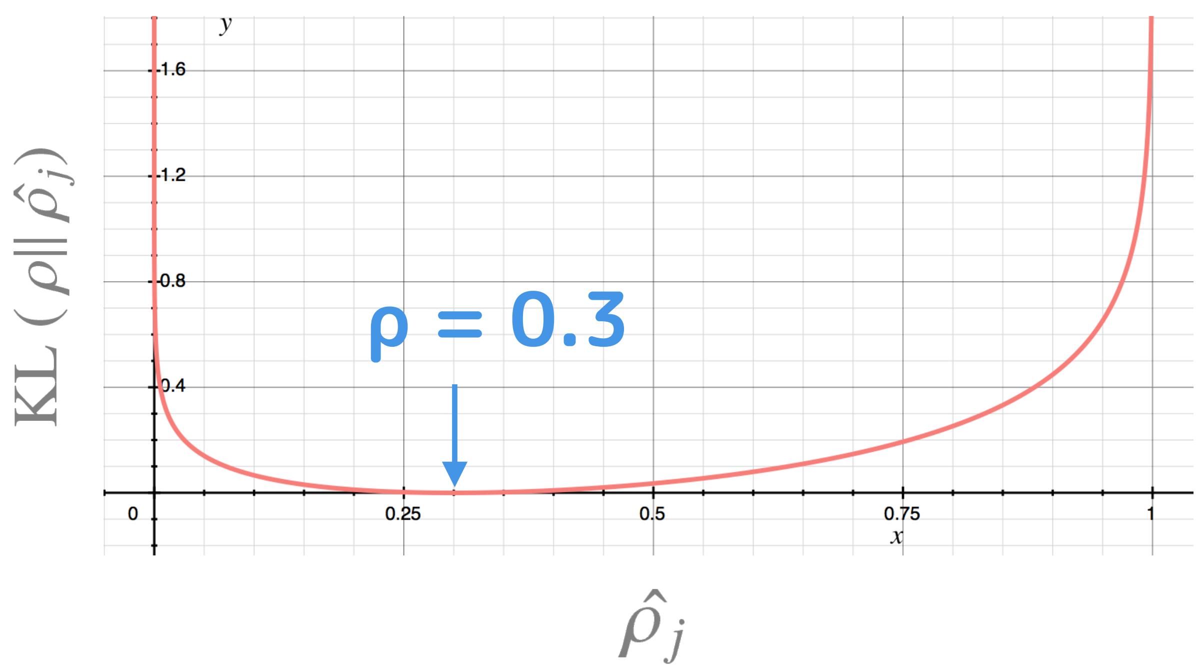
$$\mathrm{KL}\,(\,\rho \| \,\hat{\rho}_j) = \rho \log(\frac{\rho}{\hat{\rho}_j}) + (1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_j)}$$

正則化項について

$$\sum_{j=1}^{D_y} \underline{\mathrm{KL}(\rho \| \hat{\rho}_j)}$$

カルバック・ライブラー・ダイバージェンス

$$\mathrm{KL}(\rho \| \hat{\rho}_j) = \rho \log(\frac{\rho}{\hat{\rho}_j}) + (1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_j)}$$

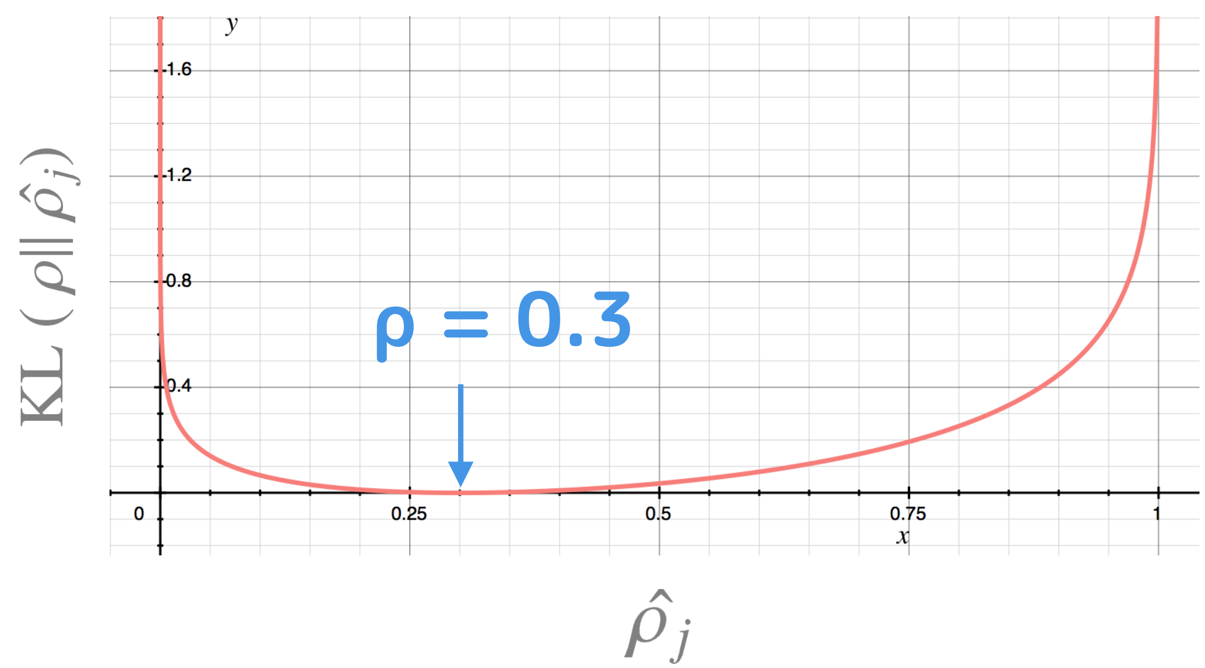


正則化項について

$$\sum_{j=1}^{D_y} \underline{\mathrm{KL}\,(\,\rho \| \,\hat{\rho}_j)}$$

カルバック・ライブラー・ダイバージェンス

$$\mathrm{KL}\,(\,\rho \| \,\hat{\rho}_j) = \rho \log(\frac{\rho}{\hat{\rho}_j}) + (1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_j)}$$



・目標値から遠いと値が大きい
・目標値に近いと値が小さい
・誤差関数を最小化していくので、KLの値が小さくなる方向に進む

↓

平均活性度 $\hat{\rho}$ が ρ に近づく

本のページ p64

スパース正則化の効果

「5.4.3 最適化」は逆伝播法の計算方法なので
詳細は本書に譲ります

本のページ p66

変更するパラメータ

$$\tilde{E}(\boldsymbol{w}) \equiv E(\boldsymbol{w}) + \beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j \right)$$

変更するパラメータ

$$\tilde{E}(w) \equiv E(w) + \underline{\beta} \sum_{j=1}^{D_y} \mathrm{KL}\,(\,\rho \| \,\hat{\rho}_j)$$

正則化の強さ

変更するパラメータ

$$\tilde{E}(w) \equiv E(w) + \underline{\beta} \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)$$

正則化の強さ

① β = 0.0 : 正則化効果なし

② β = 0.1 : 適度な正則化効果

③ β = 3.0 : 正則化効果強すぎ

これらについて見ていく

本のページ p66

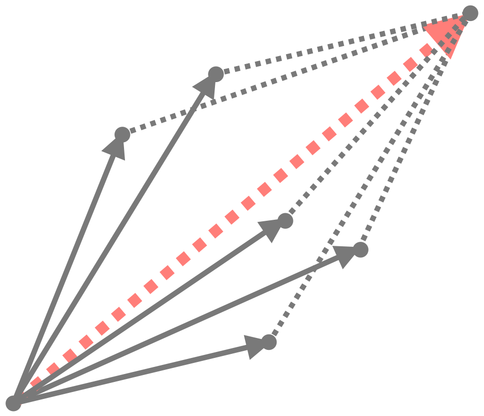
パラメータによる違い

p67 の 4枚の図を見ながらだとわかりやすいです

パラメータによる違い

p67 の 4枚の図を見ながらだとわかりやすいです

① β = 0.0 : 正則化効果なし



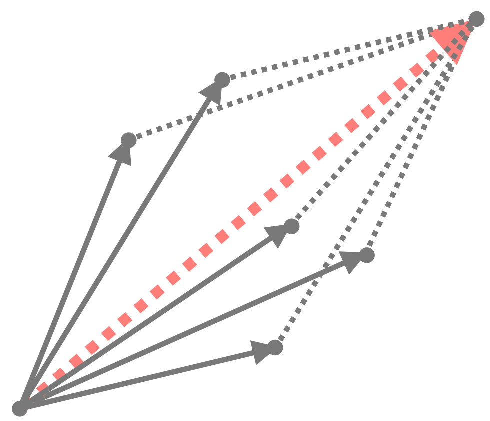
・全ての基底ベクトルWを無理やり用いて符号化する
  - 圧縮や計算効率などの恩恵を受けられない

パラメータによる違い

p67 の 4枚の図を見ながらだとわかりやすいです

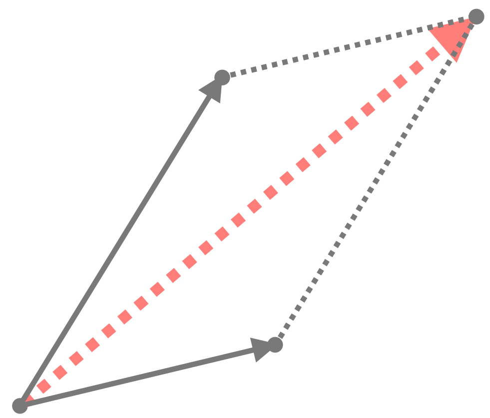
① β = 0.0：正則化効果なし

・全ての基底ベクトルWを無理やり用いて符号化する
- 圧縮や計算効率などの恩恵を受けられない



② β = 0.1：適度な正則化効果

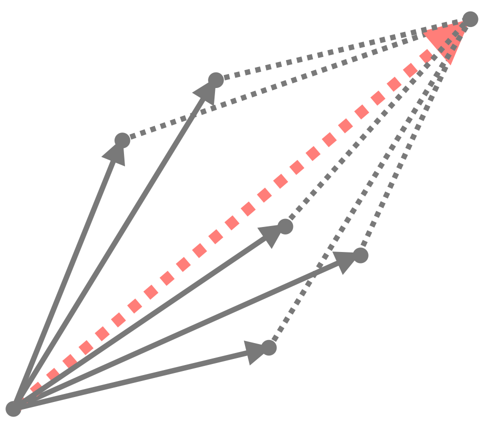
・基底ベクトルそれぞれが役割を果たし、
いくつかだけを用いて符号化する
- 低次元データを上手く生成できる



パラメータによる違い

p67 の 4枚の図を見ながらだとわかりやすいです

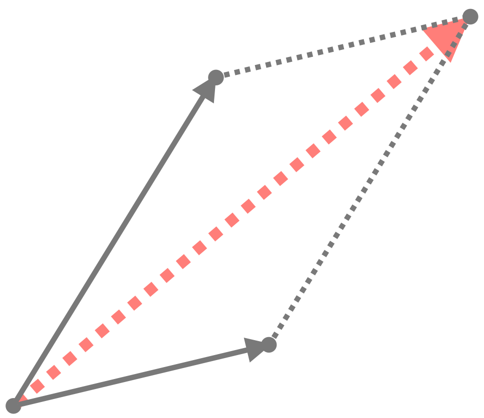
① β = 0.0：正則化効果なし



・全ての基底ベクトルWを無理やり用いて符号化する

- 圧縮や計算効率などの恩恵を受けられない

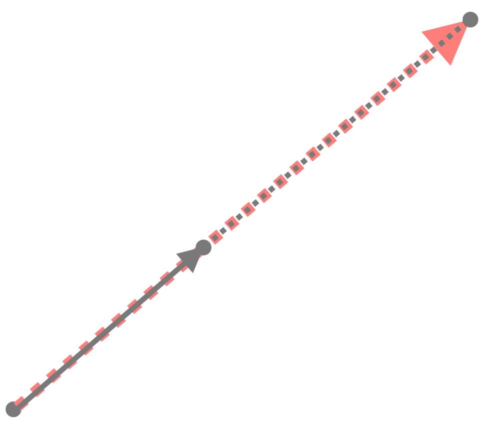
② β = 0.1：適度な正則化効果



・基底ベクトルそれぞれが役割を果たし、
いくつかだけを用いて符号化する

- 低次元データを上手く生成できる

③ β = 3.0：正則化効果強すぎ



・それぞれの基底ベクトルが「ただそれだけ」で
データを表現しようとする

- 微妙な変化などを吸収できない

スパース正則化（まとめ）

スパース正則化（まとめ）

・利点は計算量の削減やデータ圧縮

スパース正則化（まとめ）

- 利点は計算量の削減やデータ圧縮
- 中間層の出力をいくつかゼロにする

スパース正則化（まとめ）

- 利点は計算量の削減やデータ圧縮
- 中間層の出力をいくつかゼロにする
- 誤差関数に正則化項を加えて最小化する

$$\tilde{E}(w) \equiv E(w) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

スパース正則化（まとめ）

- 利点は計算量の削減やデータ圧縮
- 中間層の出力をいくつかゼロにする
- 誤差関数に正則化項を加えて最小化する
- 正則化の強さパラメータを変更することで結果が変わる

$$\tilde{E}(w) \equiv E(w) + \boxed{\beta \sum_{j=1}^{D_y} \mathrm{KL}\left(\rho \| \hat{\rho}_j\right)}$$

