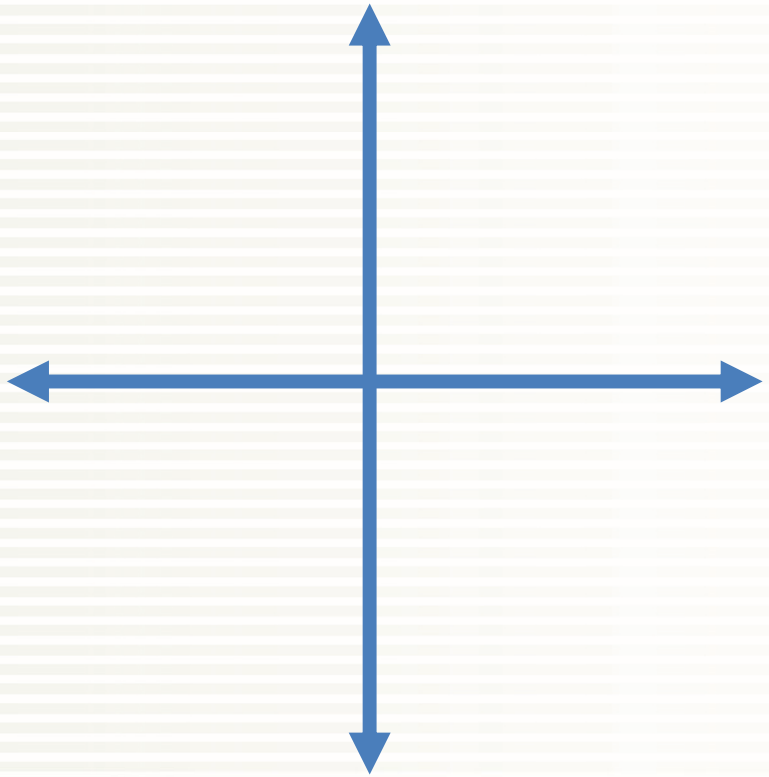
「弱いAI・強いAI」
「フレーム問題」

# What is AI (artificial intelligence)?

Intelligence の2つの軸：

human vs. rational

thought vs. behavior



# 1. Acting humanly: The Turing test approach

Turing test (1950) にパスするために、コンピュータには

自然言語処理 (natural language processing)

知識表現 (knowledge representation)

自動推論 (automated reasoning)

機械学習 (machine learning)

が必要になる。

# 1. Acting humanly: The Turing test approach

total Turing testをパスするために、ロボットには

コンピュータビジョン (computer vision)
ロボティクス (robotics)

も必要になる。

cf. 航空工学のゴール？

“machines that fly so exactly like pigeons
that they can fool even other pigeons”

# 2. Thinking humanly: The cognitive modeling approach

ヒトの思考について学ぶには:

内省 (introspection)

心理学実験 (psychological experiment)

脳機能イメージング (brain imaging)

さらには、認知科学 (cognitive science)。
しかし、AIとは通常区別される。

# 3. Thinking rationally: The “laws of thought” approach

三段論法 (syllogism): アリストテレス (Aristotle)

ソクラテスは人間である。

論理学 (logic)

確率 (probability)

Rational thought does not generate intelligent *behavior*.

# 4. Acting rationally: The rational agent approach

エージェント (agent)

合理的エージェント (rational agent)

act so as to acheive the best (expected) outcome

スタンダード・モデル (standard model)

study and construction of agents that “do the right thing”.

“the right thing” is defined by the objective we give to the agent

cf. perfect rationality vs. limited rationality

# スタンダード・モデルの問題点

人間が objective をコンピュータに与える必要がある：

チェスの場合

自動運転車の場合

value alignment problem

the machine’s objective, the human’s objective

Beneficial machines

# AIの隣接分野：哲学（philosophy）

形式的な規則から有効な結論を導くことはできるのか？

心(mind)は物理的な脳(brain)からどのように生じるのか？

知識の由来はどこか？

知識はいかにして行動につながるのか？

# AIの隣接分野：哲学（philosophy）

哲学者たち

# AIの隣接分野：数学（mathematics）

有効な結論を導くための形式的な規則とは何か？

計算できるものはどのようなものか？

不確実な情報に対してどのように推論するか？

形式論理学、確率、統計、アルゴリズム、
不完全性定理、計算可能性、P vs. NP

# AIの隣接分野：経済学（economics）

自身の好みにしたがってどのように決定を行うか？

他人と意見が合わないときはどうすべきか？

短期的には割に合わない場合はどうすべきか？

意思決定理論、ゲーム理論、オペレーションズ・リサーチ

# AIの隣接分野：神経科学（neuroscience）

脳はどのように情報処理するか？

ニューロン、オプトジェネティクス、
ブレイン・マシン・インターフェース、
シンギュラリティ

# AIの隣接分野：神経科学（neuroscience）

神経科学者たち

# AIの隣接分野：心理学（psychology）

動物およびヒトはどのように考え、行動するか？

行動主義（behaviorism）

認知心理学（cognitive psychology）

知能増幅（intelligent augmentation）

# AIの隣接分野：コンピュータ工学（computer engineering）

効率的なコンピュータはどうやって作れるか？

Mooreの法則

量子コンピューティング

# AIの隣接分野：制御理論 (control theory) とサイバネティクス（cybernetics）

人工物はその制御下でどのように行動できるか？

ホメオスタティック・デバイス、コスト関数

# AIの隣接分野：言語学 (linguistics)

言語はどのように思考に関係するか？

計算言語学

# AIの歴史(1): The inception (1943-56)

Warren McCulloch and Walter Pits (1943)

Donald Hebb (1949)

# AIの歴史(2): Early enthusiasm, great expectations (1952–69)

ダートマス会議(1956): John McCarthy, Marvin Minsky

Newell & Simon: General Problem Solver
物理記号システム仮説

Arthur Samuel: checkers

John McCarthy: Lisp

Minsky: Microworlds, Blocks world

# AIの歴史(3): A dose of reality(1966-73)

進化的計算

遺伝的プログラミング

群知能

# AIの歴史(4): Expert systems (1969–86)

DENDRAL

MYCIN

第5世代コンピュータ

# AIの歴史(5): The return of neural networks (1986-現在)

コネクショニズム

人工ニューラルネット(artificial neural network)

バックプロパゲーション

# AIの歴史(6): Probabilistic reasoning and machine learning (1987–現在)

ベンチマーク問題

隠れマルコフモデル

ベイジアンネットワーク

# AIの歴史(7): Big data (2001–現在)、Deep learning (2011–現在)

ビッグデータ

深層学習

CNN

# AIの歴史(8): State of the art

ロボット

自律的スケジューリング

機械翻訳

音声認識

レコメンデーション

# AIの歴史(8): State of the art

ゲーム

イメージキャプション

診断学

気候変動

# AIの歴史(9): かつてAIだったシステム

コンパイラ

かな漢字変換

# AIのリスクとベネフィット

自動殺人兵器

監視

偏った意思決定

雇用への影響

安全が決定的に大事な案件への応用

サイバーセキュリティ

# Human-level AI

John McCarthy (2007), Marvin Minsky (2007), Patrick Winston (2009)

“A machine should be able to learn to do anything a human can do.”

## Artificial general intelligence (AGI) movement

# Artificial superintelligence (ASI)

The Gorrila problem

King Midas problem

# 弱いAIと強いAI

John Searle (1980)

weak AI:  machines could act *as if* they are intelligent

strong AI: machines that do so are *actually* consciously thinking (not just *simulating* thinking)

その後、強いAIの定義は変化し、human-level AI や general AI を指すようになった。

cf. 中国語の部屋

# フレーム問題 (frame problem)

John McCarthy:　ある行為を記述しようとしたとき、
その行為によって変化する事柄と変化しない事柄を
すべて明示的に記述しようとするのは、記述量と
推論量が爆発して、現実的に扱えない

# フレーム問題 (frame problem)の解決

限定問題

波及問題

暗黙知の利用

環境との相互作用の利用

# 記号接地問題（symbol grounding problem）

ロボットの中に構築された記号システム内の記号が
どのようにして実世界と結びつけられるか

# グループワーク課題

現在のところ実現していない人工知能の例を挙げ、

・これまで実現していないのはなぜか、
・実現できれば社会にどのようなインパクトがあるか、
・実現するためには、どのような研究の積み重ねが
　今後必要か、
・研究の積み重ねの結果として、西暦何年に実現可能と
　考えられるか

について論じよ。各グループで意見を集約してもよいし、
複数の候補（たとえば近未来の例と遠未来の例）を
挙げてもよい。

# 探索･推論

# 人工知能をめぐる動向

- 第1次AIブーム（1956～1960年代）：探索・推論の時代
  - ダートマスワークショップ（1956）
    - 人工知能（Artificial Intelligence）という言葉が決まる
    - 世界最初のコンピュータENIAC (1946)のわずか10年後
  - 数学の定理証明、チェスを指す人工知能等
- …冬の時代

考えるのが早い人工知能

- 第2次AIブーム（1980年代）：知識の時代
  - エキスパートシステム
  - 医療診断、有機化合物の特定、…
  - 第5世代コンピュータプロジェクト：通産省が570億円
- …冬の時代

ものしりな人工知能

- 第3次AIブーム（2013年～）：機械学習・ディープラーニングの時代
  - ウェブとビッグデータの発展
  - 計算機の能力の向上

データから学習する人工知能

以下2ページの出典：松尾豊「人工知能に関する技術動向と産業分野への利用可能性」

http://venturebeat.com/2011/02/15/ibm-watson-jeopardy-2/, http://weekly.ascii.jp/elem/000/000/207/207410/

将棋電王戦

IBM ワトソン


ディープラーニング(2007-)

ILSVRCでの圧勝(2012)
Googleの猫認識(2012)
ディープマインドの買収(2013)
FB/Baiduの研究所(2013)

車・ロボットへの活用

自動運転
Pepper

ウェブ・ビッグデータ

機械学習

統計的自然言語処理
(機械翻訳など)

検索エンジンへの活用

タスクオントロジー

MYCIN(医療診断)
DENDRAL

エキスパートシステム

オントロジー

ワトソン(2011)

LOD(Linked Open Data)