本のページ p**

白色化

より学習しやすいデータへ変換

白色化とは

白色化とは

データの偏りを除去する

良い特徴を学習できるようになる

白色化とは

データの偏りを除去する

良い特徴を学習できるようになる

↓

訓練データの成分間の相関を無くす

白色化とは

データの偏りを除去する

良い特徴を学習できるようになる

↓

訓練データの成分間の相関を無くす

↓

共分散行列を対角化する！

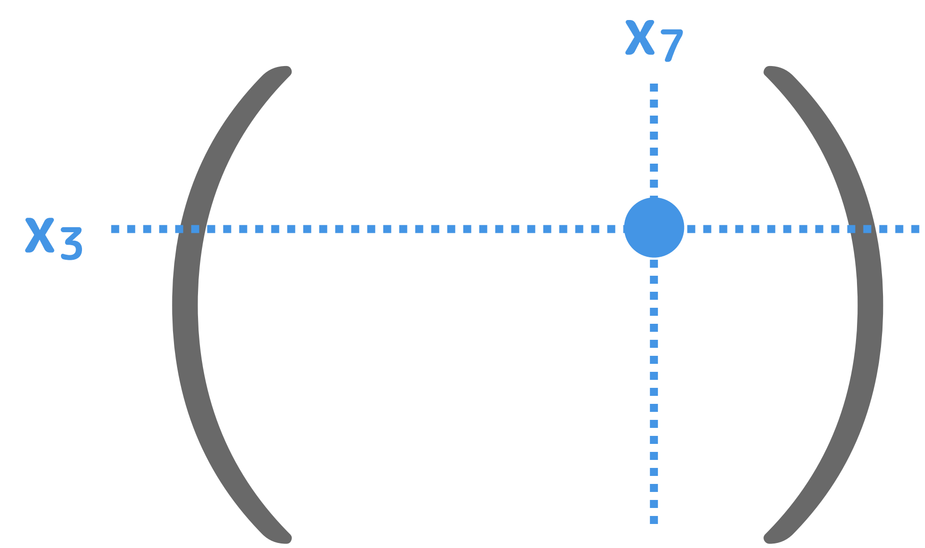
$$\boldsymbol{\Phi}_X \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$$

共分散行列の対角化

$$\boldsymbol{\Phi}_X \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$$

共分散行列の対角化

$$\mathbf{\Phi}_X \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$$



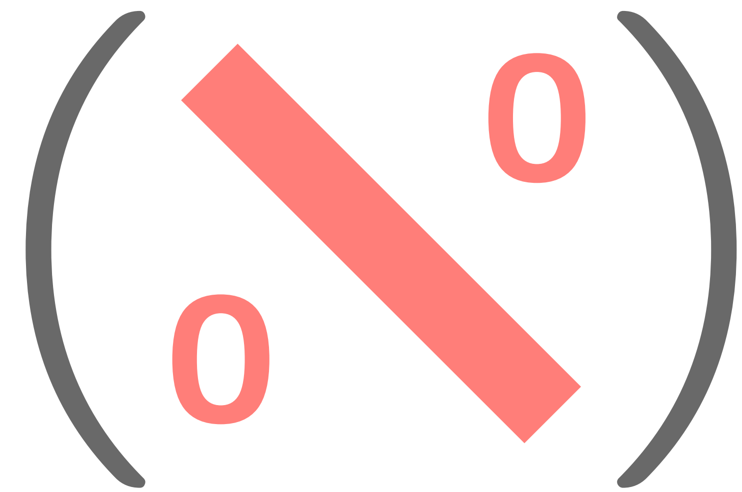
x_3とx_7に相関があれば
ここは非ゼロ

共分散行列の対角化

$$\Phi_X \equiv \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N} \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} \qquad \left(\qquad \right)$$

共分散行列の対角化

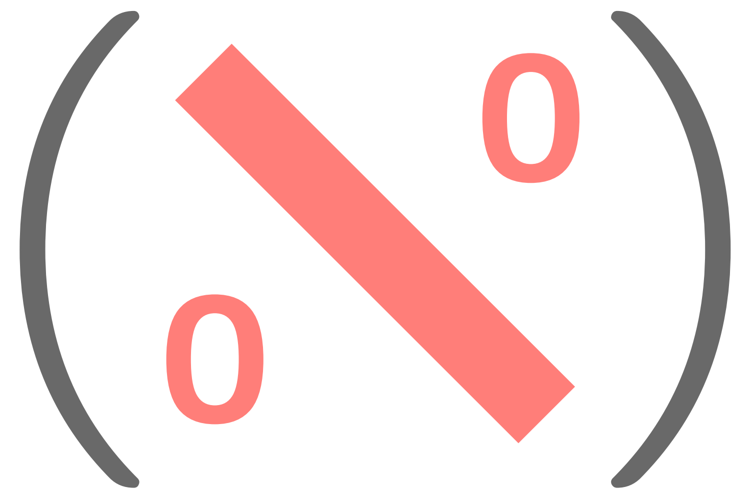
$$\Phi_X \equiv \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N} \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$$



各成分での相関がない
→ 対角成分以外がゼロ

共分散行列の対角化

$$\boldsymbol{\Phi}_X \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}} = \frac{1}{N}XX^{\mathrm{T}}$$



各成分での相関がない
→ 対角成分以外がゼロ

アプローチとしては…

訓練データxにある行列Pをかけると

共分散行列が対角行列になるとする

その行列Pをみつける

１ステップずつ見ていきます

白色化に向けて

白色化に向けて

① ある行列Pを用いて各訓練データを変換

$$\boldsymbol{u}_n = \boldsymbol{P}\boldsymbol{x}_n$$

nは訓練データ番号 (n = 1 … N)

白色化に向けて

① ある行列Pを用いて各訓練データを変換

$$\boldsymbol{u}_n = \boldsymbol{P}\boldsymbol{x}_n$$

nは訓練データ番号 (n = 1 … N)

② uの共分散行列を考える

$$\boldsymbol{\Phi}_U \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{u}_n\boldsymbol{u}_n^{\mathrm{T}} = \frac{1}{N}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}$$

これが対角行列になるように行列Pを決める！

白色化に向けて

白色化に向けて

③ 目標対角行列を単位行列とする

$$\boldsymbol{\Phi}_U \equiv \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{u}_n\boldsymbol{u}_n^{\mathrm{T}} = \frac{1}{N}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} \longrightarrow \boldsymbol{\Phi}_U = \boldsymbol{I}$$

白色化に向けて

③ 目標対角行列を単位行列とする

$$\Phi_U \equiv \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{u}_n \boldsymbol{u}_n^{\mathrm{T}} = \frac{1}{N} \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} \longrightarrow \Phi_U = I$$

④ Pの満たすべき式が得られる

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P} = \Phi_X^{-1}$$

$\boldsymbol{U} = \boldsymbol{P}\boldsymbol{X}$ を代入して得られる

白色化に向けて

白色化に向けて

⑤ 右辺のΦ_Xについて考える

$$P^{\mathrm{T}} P = \boxed{\Phi_X^{-1}}$$

これについて考える

白色化に向けて

⑤ 右辺のΦ_Xについて考える

$$P^{\mathrm{T}}P = \boxed{\Phi_X^{-1}}$$

これについて考える

⑥ Φ_Xを固有ベクトルと固有値を用いて分解

$$\Phi_X = \underbrace{E}_{\text{固有ベクトルを列ベクトルとする行列}}\underbrace{D}_{\text{固有値を対角に持つ対角行列}}E^{\mathrm{T}}$$

固有値を対角に持つ対角行列

固有ベクトルを列ベクトルとする行列

白色化に向けて

白色化に向けて

⑦ Φ_Xの逆行列を考える

$$\Phi_X = EDE^{\mathrm{T}}$$

$$\downarrow$$

$$\boxed{\Phi_X^{-1} = ED^{-1}E^{\mathrm{T}}}$$

$$EE^{\mathrm{T}} = E^{\mathrm{T}}E = I$$

直交行列の特性を利用して算出

白色化に向けて

⑦ Φ_Xの逆行列を考える

$$\Phi_X = EDE^{\mathrm{T}}$$

$$\downarrow$$

$$\boxed{\Phi_X^{-1} = ED^{-1}E^{\mathrm{T}}}$$

$$EE^{\mathrm{T}} = E^{\mathrm{T}}E = I$$

直交行列の特性を利用して算出

⑧ Pが満たすべき式に代入しPを求める

$$\left.\begin{aligned} P^{\mathrm{T}}P &= \Phi_X^{-1} \\ \Phi_X^{-1} &= ED^{-1}E^{\mathrm{T}} \end{aligned}\right\} \quad P = QD^{-1/2}E^{\mathrm{T}}$$

Q：Pと同サイズの直交行列

$D^{-1/2}$：対角成分を-1/2乗した対角行列

白色化に向けて

白色化に向けて

⑨ 行列Pを再考

$$P = QD^{-1/2}E^{\mathrm{T}}$$

- 行列Pを訓練データx_nに作用させ、白色化データを得て、それを学習に使用
  - 共分散行列が対角行列になるデータが得られる
- 行列Qの自由度の分、行列Pの解がある
  - Q=I だったら、それもPの解のひとつ

白色化に向けて

⑨ 行列Pを再考

$$P = QD^{-1/2}E^{\mathrm{T}}$$

- 行列Pを訓練データx_nに作用させ、白色化データを得て、それを学習に使用
  - 共分散行列が対角行列になるデータが得られる
- 行列Qの自由度の分、行列Pの解がある
  - Q=I だったら、それもPの解のひとつ

共分散行列の固有ベクトルを使用することから

PCA白色化

（PCA: 主成分分析）

という

白色化に向けて

白色化に向けて

⑩ 行列Pを対称行列に制限する

$$P_{ZCA} = ED^{-1/2}E^{\mathrm{T}}$$

- Q=E で得られる
  - こうするとP=P^{T}となる（対称行列）

白色化に向けて

⑩ 行列Pを対称行列に制限する

$$P_{ZCA} = ED^{-1/2}E^{\mathrm{T}}$$

・Q=E で得られる

- こうするとP=P$^{\mathrm{T}}$となる（対称行列）

対称であること（ゼロ位相である）から

ZCA白色化

（ZCA: ゼロ位相）

という

白色化に向けて

白色化に向けて

⑪ 特定の成分の分散がとても小さい時

$D^{-1/2}$の計算が問題になる時がある

$$P_{ZCA} = E(D + \underline{\varepsilon} I)^{-1/2} E^{\mathrm{T}}$$

εを導入して解決！

例えば 10^-6 などを与える

白色化に向けて

白色化に向けて

⑫ PCAとZCAの特徴

PCA白色化　高周波成分が強調され、元のデータの構造は見られない

ZCA白色化　境界が強調され、エッジ部分が残っている

白色化に向けて

⑫ PCAとZCAの特徴

PCA白色化　高周波成分が強調され、元のデータの構造は見られない

ZCA白色化　境界が強調され、エッジ部分が残っている

白色化の手法によって様々

上手く学習できる場合とできない場合があり
いろいろ試してみないとわからない

※PCAとZCAの詳細な違いは本書に譲ります

本のページ　p70

白色化（まとめ）

白色化（まとめ）

- 訓練データに偏りがあると上手く学習できないので、訓練データの各要素の分散をゼロにする

白色化（まとめ）

- 訓練データに偏りがあると上手く学習できないので、訓練データの各要素の分散をゼロにする
- 共分散行列が対角行列になるように行列Pで変換する

$$P = QD^{-1/2}E^{\mathrm{T}}$$

白色化（まとめ）

- 訓練データに偏りがあると上手く学習できないので、訓練データの各要素の分散をゼロにする
- 共分散行列が対角行列になるように行列Pで変換する
- 対角化の方法によって白色化の結果も変わる

$$P = QD^{-1/2}E^{\mathrm{T}}$$

ディープネットの事前学習

より学習しやすい初期値を獲得

自己符号化器のおいしい部分

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

② ディープネットの事前学習

・上手く学習できるような初期値を得る

自己符号化器のおいしい部分

① データを良く表す「特徴」を獲得

・訓練データ群からデータの基礎となる「特徴」を得る

② ディープネットの事前学習

・上手く学習できるような初期値を得る

次はここに注目！

ディープネットの事前学習

ディープネットの事前学習

多層の順伝播型ネットワークは学習が難しい

ディープネットの事前学習

多層の順伝播型ネットワークは学習が難しい

↓

初期値を上手く選定 すれば学習がうまくいく！

ディープネットの事前学習

多層の順伝播型ネットワークは学習が難しい

↓

初期値を上手く選定すれば学習がうまくいく！

↓

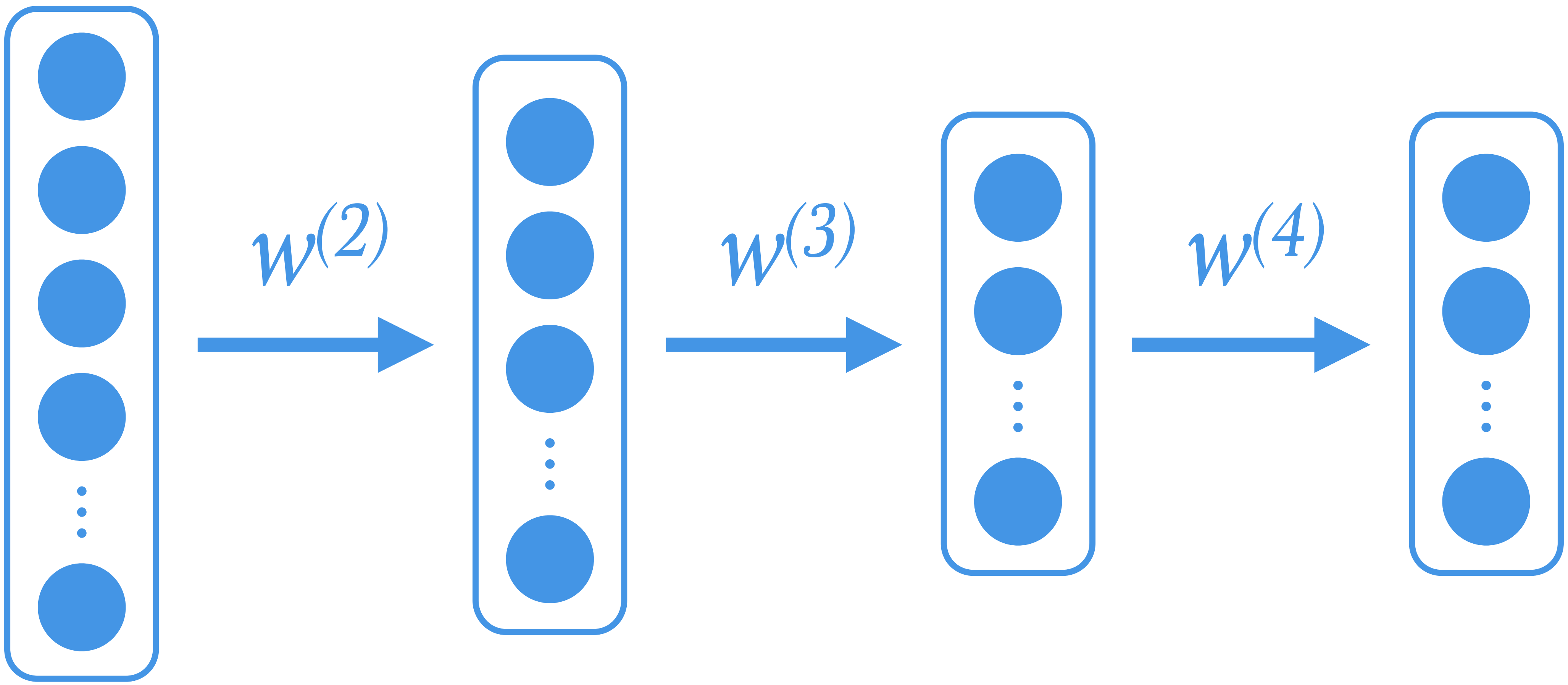
自己符号化器を用いて初期値を決定する

これが基本的な方法

自己符号化器を用いた事前学習

あるディープネットの例 → 何らかの学習を行いたい

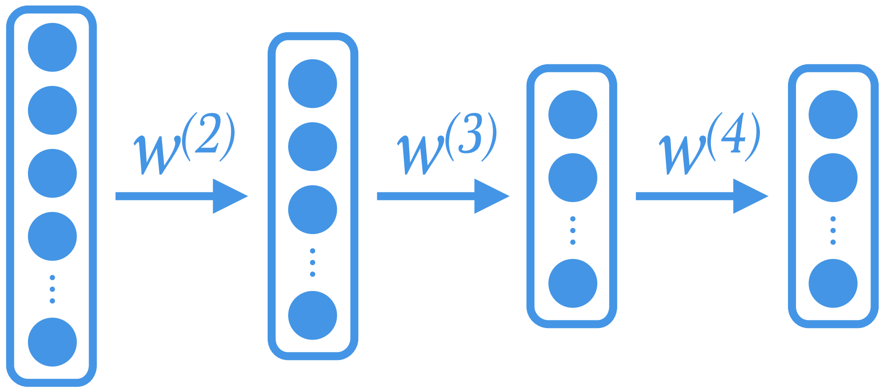
（画像判別器など）



この多層ディープネットの**各初期値** $w^{(2)}, w^{(3)}, w^{(4)}$ を決定する

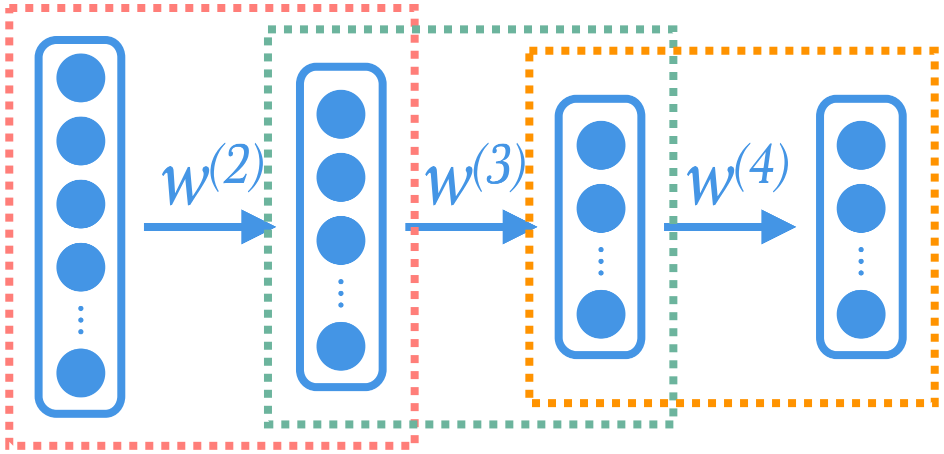
自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



自己符号化器を用いた事前学習

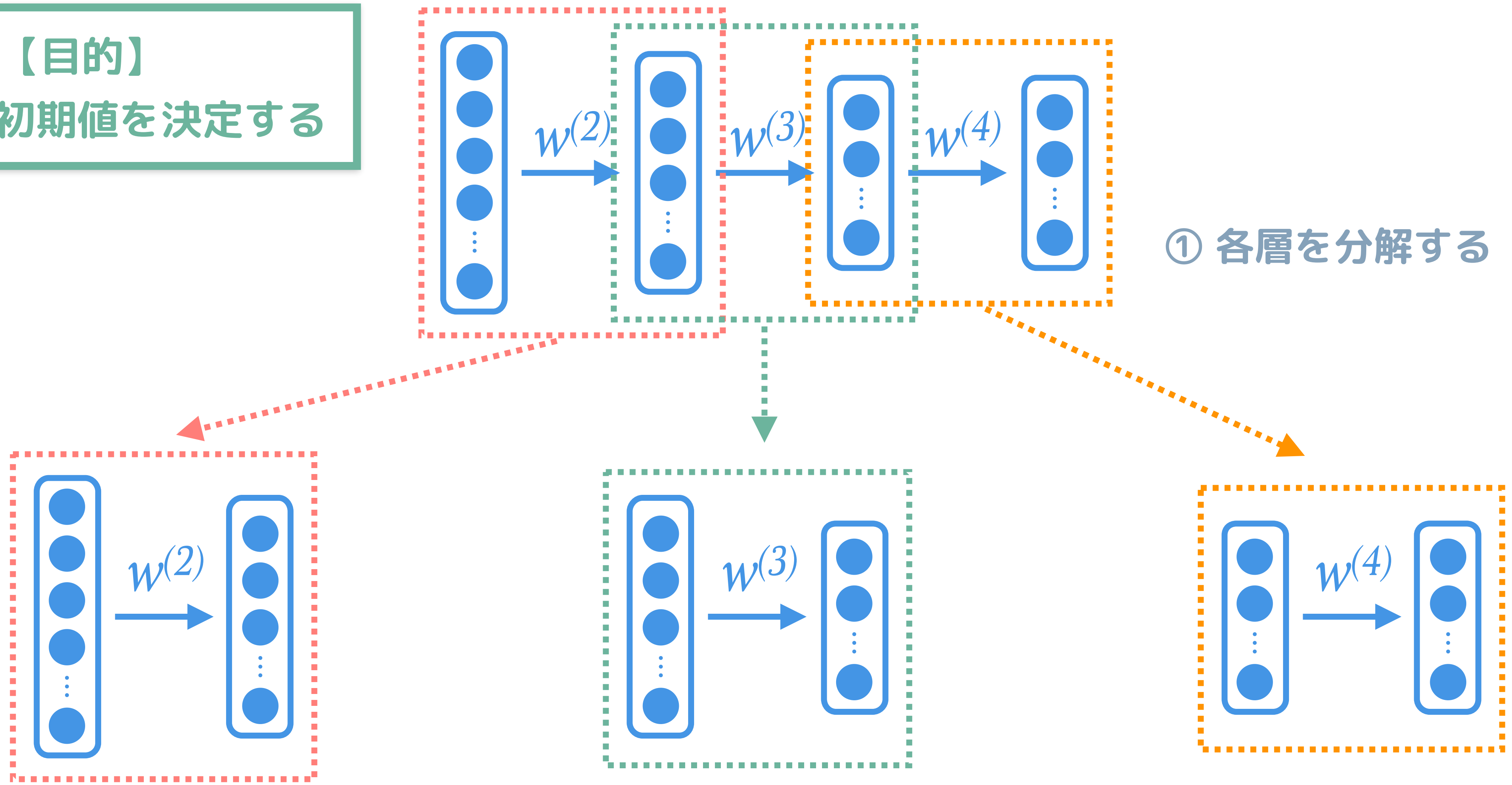
【目的】
重みwの初期値を決定する



① 各層を分解する

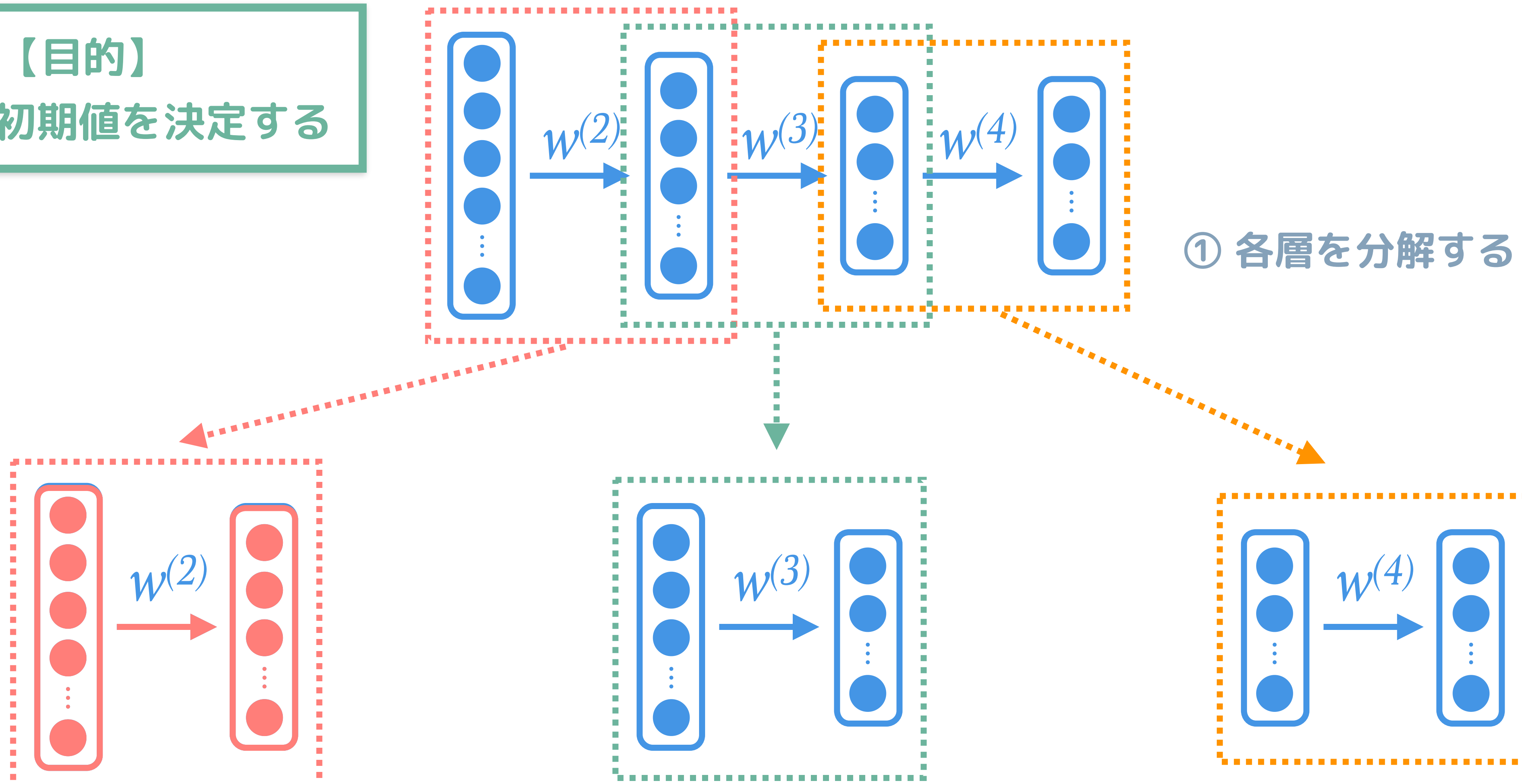
自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