Eliza

対話システムの研究

Caloプロジェクト

Siri(2012)

bot

探索
迷路・パズル

プランニング
STRIPS

チェス(1997)
Deep Blue

将棋(2012-)
電王戦

囲碁

1956 1970 1980 1995 2010 2015

第一次AIブーム
(推論・探索)

第二次AIブーム
(知識表現)

第三次AIブーム
(機械学習・ディープラーニング)

# 命題論理と述語論理

いずれも記号論理の一種。

命題論理：命題間の論理的関係を表現するが、
命題自体を分析できない
P: “Tom likes apples”

述語論理：命題を主語と述語に分け、内部を記述できる
like(Tom, apple)

# 述語論理の記号

定数記号：特定の個体を表す　サトシ
変数記号：任意の個体　　$x$

関数記号：個体間の関係　MOTHER

述語記号：個体に関する性質・状態　like

論理記号：否定、連言、選言、含意、同値
限量記号：全称記号、存在記号

例：サトシの母が好きでないものがある

# 述語論理式

定数記号：特定の個体を表す　Satoshi
変数記号：任意の個体　　$x$

関数記号：個体間の関係　MOTHER

述語記号：個体に関する性質・状態　like

論理記号：否定、連言、選言、含意、同値
限量記号：全称記号、存在記号

例：　$\exists x \neg \mathrm{like}(\mathrm{MOTHER}(\mathrm{Satoshi}), x)$

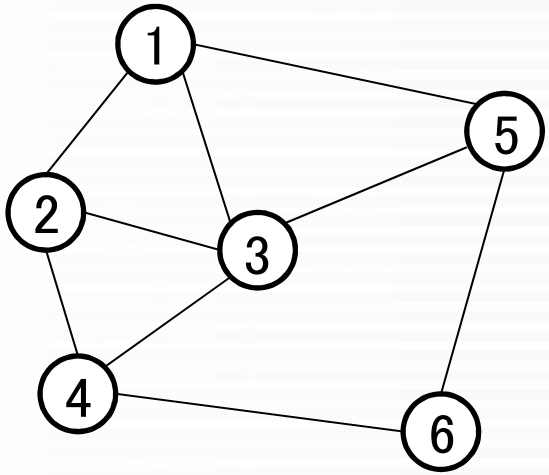
# 述語論理式の例

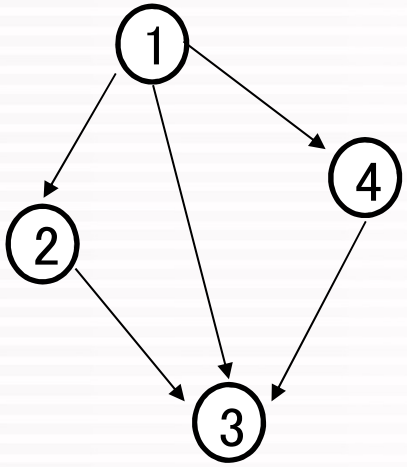
1. 私は本を持っている。
2. 私は本かノートを持っている。
3. すべての男子はケーキが好きだ。
4. 誰も自分の背中を触れない。
5. ペンギン以外の鳥は飛ぶ。

# 探索による問題解決

# グラフ

グラフ：頂点の集合 $V$ と辺の集合 $E$ の組
無向・有向グラフ：グラフの各辺に向きがない・あるもの




無向グラフの例: ソーシャルネットワーク、交通ネットワーク
有向グラフの例: タスクの依存関係、ゲームの局面遷移

# グラフによる定式化

1. 問題をグラフ G = (V, E) で表現する。

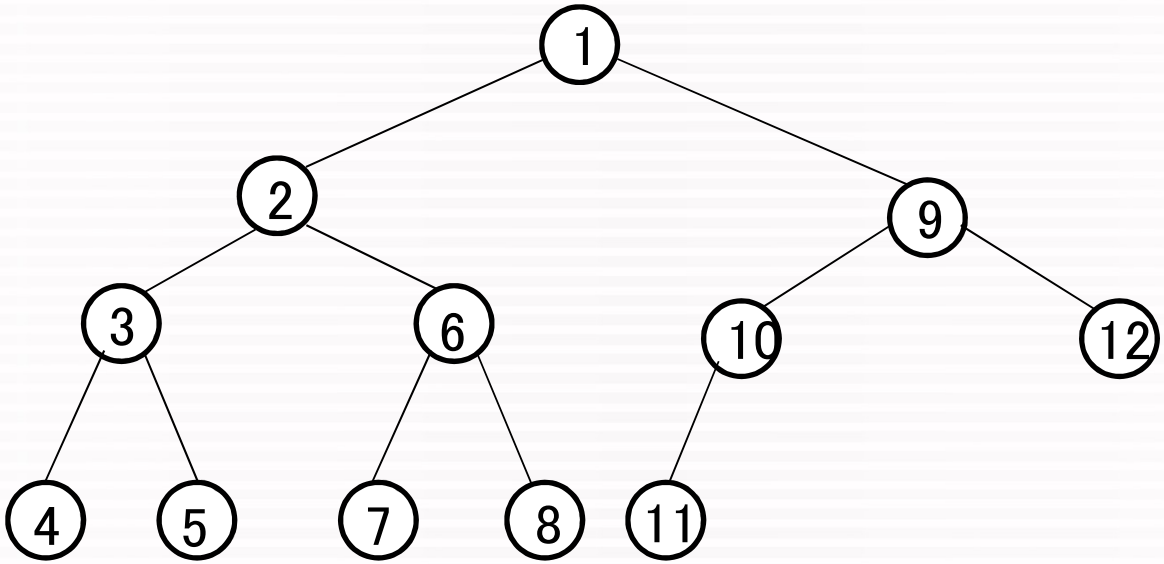
2. 状態全体がノード集合Vとなる。

3. 状態 i でオペレータが適用可能であり、その適用で
   状態が j に変化するとき、グラフのノード i, j間にはエッジ（辺）
   がある、すなわち (i, j) ∈ E。

4. 探索による問題解決とは、初期状態 s ∈ V とゴール状態
   g ∈ V が与えられたとき、s から g に到達するグラフ上の
   パス（経路）を見いだすこと。

# 縦型探索（深さ優先探索）

DFS （depth-first search）とも



# 縦型探索（深さ優先探索）

OL (open list):  これから探索する状態を格納

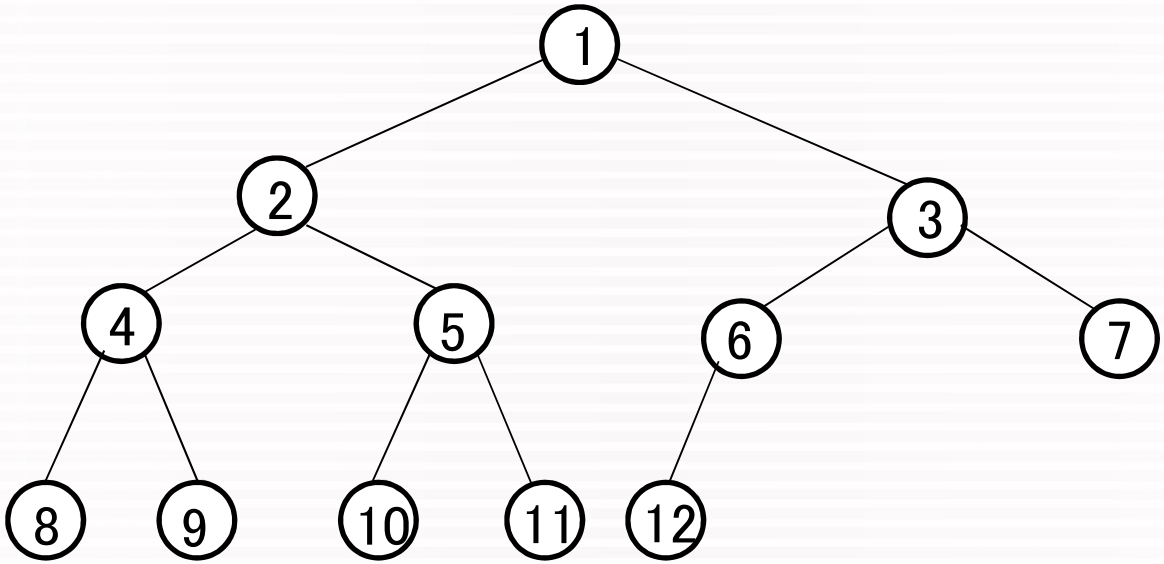
CL (closed list): すでに探索した状態を記録し、同じ状態を
繰り返し探すことを防ぐ。