自己符号化器を用いた事前学習

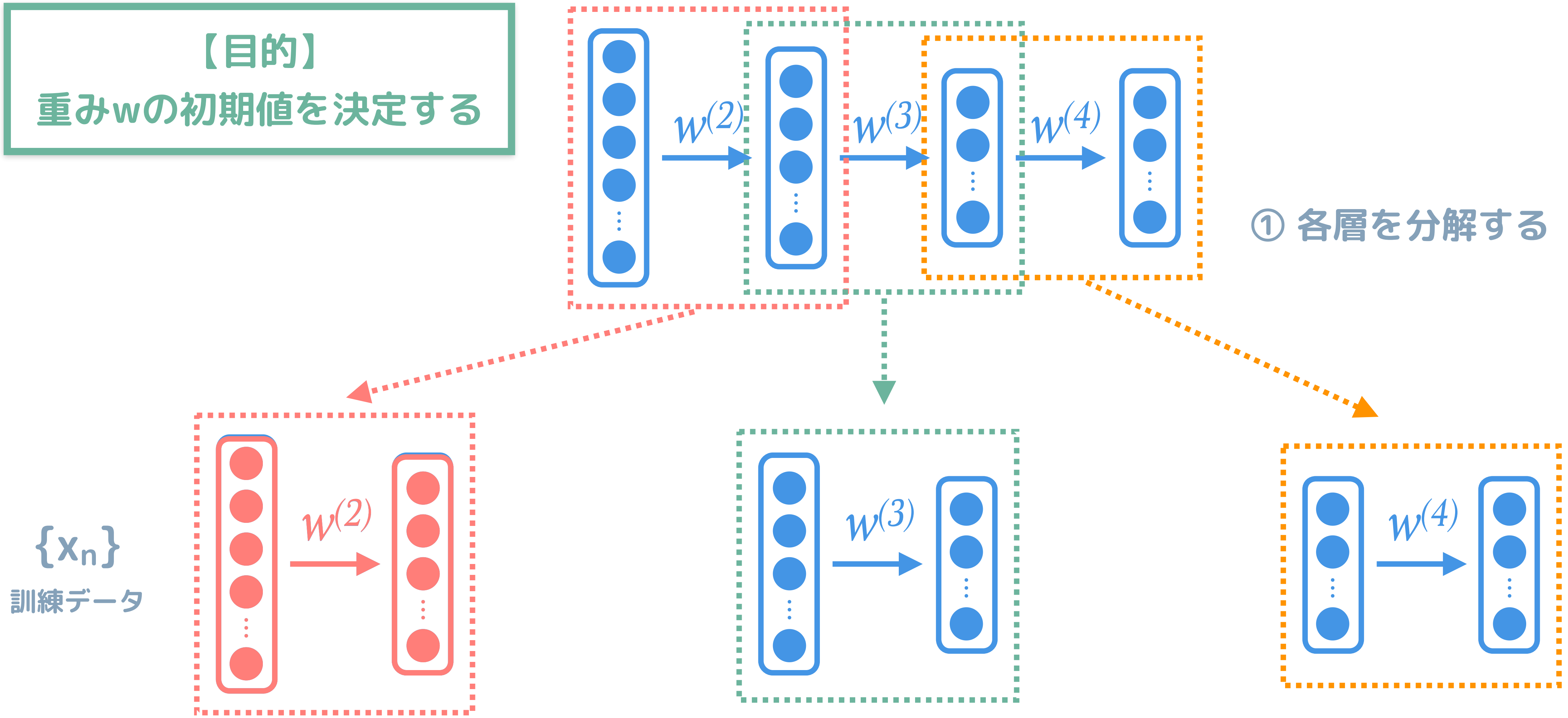
【目的】
重みwの初期値を決定する



② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



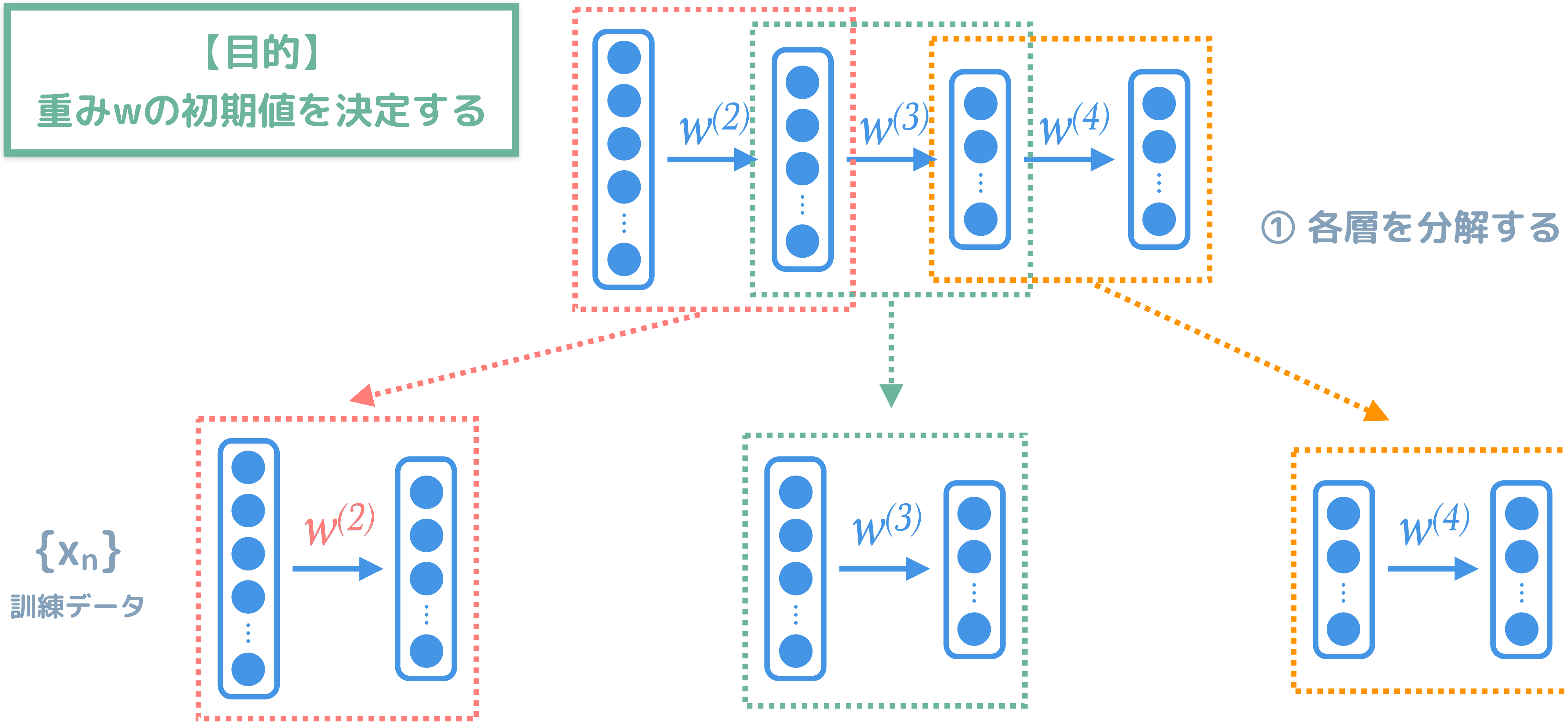
① 各層を分解する

② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

学習済み$w^{(2)}$を初期値に！

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する

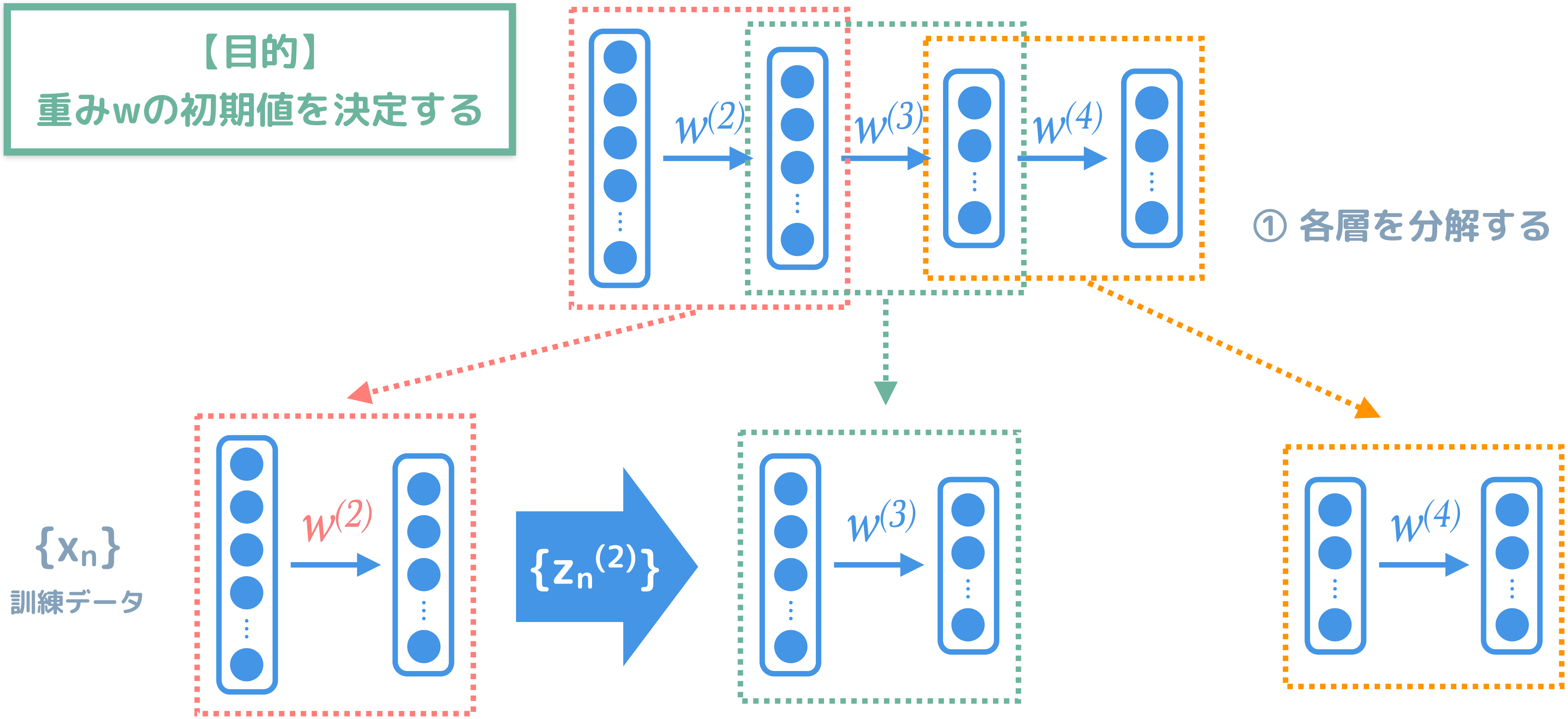


② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

学習済み$w^{(2)}$を初期値に！

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



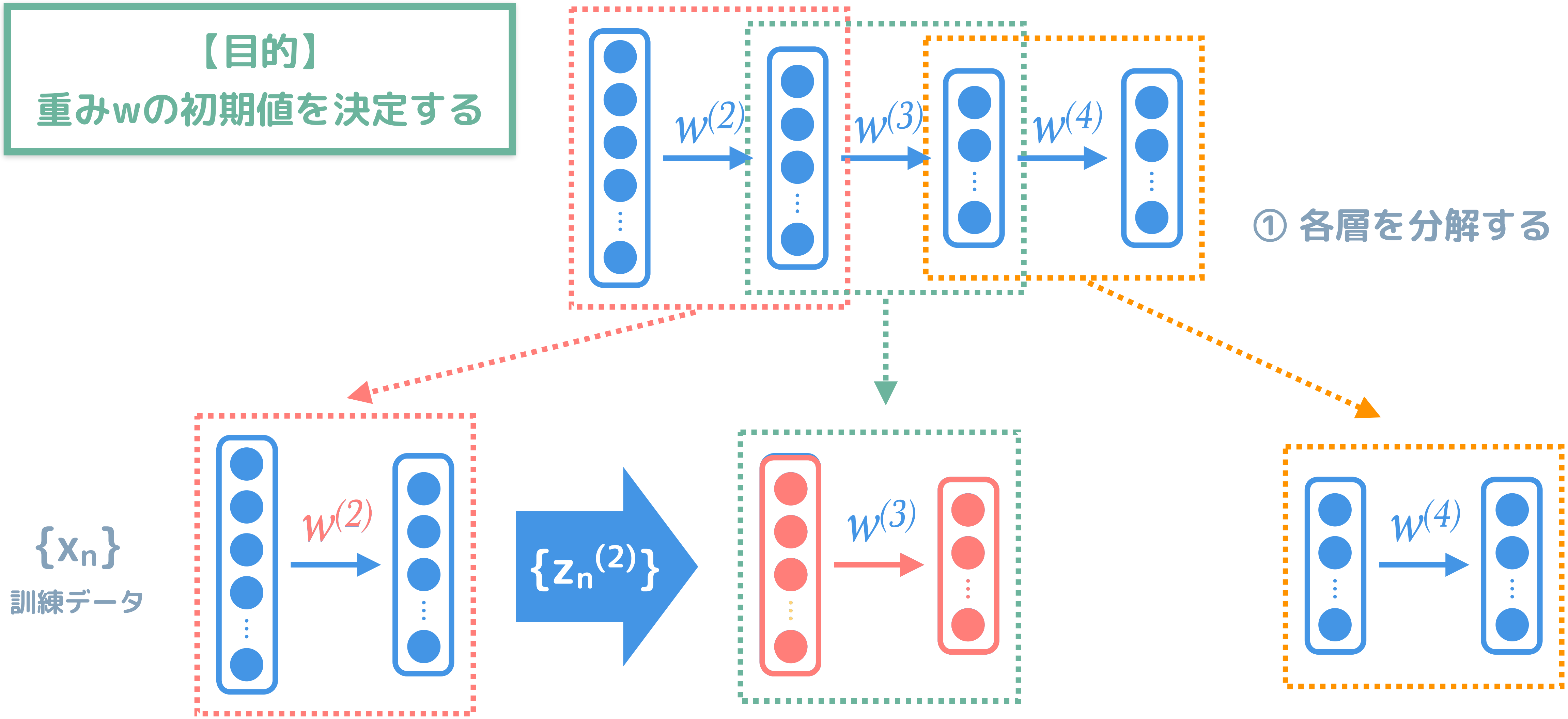
② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

③ 学習した$w^{(2)}$をセットし
訓練データ$\{x_n\}$を通し
出力$\{z_n^{(2)}\}$を得る

学習済み$w^{(2)}$を初期値に！

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

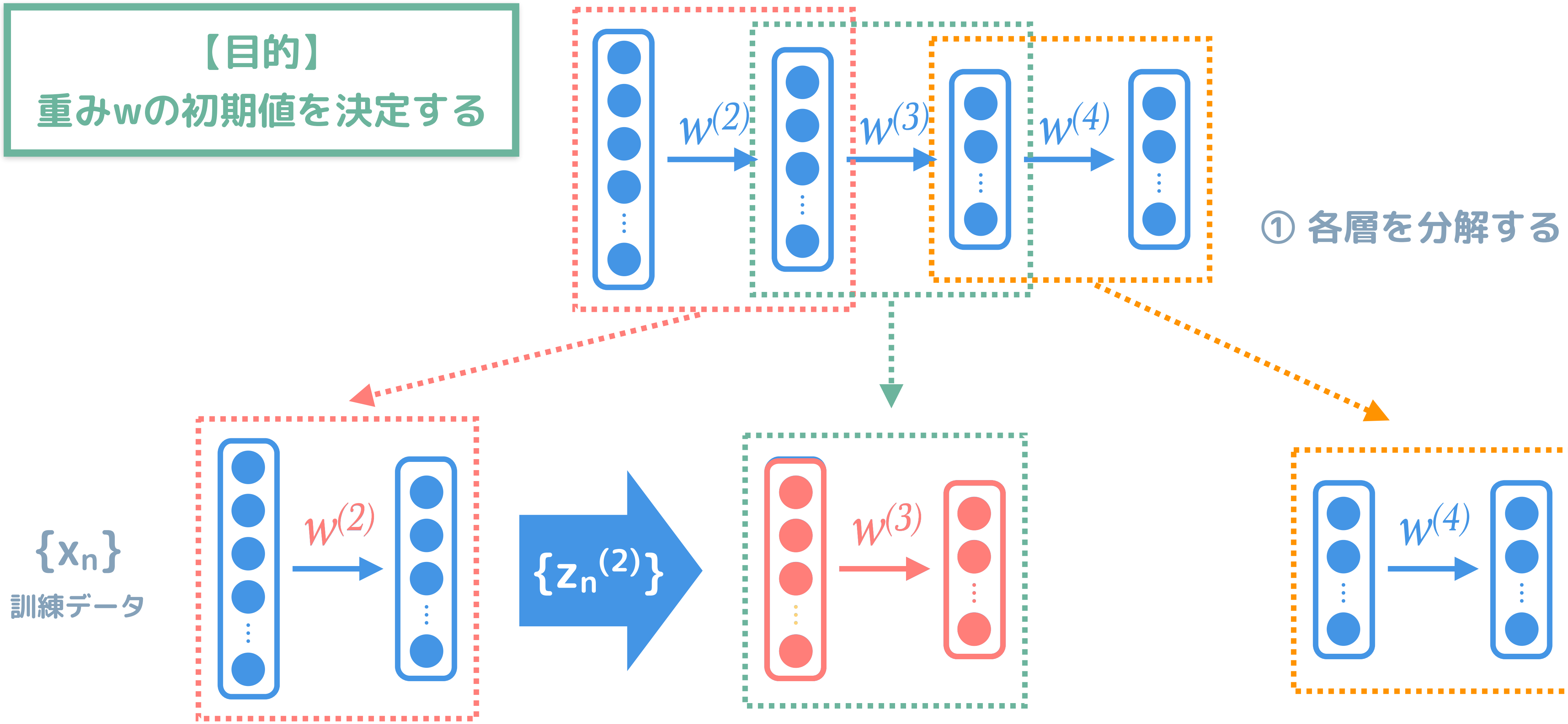
③ 学習した$w^{(2)}$をセットし
訓練データ$\{x_n\}$を通し
出力$\{z_n^{(2)}\}$を得る

④ 自己符号化器を構成し
直前の$\{z_n^{(2)}\}$を訓練データとし
重み$w^{(3)}$を学習する

学習済み$w^{(2)}$を初期値に！

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

③ 学習した$w^{(2)}$をセットし
訓練データ$\{x_n\}$を通し
出力$\{z_n^{(2)}\}$を得る

④ 自己符号化器を構成し
直前の$\{z_n^{(2)}\}$を訓練データとし
重み$w^{(3)}$を学習する

学習済み$w^{(2)}$を初期値に！

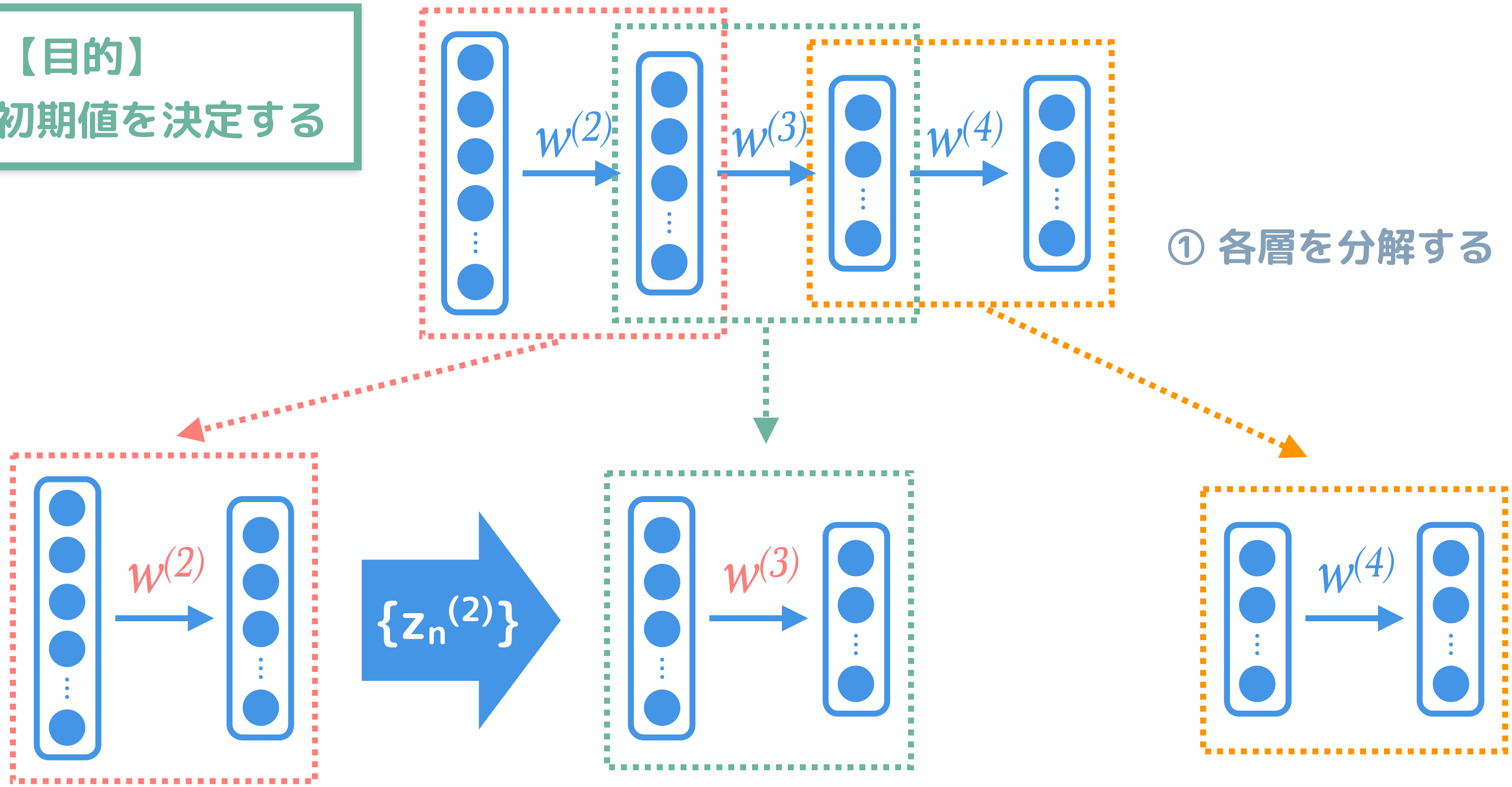
学習済み$w^{(3)}$を初期値に！

本のページ p73

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する

$\{x_n\}$
訓練データ



② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

③ 学習した$w^{(2)}$をセットし
訓練データ$\{x_n\}$を通し
出力$\{z_n^{(2)}\}$を得る

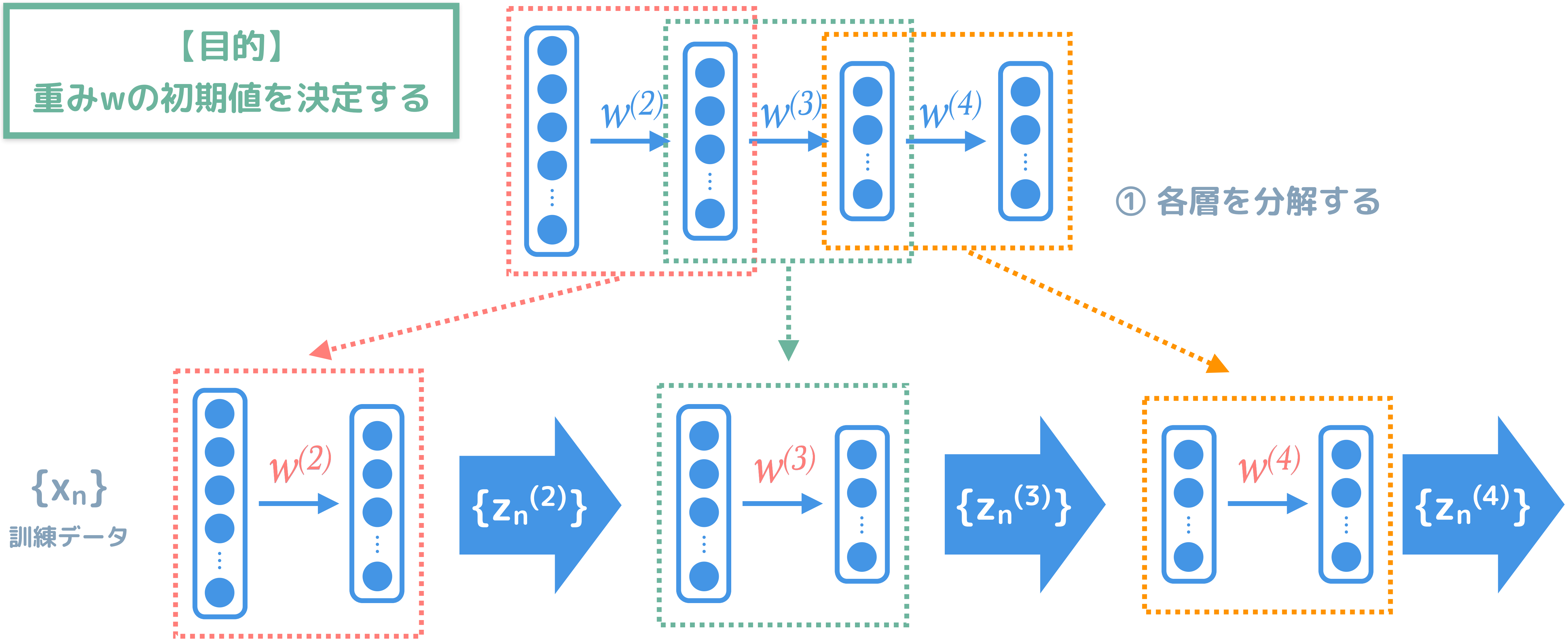
④ 自己符号化器を構成し
直前の$\{z_n^{(2)}\}$を訓練データとし
重み$w^{(3)}$を学習する