親子ポインタリスト：これまでにたどったパス（経路）の記録。

OLの先頭に、次に探索する子ノードをすべて追加する。
Last In First Out のスタック構造。

# 横型探索(幅優先探索)

BFS (breadth-first search) とも



# 横型探索(幅優先探索)

OL (open list): これから探索する状態を格納

CL (closed list): すでに探索した状態を記録し、同じ状態を繰り返し探すことを防ぐ。

親子ポインタリスト:これまでにたどったパス(経路)の記録。

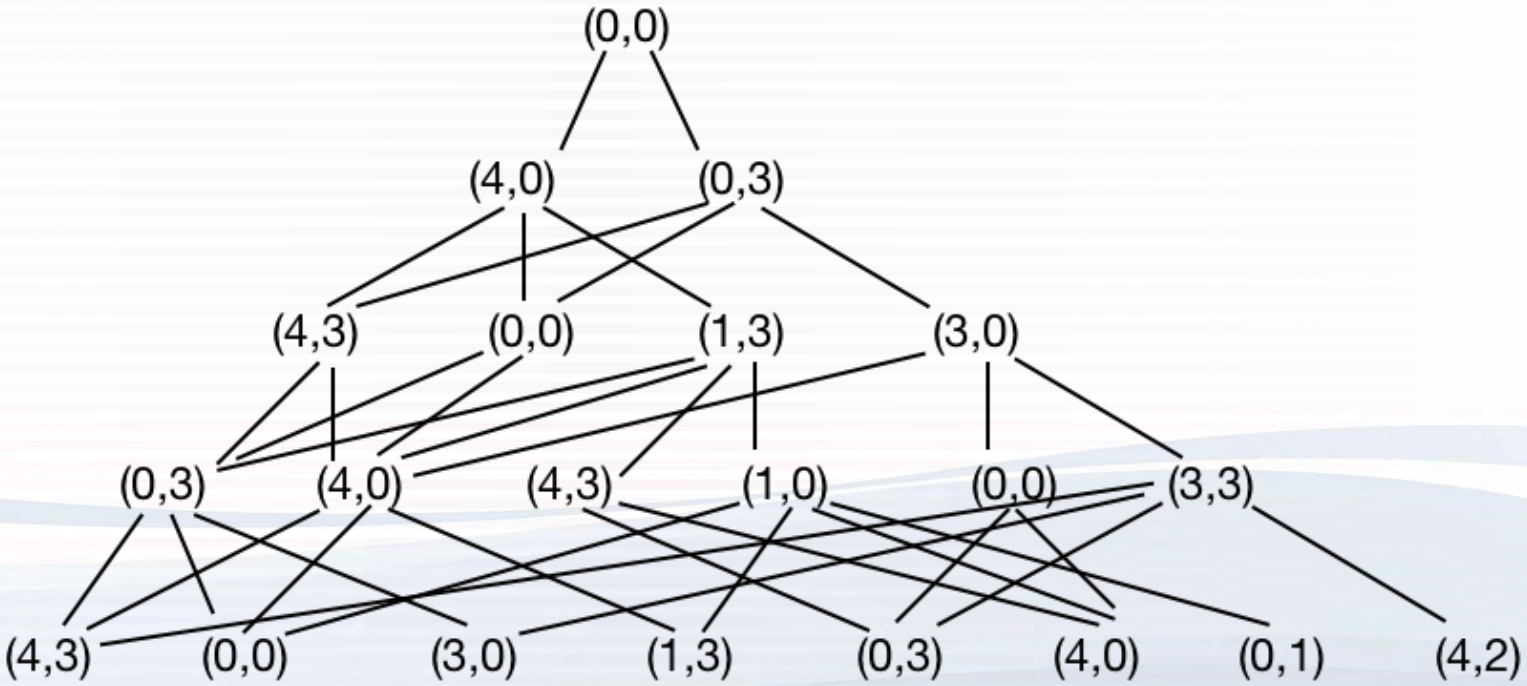
OLの末尾に、次に探索する子ノードをすべて追加する。
First In First Out のキュー構造。

# 探索の例

例1：迷路探索シミュレータ

https://algoful.com/Archive/Algorithm/DFS

例2：４Ｌと３Ｌの容器を用いて、２Ｌの水を最小の操作回数ではかる方法



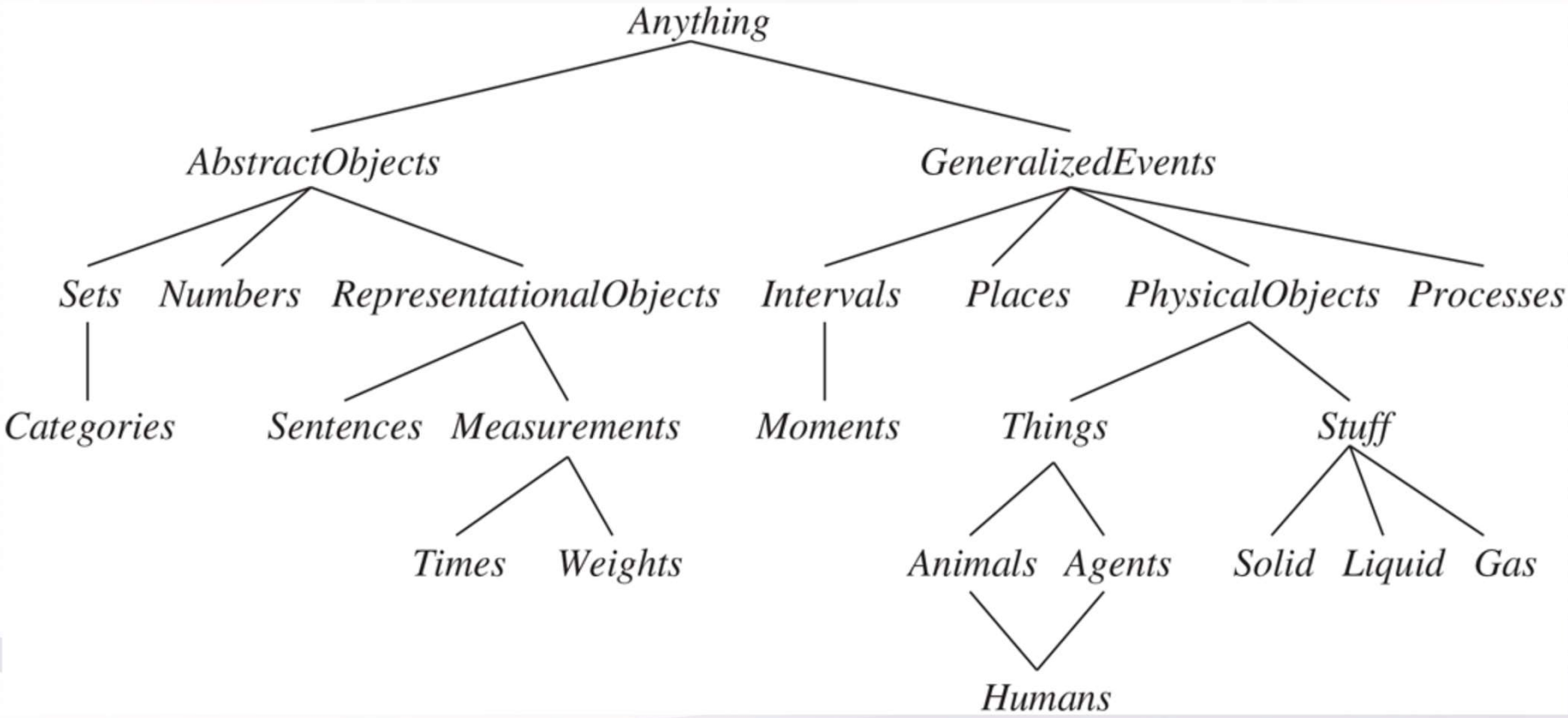
# 知識表現

# 知識表現（knowledge representation）

人工知能プログラムが扱える形式で知識を表現すること。
扱いやすいよう、適切に構造化されている。

# オントロジー（ontology）

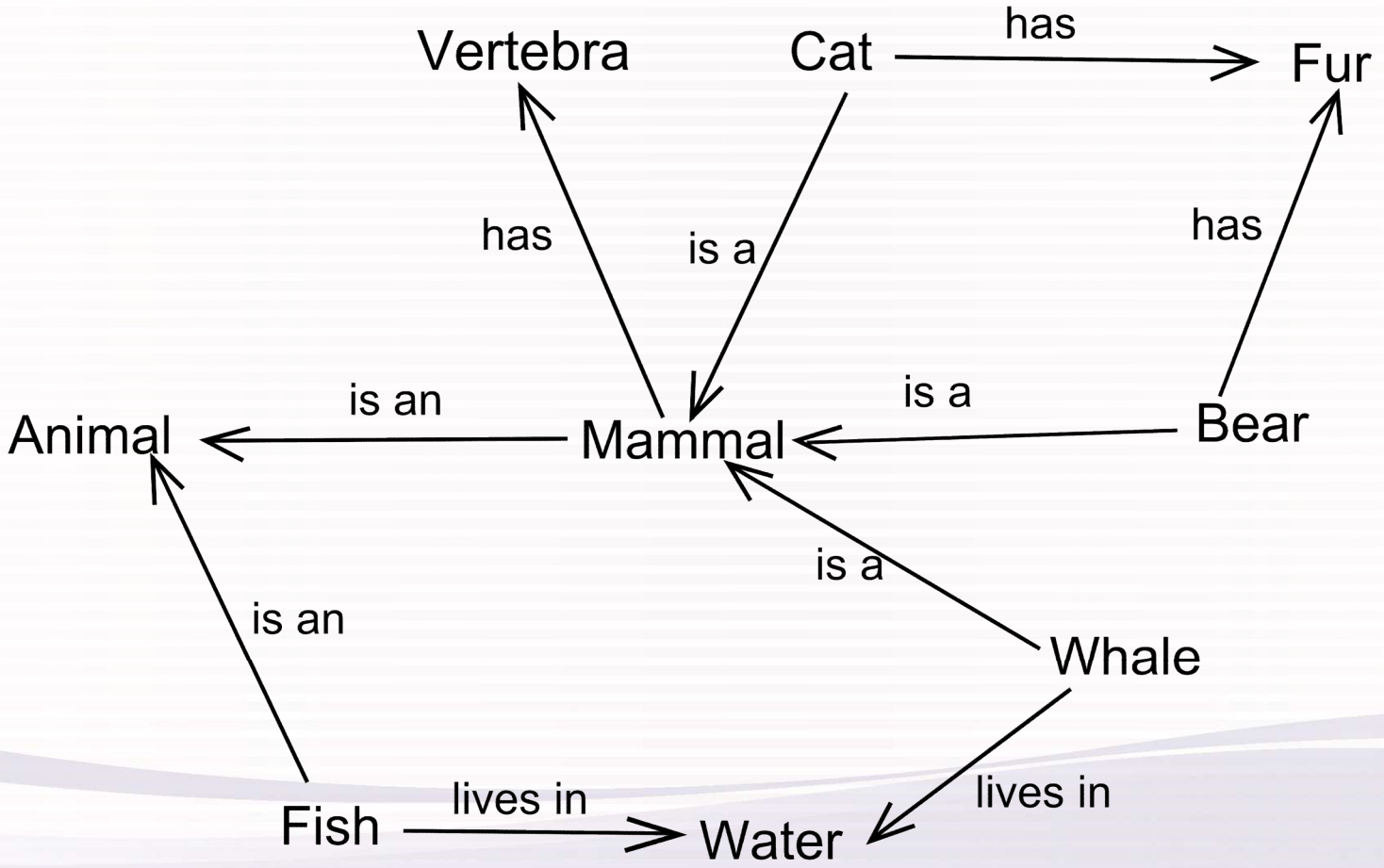
Representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities




The upper ontology of the world

# 意味ネットワーク（semantic network）

概念のラベルをノードとし、ノード間をリンクで結んで
作成したネットワークを用いて意味を表現する。



# フレーム(frame)表現

意味ネットワークにおける、概念を表現するノードの
内部に、スロットという構造をもち、値を代入できる。

# プロダクションシステム(production system)

if–then 形式で概念の関係を表すプロダクションルールを用いて知識を表現する。

# 述語(predicate)による知識表現

述語とは論理学の概念で、ある具体的事例を与えると
真偽を決定できる記述のこと。

# 開世界仮説と閉世界仮説

開世界仮説：意味ネットワークに記述された知識以外にも知識が存在する。

閉世界仮説：意味ネットワークに記述された知識は完全であり、それ以外に知識は存在しない。

（例）デスクトップにはディスク装置がありますか？

→ わかりません（開世界仮説）
→ ディスク装置はありません（閉世界仮説）

# エキスパートシステム（expert system）

知識ベース（長期記憶）:
プロダクションルール*の形の知識の集まり

＊ IF L1, L2, ..., Ln THEN R1, R2, ..., Rm

推論エンジン：知識ベースを用いた推論を実行

ワーキングメモリ（短期記憶）:
事実や推論の途中状態を保持する

# 前向き推論

ワーキングメモリ中の事実と、長期記憶中のルールの
条件部の照合を行い、条件部が成立するすべてのルールを
求める。選択されたルールの実行部をワーキングメモリに
追加する。

例:
長期記憶
IF Y THEN B
IF X, B THEN A.