学習済み$w^{(2)}$を初期値に！

学習済み$w^{(3)}$を初期値に！

自己符号化器を用いた事前学習

【目的】
重みwの初期値を決定する



② 自己符号化器を構成し
訓練データ$\{x_n\}$から
重み$w^{(2)}$を学習する

③ 学習した$w^{(2)}$をセットし
訓練データ$\{x_n\}$を通し
出力$\{z_n^{(2)}\}$を得る

④ 自己符号化器を構成し
直前の$\{z_n^{(2)}\}$を訓練データとし
重み$w^{(3)}$を学習する

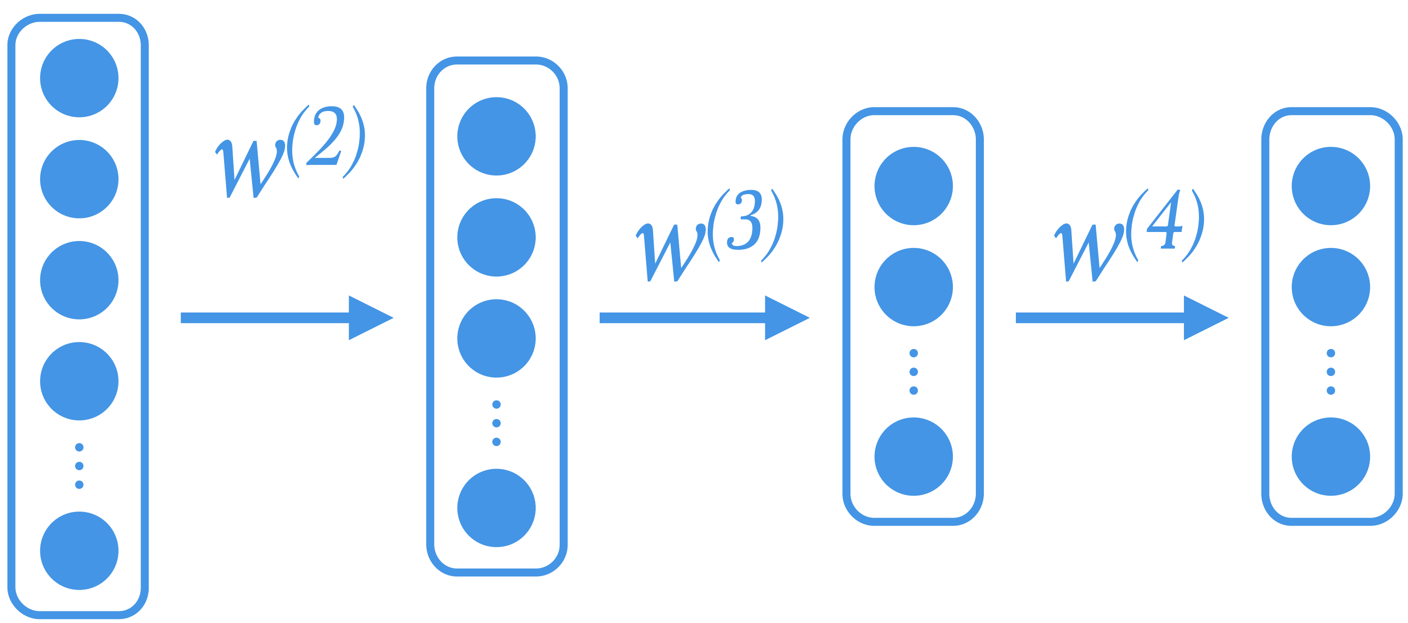
⑤ 繰り返し

学習済み$w^{(2)}$を初期値に！

学習済み$w^{(3)}$を初期値に！

自己符号化器を用いた事前学習

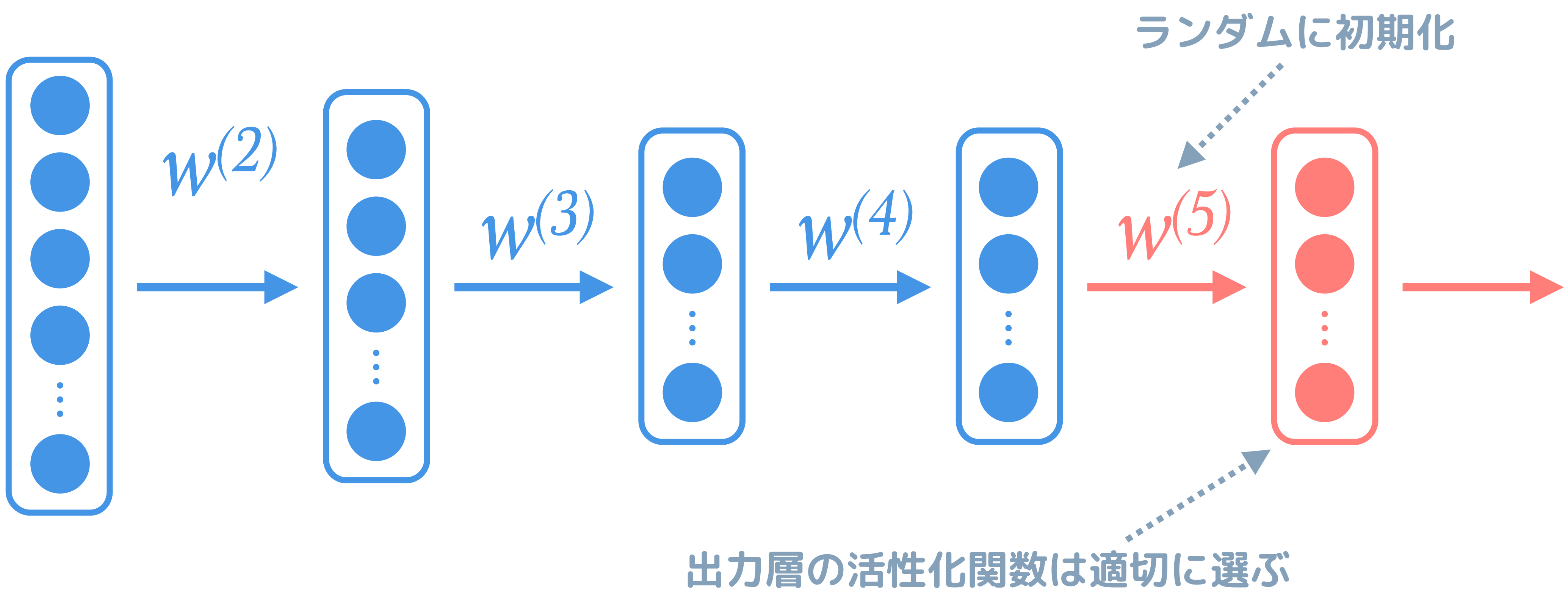
初期値が決まったので元のディープネットに戻り
本来の目的である教師あり学習（画像判別器の学習など）を行う



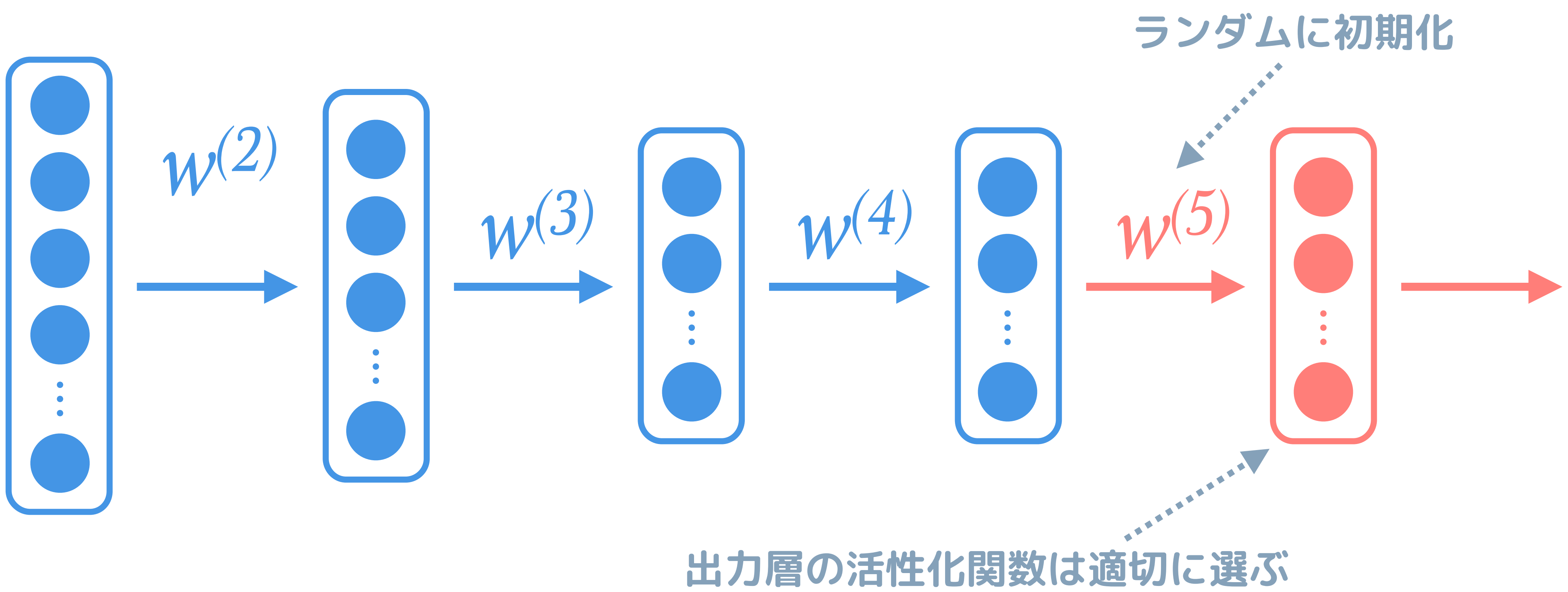
自己符号化器を用いた事前学習

初期値が決まったので元のディープネットに戻り
本来の目的である教師あり学習（画像判別器の学習など）を行う

その前に「出力層」を１層追加する

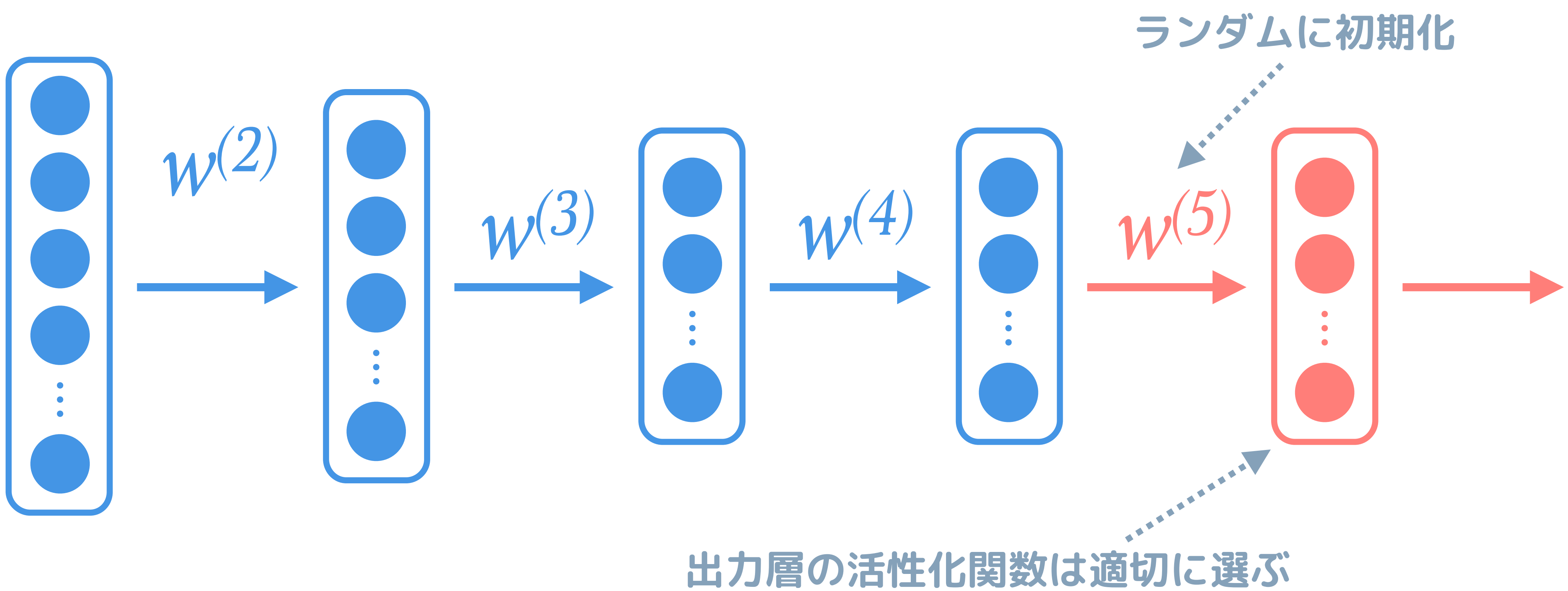


自己符号化器を用いた事前学習



自己符号化器を用いた事前学習

学習がより上手くすすむ！



良い学習できる初期値を得ることができた

良い学習できる初期値を得ることができた

事前学習には
他にもまだポイントがある

本のページ p74

その他のポイント

その他のポイント

- 特徴抽出器としての役割
  - 初期値を学習済みのネットワークを利用
  - 出力層を追加せず $\{z_n^{(4)}\}$ をxの特徴量みなす

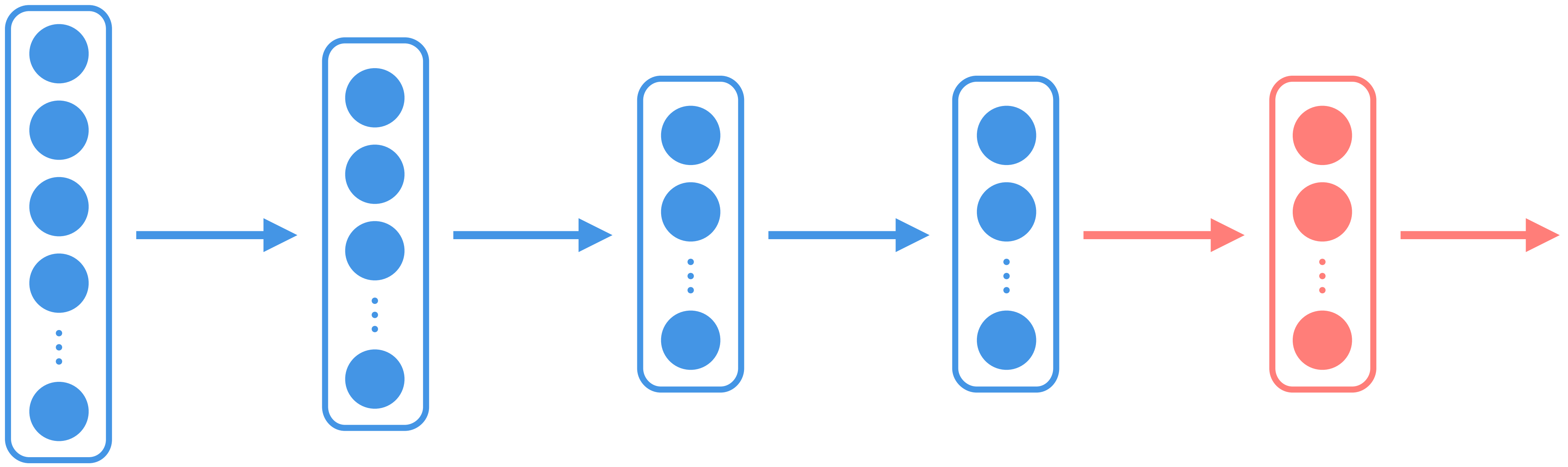
その他のポイント

- 特徴抽出器としての役割
  - 初期値を学習済みのネットワークを利用
  - 出力層を追加せず $\{z_n^{(4)}\}$ をxの特徴量みなす
- 学習がうまくいくのは経験則
  - 理論的になぜそうなるのかはわかっていない
  - データの広がりをうまく捉えているから…?

ディープネットの事前学習（まとめ）

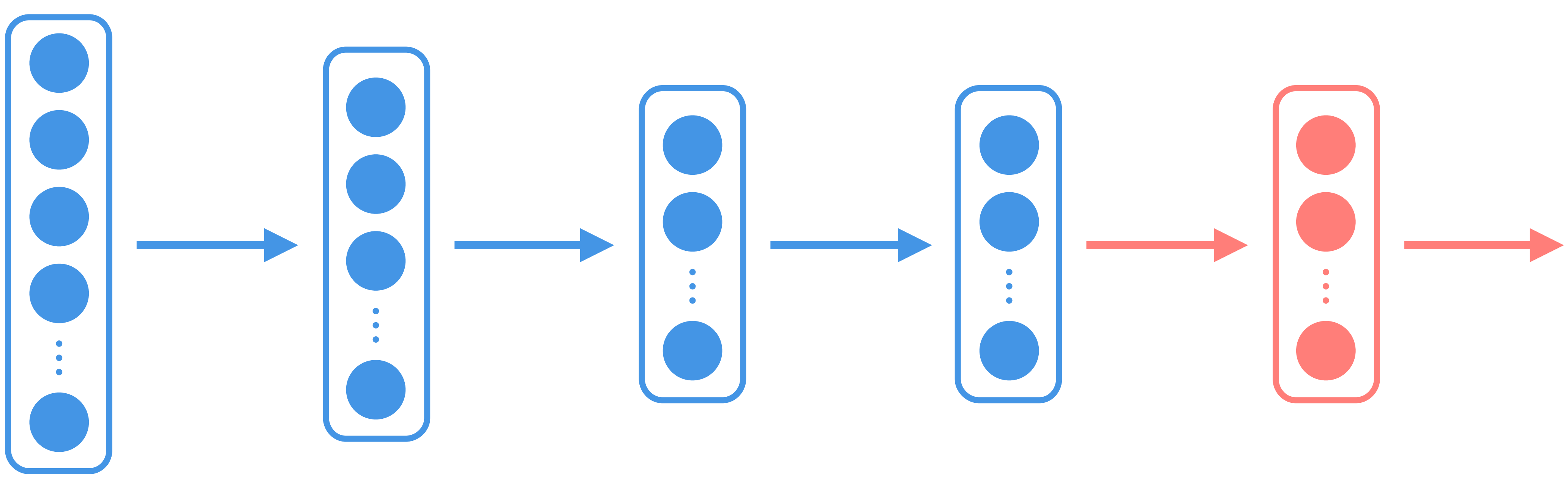
ディープネットの事前学習（まとめ）

・自己符号化器を用いて**各層の初期値**を得る



ディープネットの事前学習（まとめ）

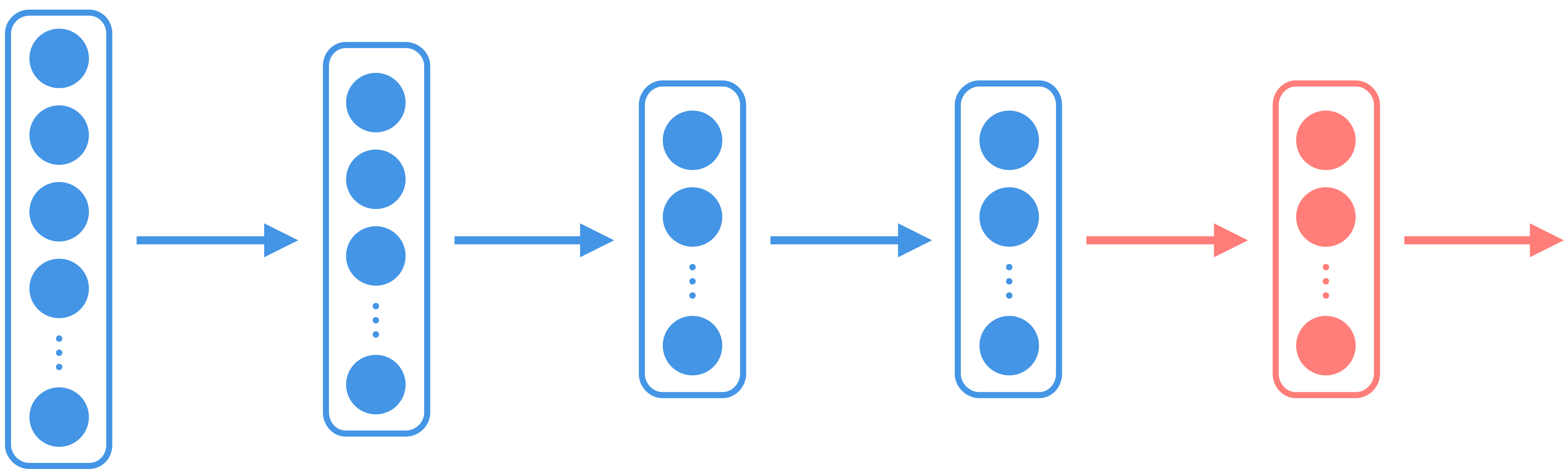
- 自己符号化器を用いて各層の初期値を得る
- そうすると、本来の目的の教師あり学習がうまくいく