ワーキングメモリ
X
Y

# 後ろ向き推論

ユーザが与えたゴールがワーキングメモリの中の事実として
存在するか確認する。なければ、ゴールを実行部に含む
ルールを探し、ルールの条件部を新たなゴールとする。

例:
長期記憶
IF Y THEN B
IF X, B THEN A.

ワーキングメモリ
X
Y

# 曖昧な知識の表現

確信度：エキスパートシステムのMYCINで用いられた。
事実に対する確信度とルールに対する確信度を表現

非古典論理：古典論理（命題論理、述語論理）を
拡張したもの。

3値論理：真理値が0, 1, 0.5(わからない)
ファジー論理：真理値が0から1の連続値をとる

ベイズの定理：知識獲得による確率の変化を記述

# 条件つき確率

事象Aが起きたときに事象Bが起きる確率

# ベイズの定理の応用

スパムメールの判別：
「割引」という単語が含まれるメールがスパムとなる確率？

# ベイズの定理の応用

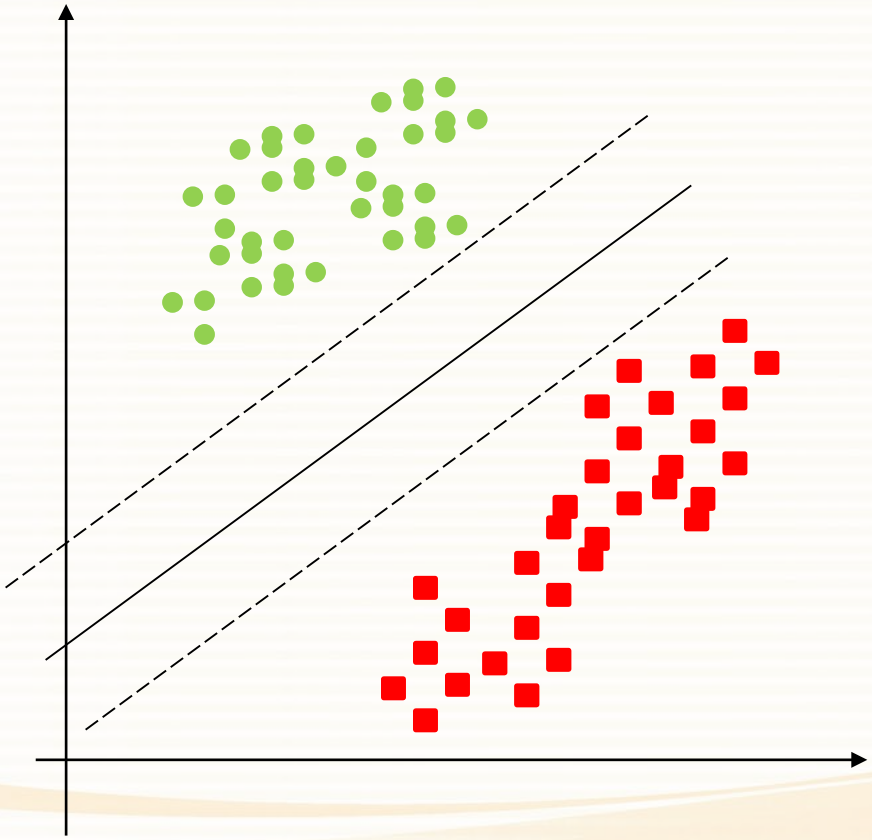
スパムメールの判別：
「割引」という単語が含まれるメールがスパムとなる確率？

# データサイエンス応用コース
# （データの分類・サポートベクターマシン
# ランダムフォレスト）

高野　渉
大阪大学

# 識別関数

2つのクラスに属するデータの境界（識別面）
を関数にて表す．この関数を識別関数という．



この3つの識別面の中で
どれがよりいい識別面で
あろうか？

# 識別関数

学習データセット $\{(\boldsymbol{x}_i, y_i) \mid i = 1,2,\cdots,N, \boldsymbol{x} \in R^n, y \in (-1,1)\}$

サポートベクターマシン(Support Vector Machine : SVM)

２つの識別面を線形関数として表す.

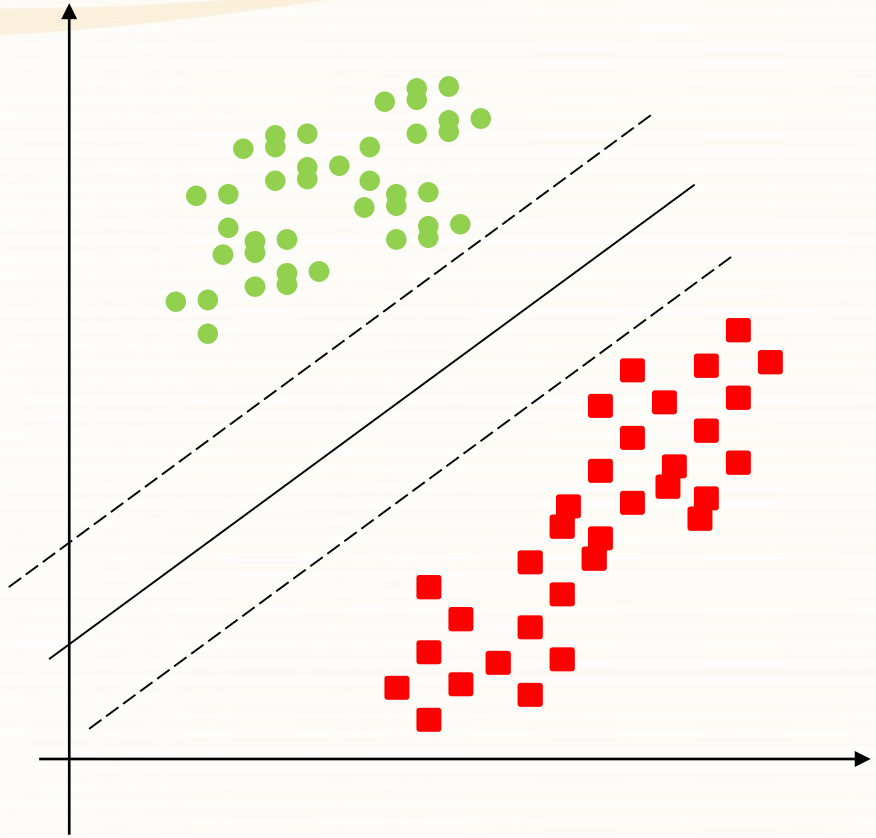
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} - b$$

観測データを識別関数に代入して,

$$f(\boldsymbol{x}) \geq 0 \text{ なら } y = 1$$
$$f(\boldsymbol{x}) < 0 \text{ なら } y = -1$$

として識別されるとする.

# 識別関数



この3つの識別面の中で
どれがよりいい識別面で
あろうか？

2つのクラスから遠い平面を識別面として採用する
（マージン最大化）

# サポートベクターマシン

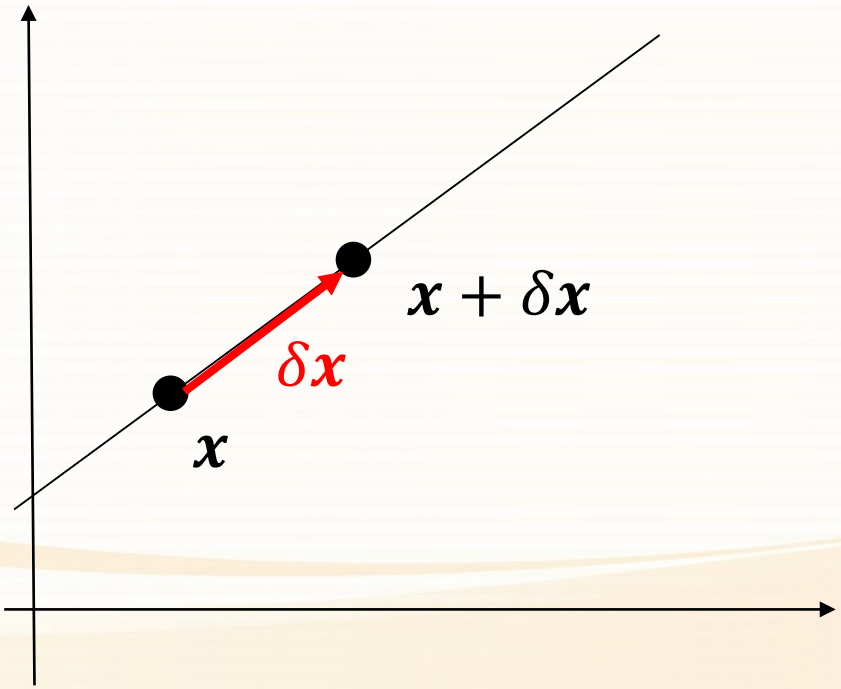
識別面と法線ベクトル

識別面上の2点 $\boldsymbol{x}, \boldsymbol{x}+\delta\boldsymbol{x}$ を考える．識別面上にあるので，

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} - b = 0,$$

$$f(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{x}+\delta\boldsymbol{x}) - b = 0$$

が成立する



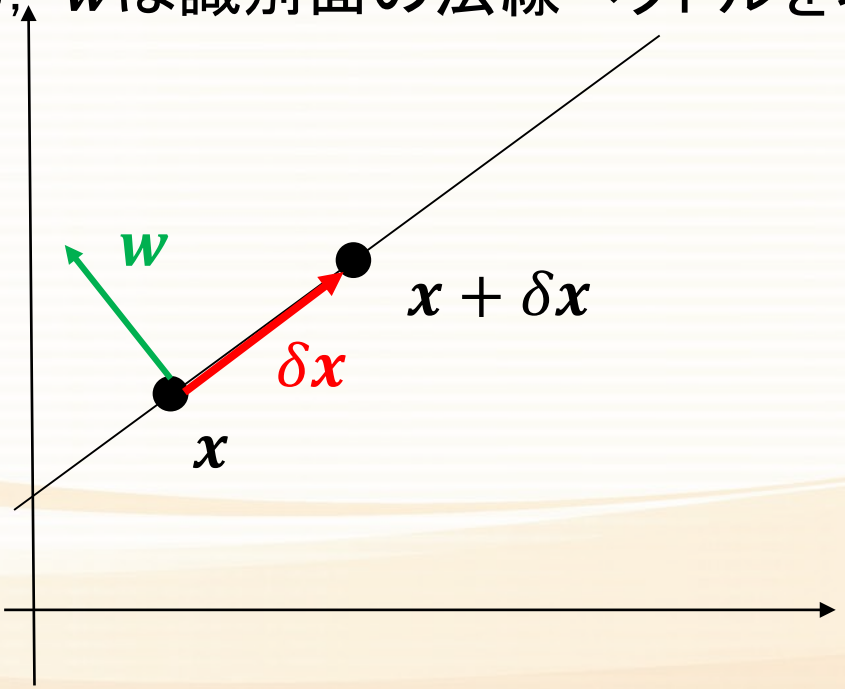
# サポートベクターマシン

先の2式の差より，

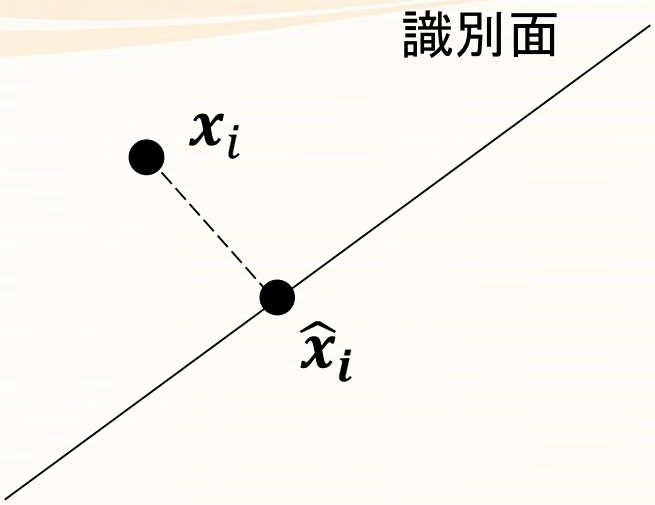
$$f(\boldsymbol{x}+\delta\boldsymbol{x})-f(\boldsymbol{x})=\boldsymbol{w}^T\delta\boldsymbol{x}=0,$$

の関係式が得られる．

$\boldsymbol{w}$と$\delta\boldsymbol{x}$の内積がゼロ，すなわち2つのベクトルは直交することから，$\boldsymbol{w}$は識別面の法線ベクトルを表すことになる．



# サポートベクターマシン




学習データ $x_i$ と識別面との距離は,

学習データ $x_i$ から識別面へ下した垂線の足を $\hat{x}_i$ とすると

$$d_i = \frac{y_i \, \boldsymbol{w}^T (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)}{\|\boldsymbol{w}\|}$$

# サポートベクターマシン

$\boldsymbol{w}^T\hat{\boldsymbol{x}}_{\boldsymbol{i}} - b = 0$ なので,

$$d_i = \frac{y_i\, \boldsymbol{w}^T(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_{\boldsymbol{i}})}{\|\boldsymbol{w}\|}$$

$$= \frac{y_i\, (\boldsymbol{w}^T\boldsymbol{x}_i - \boldsymbol{w}^T\hat{\boldsymbol{x}}_{\boldsymbol{i}})}{\|\boldsymbol{w}\|}$$

$$= \frac{y_i\, (\boldsymbol{w}^T\boldsymbol{x}_i - b)}{\|\boldsymbol{w}\|}$$

# サポートベクターマシン

距離 $d_i$の最小値の分子を1とするように制約を設けるとマージンは

$$d = \frac{1}{\|\boldsymbol{w}\|}$$

となる．マージン最大化は，$\|\boldsymbol{w}\|$の最小化と同値である．

したがって，マージン最大化の識別関数を求める問題は，

$$minimize\ \frac{1}{2}\ \|\boldsymbol{w}\|^2$$

$$subject\ to\ \ y_i(\boldsymbol{w}^T\ \boldsymbol{x}_i\ - b)\ - 1 \geq 0, \qquad i = 1,2,\cdots,N$$

# サポートベクターマシン

Lagrange 緩和問題

$$\max_{\alpha}\left[\min_{\boldsymbol{w},b}\left\{\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_i \alpha_i\{y_i(\boldsymbol{w}^T\boldsymbol{x}_i-b)-1\}\right\}\right]$$

制約を満たさない場合
は罰則項として機能する.

$$L=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_i \alpha_i\{y_i(\boldsymbol{w}^T\boldsymbol{x}_i-b)-1\}$$

$$\frac{\partial L}{\partial \boldsymbol{w}}=\boldsymbol{w}^T-\sum_i \alpha_i y_i \boldsymbol{x}_i{}^T=0$$

# サポートベクターマシン

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w}^T - \sum_i \alpha_i y_i \boldsymbol{x}_i{}^T = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i \quad \cdots (1)$$

法線ベクトルが学習データの
線形和で表すことができる.

$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0 \quad \cdots (2)$$

式(1) をLagrange関数に代入する

# サポートベクターマシン

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \boldsymbol{x}_i\right)^T \left(\sum_j \alpha_j y_j \boldsymbol{x}_j\right)$$

$$- \sum_i \alpha_i \left\{ y_i \sum_j \alpha_j y_j \boldsymbol{x}_j{}^T \boldsymbol{x}_i - y_i b - 1 \right\}$$

$$= \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j$$

$$- \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j + \sum_i \alpha_i y_i b + \sum_i \alpha_i$$

式(2) よりゼロ

# サポートベクターマシン

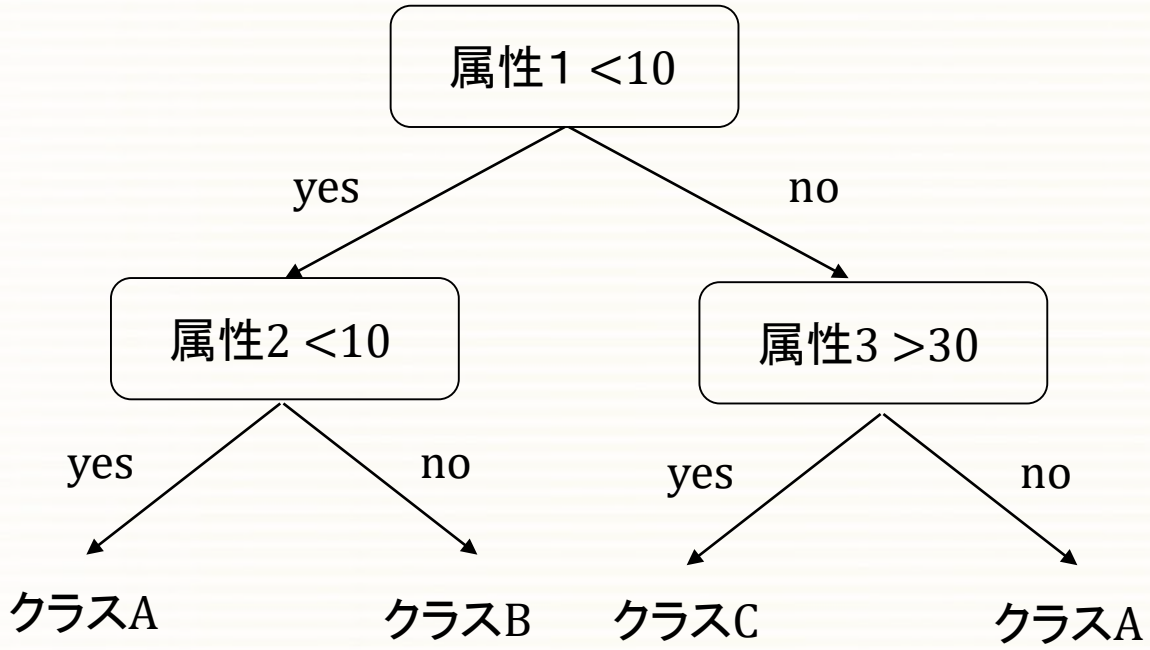
$$L = -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j + \sum_i \alpha_i$$

この最適化問題は2次計画問題として解くことができる

$\alpha_i \neq 0$に対応する$\boldsymbol{x}_i$だけを用いて，識別面の法線ベクトルが式(1)として計算できる．この$\boldsymbol{x}_i$をサポートベクトルと呼ぶ．