本のページ p**

ディープネットの事前学習（まとめ）

- 自己符号化器を用いて各層の初期値を得る
- そうすると、本来の目的の教師あり学習がうまくいく
- うまくいくのは経験則



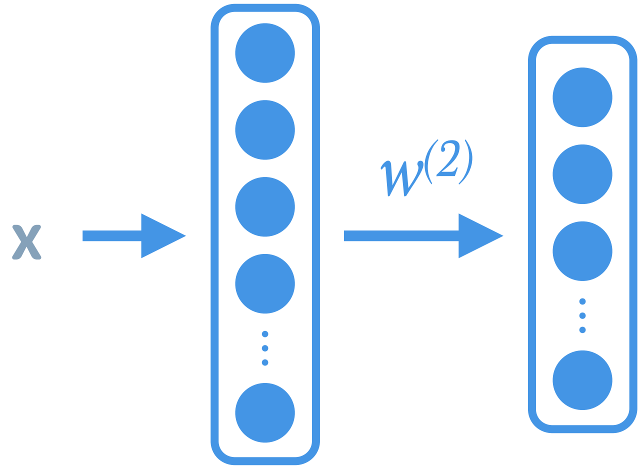
その他の自己符号化器

更に拡張した自己符号化器が存在する

多層自己符号化器

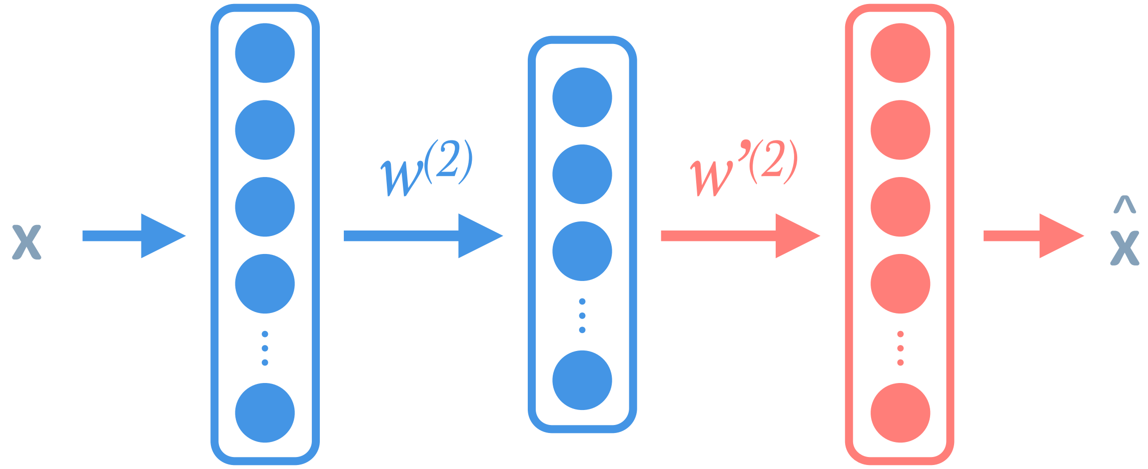
本のページ p74

多層自己符号化器



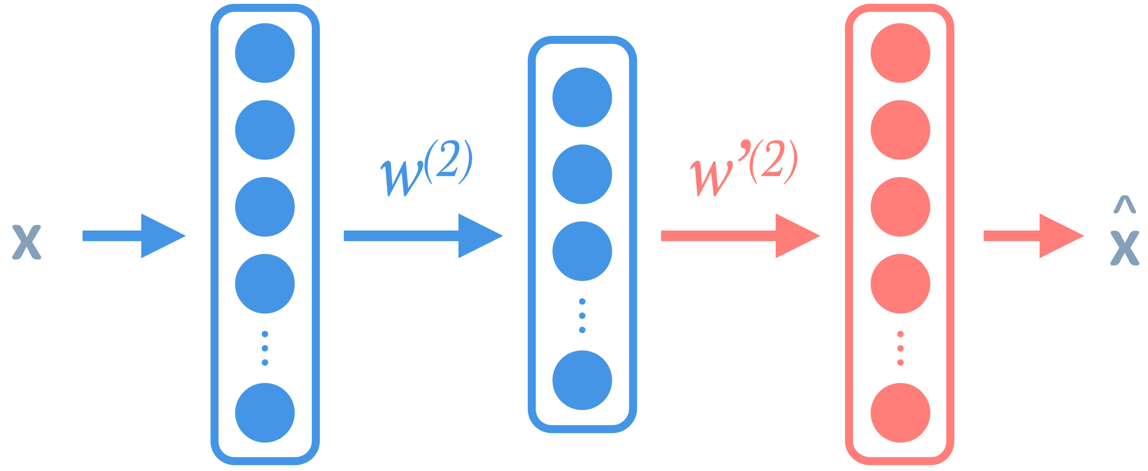
単層ネットワーク

多層自己符号化器

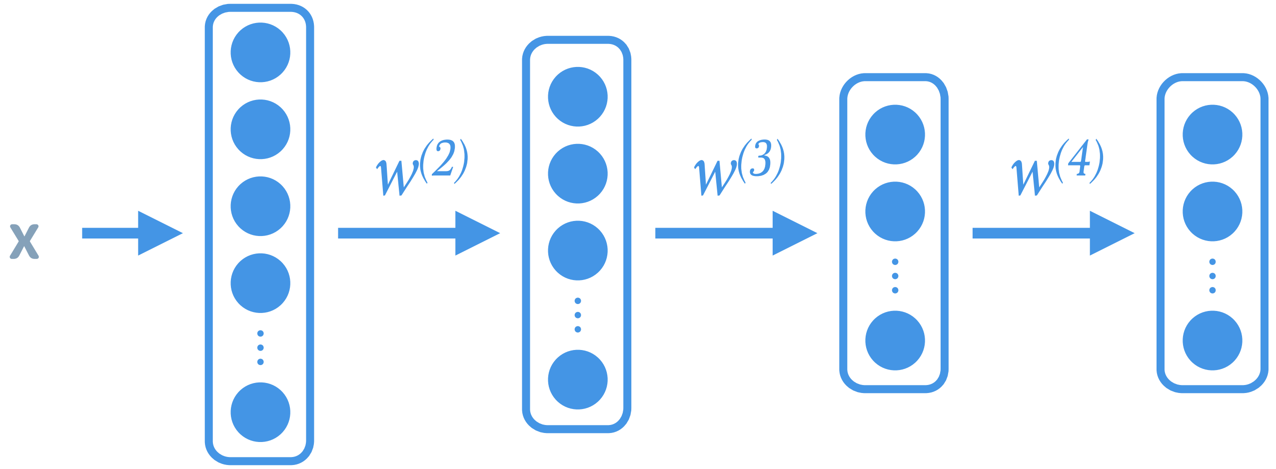


単層ネットワーク → 自己符号化器

多層自己符号化器

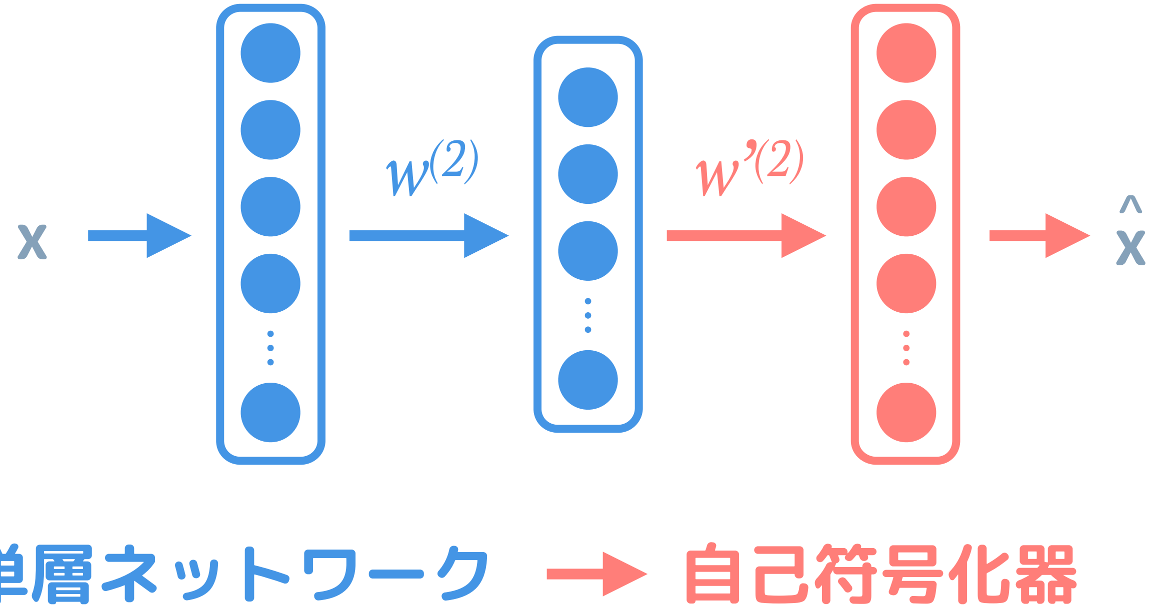


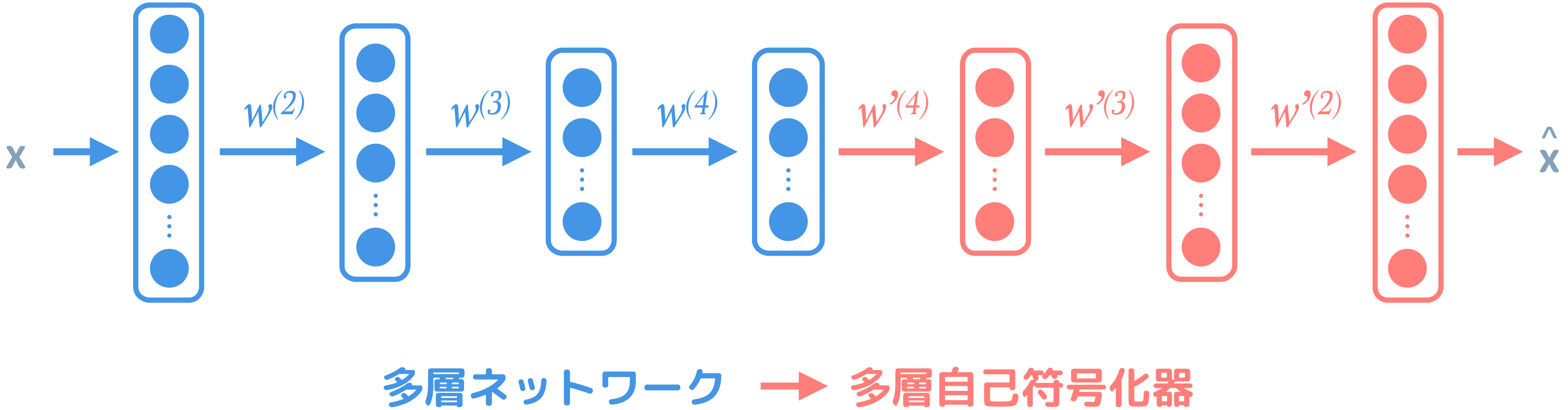
単層ネットワーク → 自己符号化器



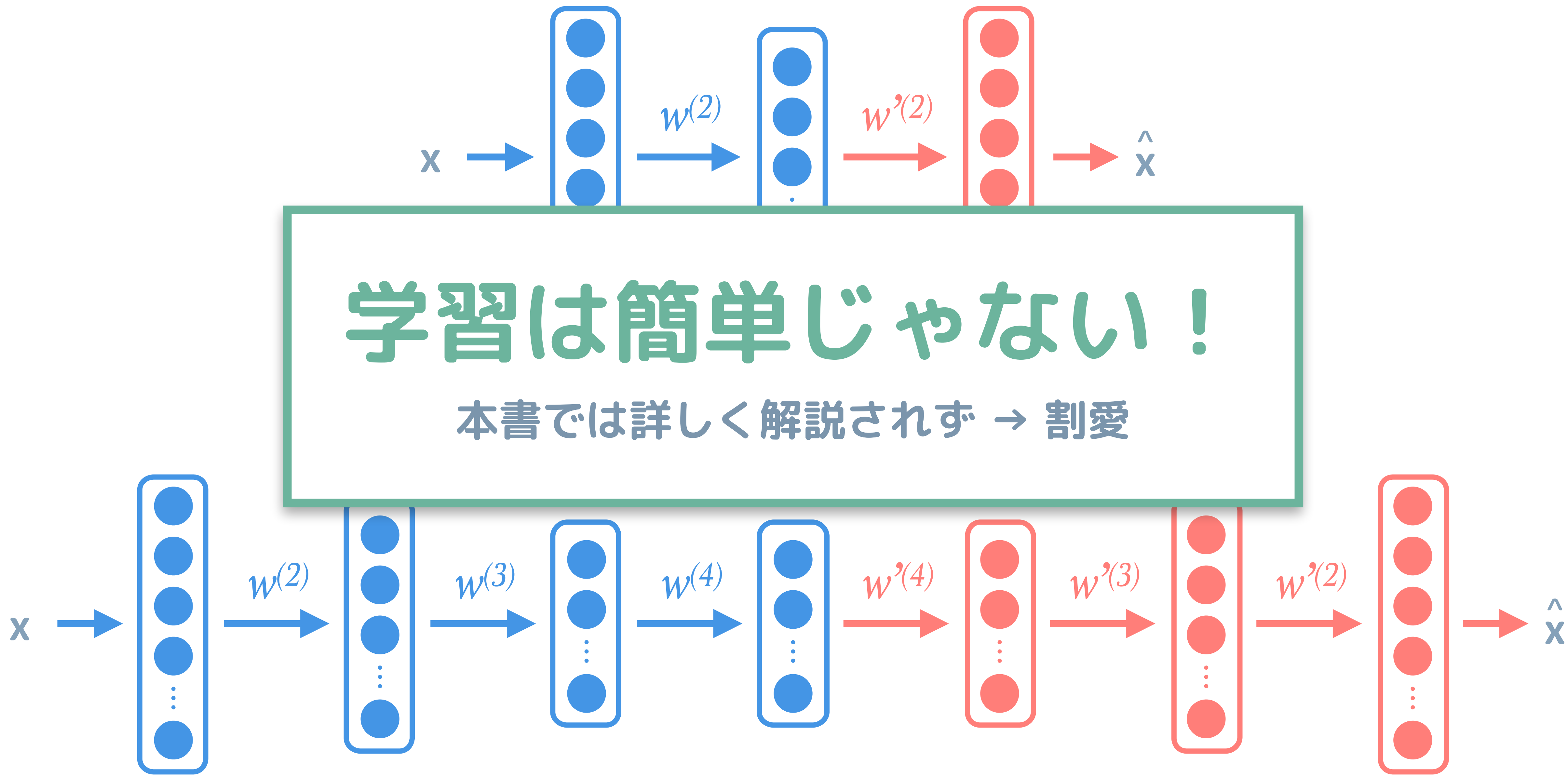
多層ネットワーク

多層自己符号化器


x
$w^{(2)}$
$w'^{(2)}$
$\hat{x}$
単層ネットワーク → 自己符号化器


x
$w^{(2)}$
$w^{(3)}$
$w^{(4)}$
$w'^{(4)}$
$w'^{(3)}$
$w'^{(2)}$
$\hat{x}$
多層ネットワーク → 多層自己符号化器

多層自己符号化器



学習は簡単じゃない！

本書では詳しく解説されず → 割愛

多層ネットワーク → 多層自己符号化器

デノイジング自己符号化器

デノイジング自己符号化器

その前に **制約ボルツマンマシン（RBM）** について述べます

デノイジング自己符号化器

その前に **制約ボルツマンマシン（RBM）** について述べます

- 用途や使い方は**自己符号化器とほぼ同じ**
- 最適化の手順には**確率的な要素**がある
- 自己符号化器よりも**性能が少し上**

本のページ p74

デノイジング自己符号化器

その前に 制約ボルツマンマシン（RBM） について述べます

・用途や使い方は自己符号化器とほぼ同じ
・最適化の手順には確率的な要素がある
・自己符号化器よりも性能が少し上

このままじゃ悔しい自己符号化器！

デノイジング自己符号化器

その前に **制約ボルツマンマシン（RBM）** について述べます

- 用途や使い方は**自己符号化器とほぼ同じ**
- 最適化の手順には**確率的な要素**がある
- 自己符号化器よりも**性能が少し上**

このままじゃ悔しい自己符号化器！

↓

デノイジング自己符号化器 の登場

本のページ p74

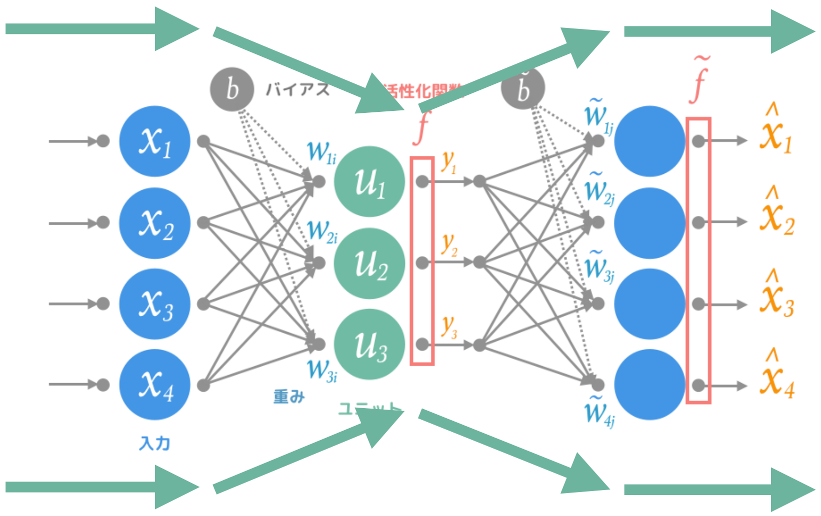
デノイジング自己符号化器

- 自己符号化器の拡張
- ネットワークの構造自体は自己符号化器と全く同じ
- 以下の方法で学習する

デノイジング自己符号化器

- 自己符号化器の拡張
- ネットワークの構造自体は自己符号化器と全く同じ
- 以下の方法で学習する

① 訓練データにランダムノイズを足す
② ノイズの乗ったデータで符号化と復号化を行う
③ 出力をノイズの乗る前のデータと比較する



入力の訓練データには
ノイズを入れるが、
出力との誤差比較には
ノイズの入っていない
訓練データを用いる

デノイジング自己符号化器

本のページ　p76

デノイジング自己符号化器

入力を再現するだけでなく

ノイズを除去する

効果が期待されている

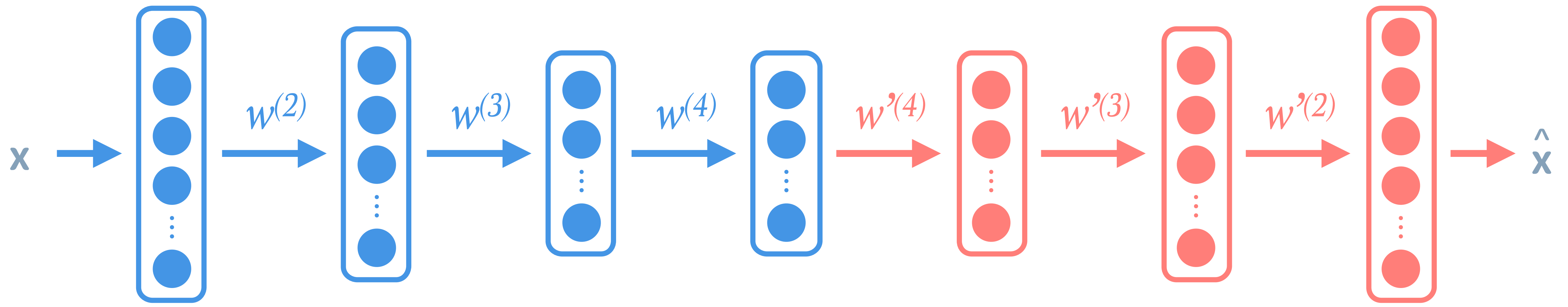
最終的にRBMと**同程度の性能**が報告されている

本のページ p76

その他の自己符号化器（まとめ）

その他の自己符号化器（まとめ）

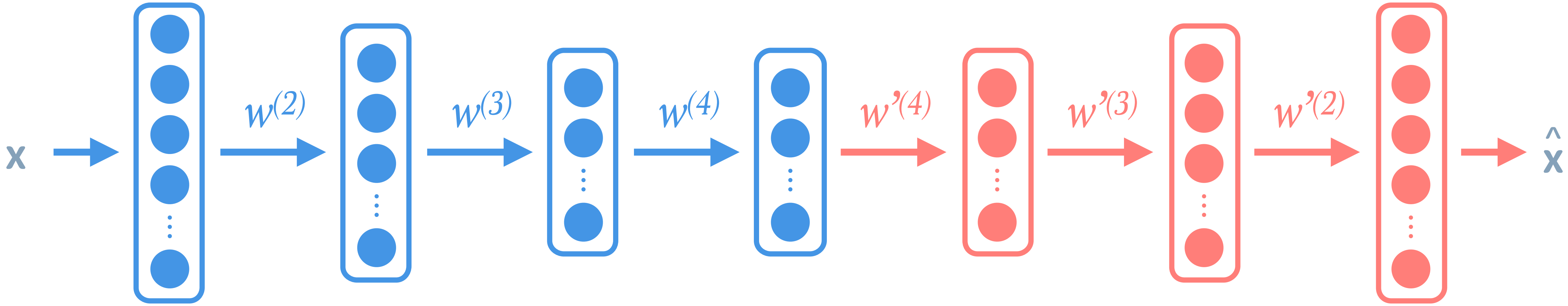
- **多層自己符号化器**は学習が難しい



本のページ p**

その他の自己符号化器（まとめ）

- 多層自己符号化器は学習が難しい
- デノイジング自己符号化器はノイズ除去効果に期待
  - RBMと同程度の性能を誇る



本のページ p**

全体まとめ

あともう少しです

全体まとめ

自己符号化器	・特徴の獲得 ・良い学習ができる初期値の獲得
スパース正則化	・計算量削減 ・データ量削減
白色化	・良い学習のための前処理
事前学習	・良い学習のための初期値獲得

今回の内容は「知識」としての内容なので
他の資料や論文を読む際の手助けになればと思います

実際に実務で利用できるようにするには
まだまだいろんな知識・経験が必要だと思います

よりよい自己符号化器ライフを！

ご清聴ありがとうございました