# 決定木 (Decision Tree)

識別する条件を階層構造（木構造）にて記述し、上位から下位の識別条件に照らしあわせながらデータを識別する。



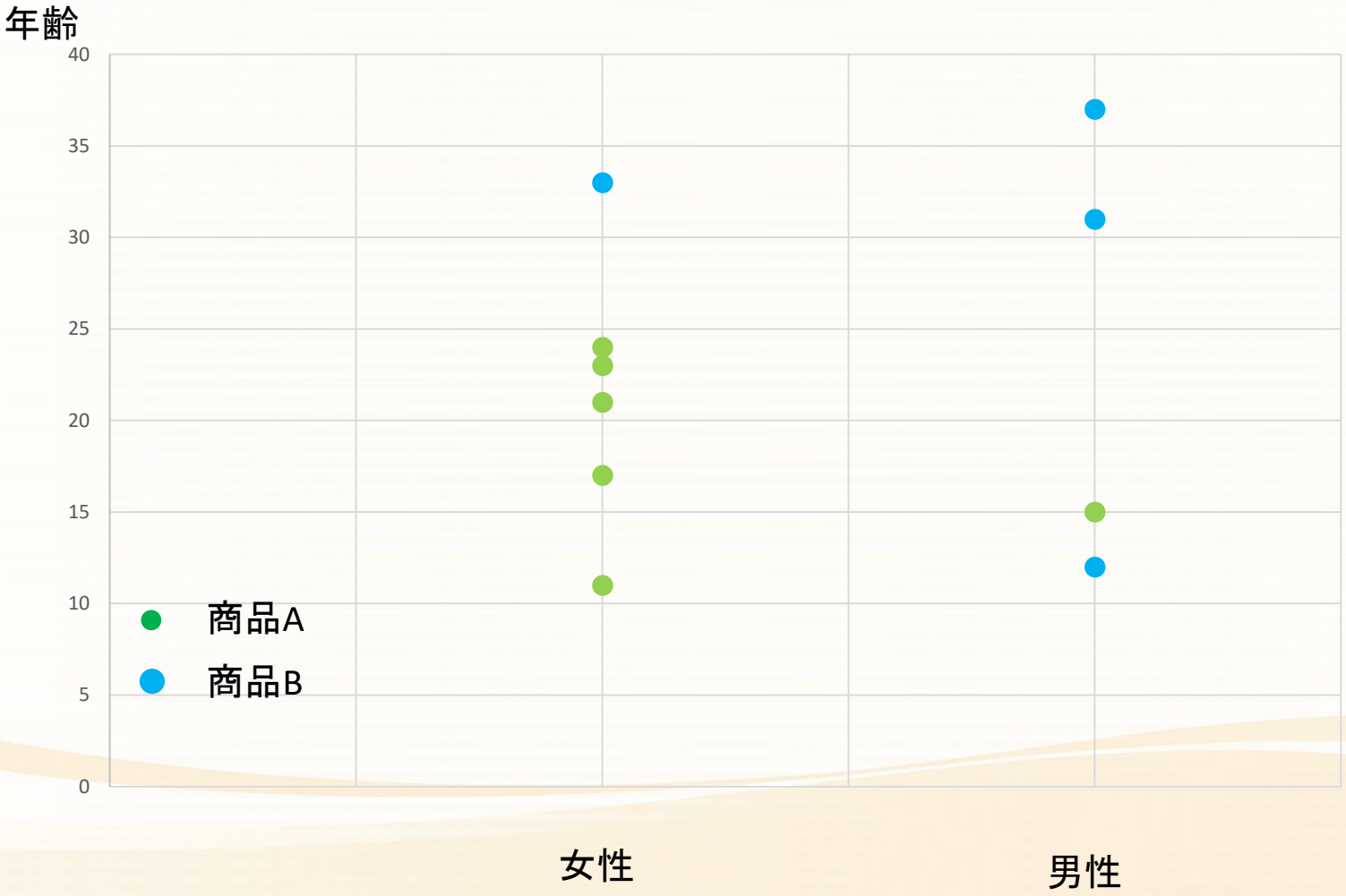
# 決定木 (Decision Tree)

2つの商品（商品A,B）のどちらか好きなほうを選択したとき、
ユーザーの性別および年齢情報を収集した。

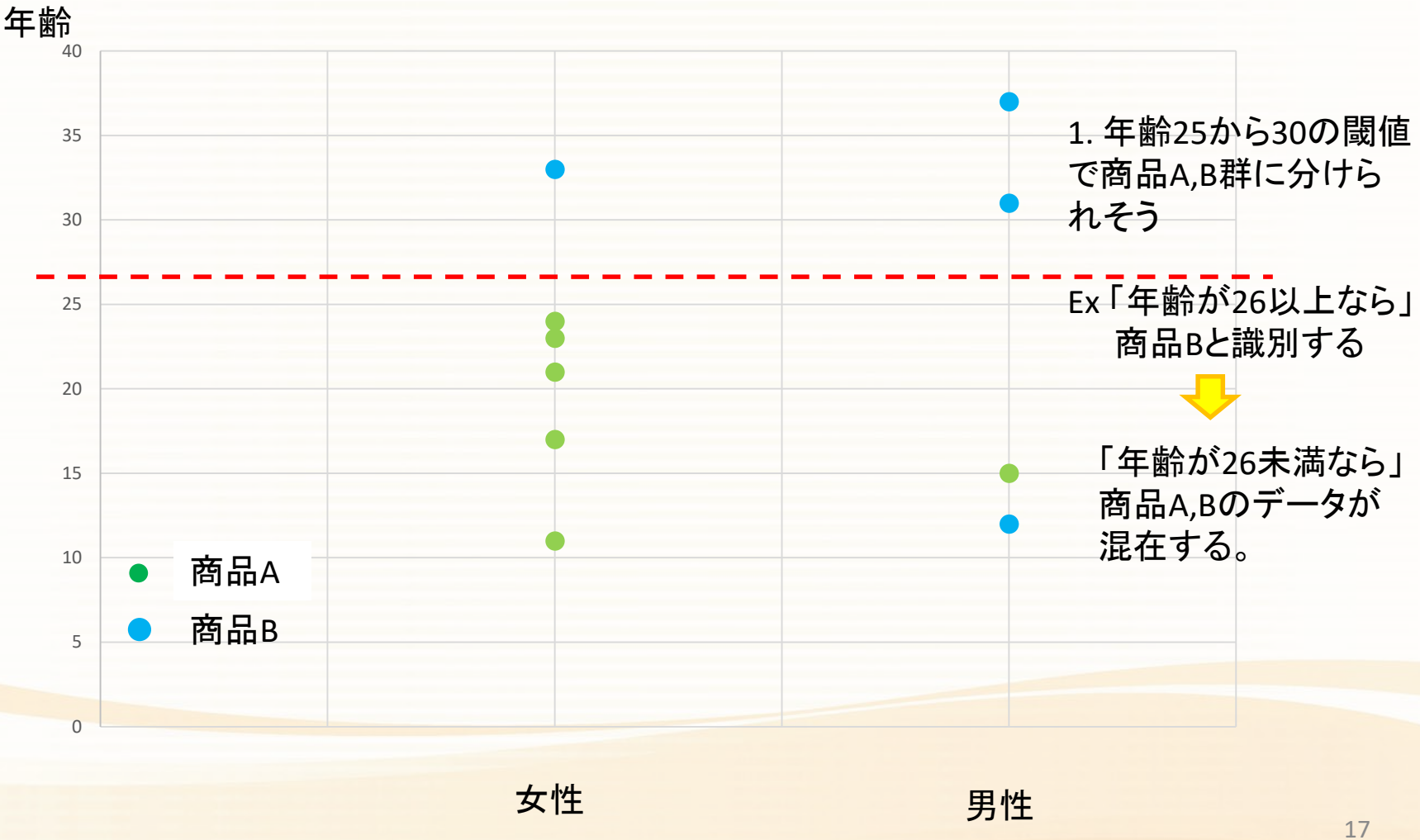
| ID | 性別 | 年齢 | 商品 |
|---|---|---|---|
| 1 | 女性 | 11 | A |
| 2 | 女性 | 23 | A |
| 3 | 女性 | 24 | A |
| 4 | 男性 | 12 | B |
| 5 | 男性 | 15 | A |
| 6 | 女性 | 33 | B |
| 7 | 男性 | 37 | B |
| 8 | 女性 | 17 | A |
| 9 | 女性 | 21 | A |
| 10 | 男性 | 31 | B |

# 決定木 (Decision Tree)

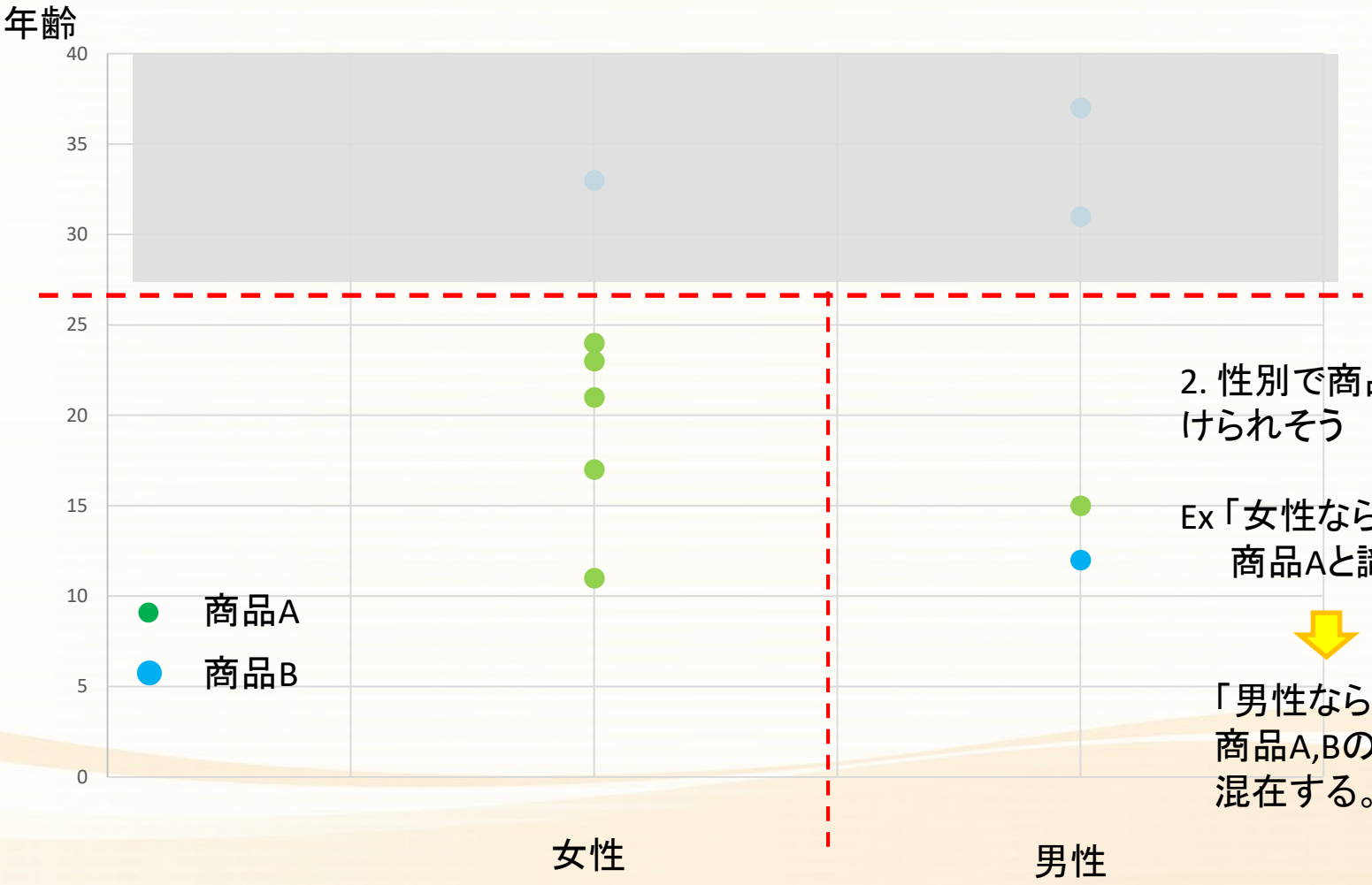
性別・年齢と選択商品のデータを散布図にして可視化する。
商品A,Bを識別するための条件を見つけてみよう



# 決定木 (Decision Tree)



# 決定木 (Decision Tree)

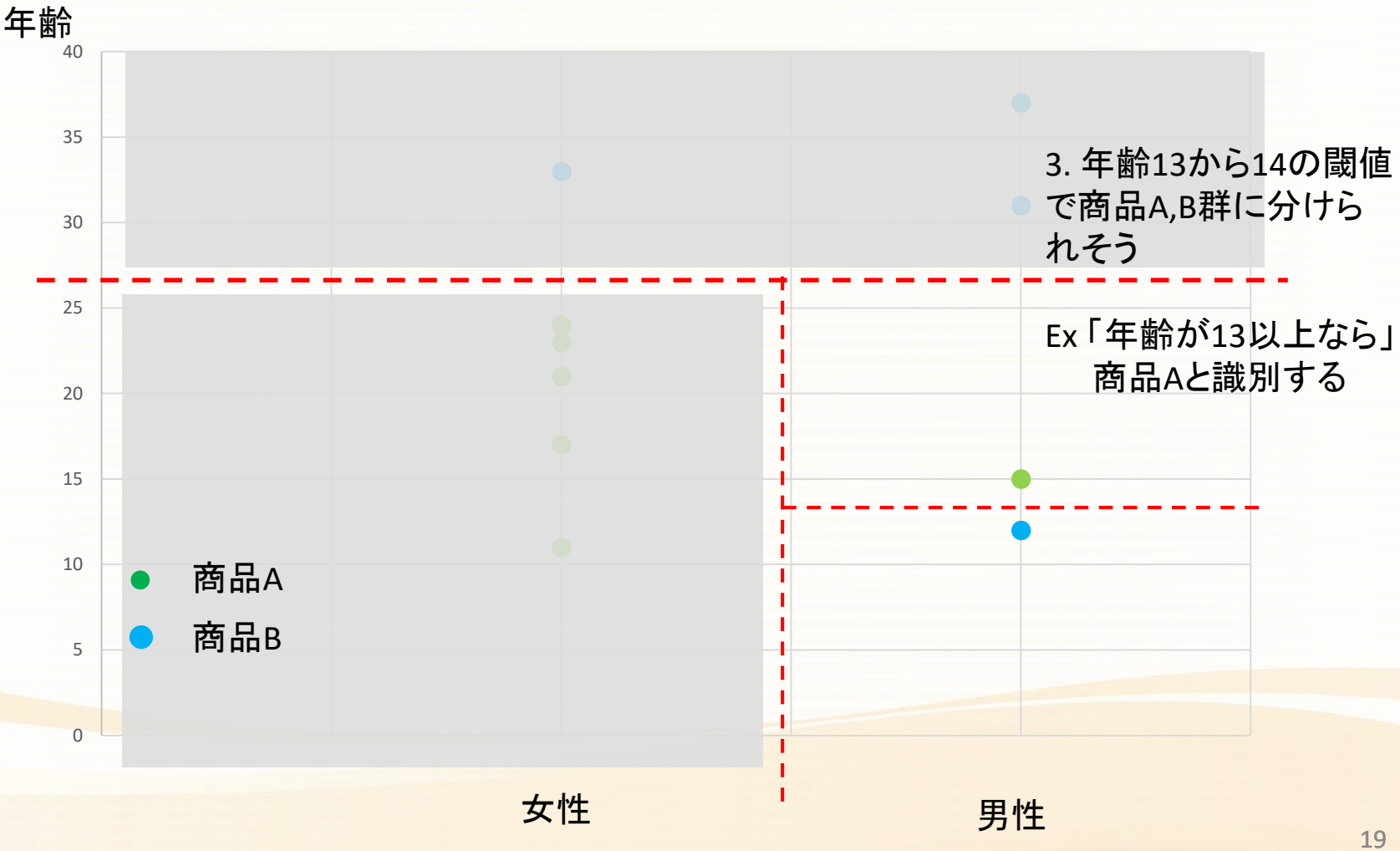



2. 性別で商品A,B群に分けられそう

Ex「女性なら」
商品Aと識別する

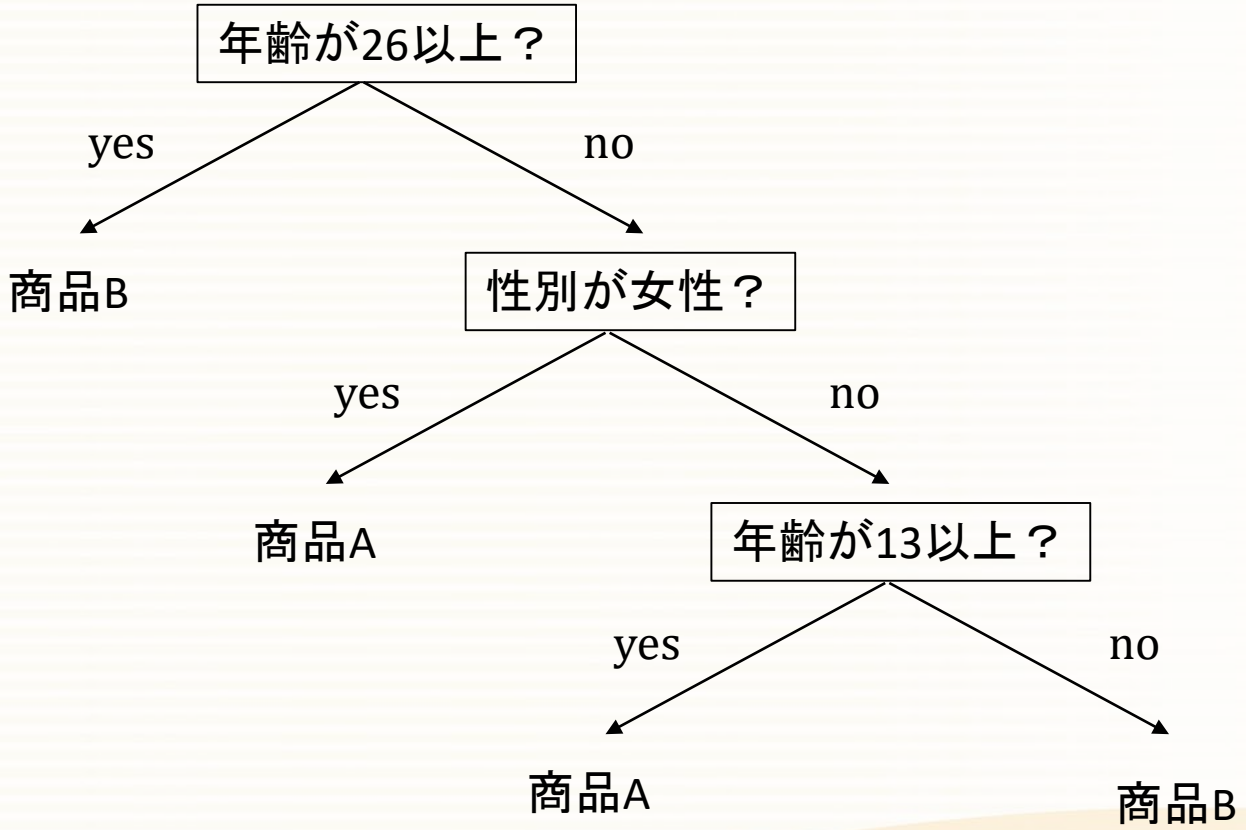
「男性なら」
商品A,Bのデータが混在する。

# 決定木 (Decision Tree)



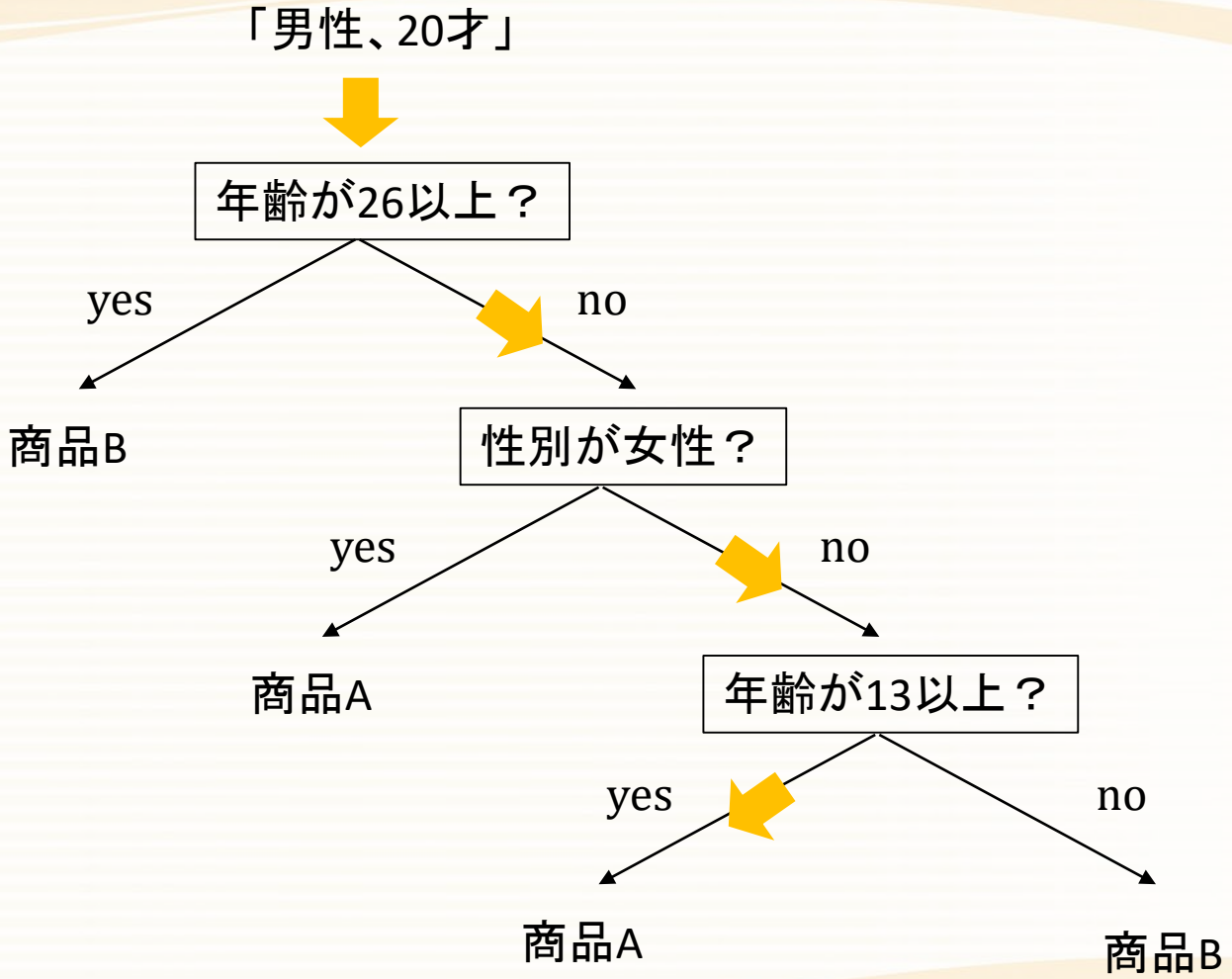
# 決定木 (Decision Tree)

以上の識別条件を階層構造にまとめる。




この決定木を用いて、男性、20才のユーザーがどちらを選択するか考えてみる

# 決定木 (Decision Tree)




「男性、20才」を入力として、上段の条件ごとに判別を繰り返していく。
最終的にたどり着く答えは、商品Aを選択。

# 決定木 (Decision Tree)

決定木の各ノードに設定された判別条件をどのように決めるのか？

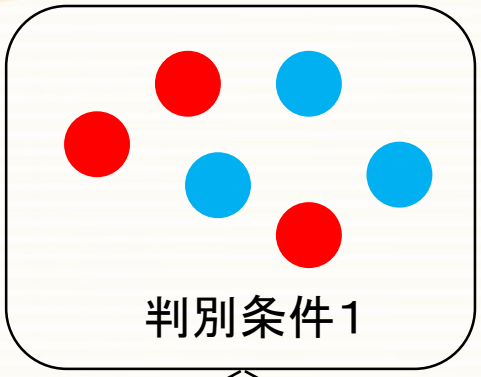
指針・方法：

判別条件によってデータの集合に分離される。その集合内で同じクラスのデータが集まるように判別条件を決める必要ある。同じクラスのデータがあるまっているかどうかは、エントロピー（平均情報量）という指標を用いて数値化する。エントロピーが小さいほど、同じクラスのデータが集まっていることを示唆し、エントロピーが小さくなるような判別条件を求めて、決定木を構築することを学習という。

# 決定木 (Decision Tree)

判別条件1の場合：

エントロピー＝1

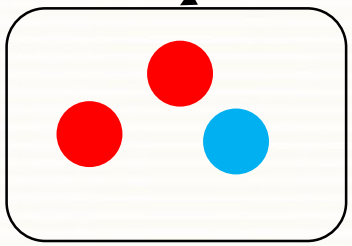


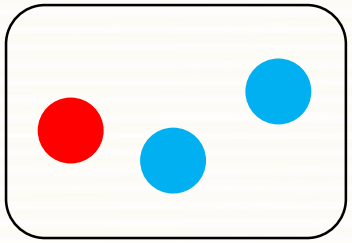

クラスAのデータ

クラスBのデータ

yes

no





エントロピー＝0.138

エントロピー＝0.138

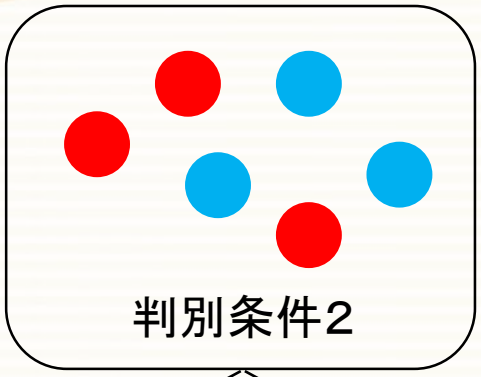
データ数の重みづけ平均エントロピー ＝ 3/6 × 0.138+3/6 × 0.138
＝ 0.138

この判別条件の利得（エントロピーの変化量） ＝ 1 − 0.138
＝0.862

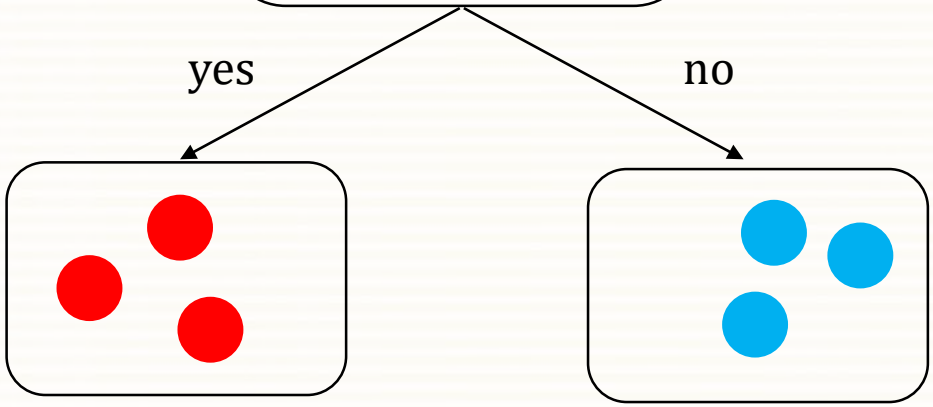
# 決定木 (Decision Tree)

判別条件2の場合：

エントロピー＝1








エントロピー＝0

エントロピー＝0

データ数の重みづけ　＝ 3/6×0+3/6×0

平均エントロピー　＝ 0

この判別条件の利得　＝ 1.0

判別条件1の利得 ＜ 判別条件2の利得 ➡ 判別条件2のほうが良い

# エントロピー（情報量）

発生する確率が低い出来事が分かったほうが、得られる情報は大きいと考える。

（すべて赤玉の箱から赤玉を引いて得られる情報は特にない。

赤玉と青玉が入った箱から赤玉を引くとある情報を得ることができる。）

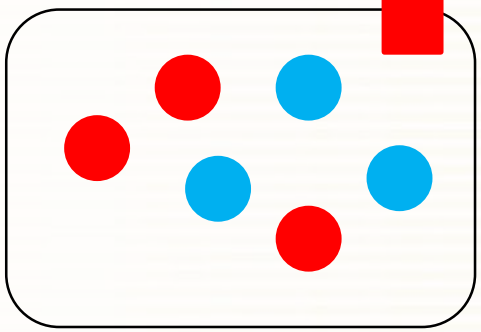
ある出来事が発生する確率 $p$ とすると、この出来事が有する情報量を

$$I = \log_2 \frac{1}{p} = -\log_2 p$$

と定義する。

# エントロピー（平均情報量）

赤玉3個、青玉3個入っている箱から玉を1個引くときに得られる情報量の平均値を考える。



$-\log_2 \frac{3}{6} = 1$　赤玉1個引く場合の情報量

$\frac{3}{6}$　赤玉1個引く確率

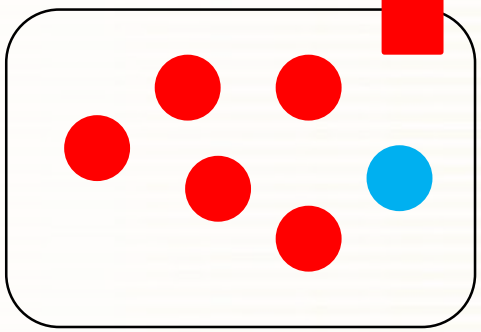
$-\frac{3}{6}\log_2 \frac{3}{6} - \frac{3}{6}\log_2 \frac{3}{6} = 1$　平均情報量

$-\log_2 \frac{3}{6} = 1$　青玉1個引く場合の情報量

$\frac{3}{6}$　青玉1個引く確率

# エントロピー（平均情報量）

赤玉3個、青玉3個入っている箱から玉を1個引くときに得られる情報量の平均値を考える。



$-\log_2 \frac{5}{6} = 0.263$　赤玉1個引く場合の情報量

$\frac{5}{6}$　赤玉1個引く確率

$-\frac{5}{6}\log_2 \frac{5}{6} - \frac{1}{6}\log_2 \frac{1}{6} = 0.650$　平均情報量

$-\log_2 \frac{1}{6} = 2581$　青玉1個引く場合の情報量

$\frac{1}{6}$　青玉1個引く確率

玉の数が均等の場合に比べて平均情報量が小さい

# エントロピー（平均情報量）

ある事象 $k$ が発生する確率 $p_k$ として、平均情報量は、

$$H = \sum_{k=1}^{K} p_k \log_2 \frac{1}{p_k} = -\sum_{k=1}^{K} p_k \log_2 p_k$$

として定義される。

$p_k = \frac{1}{K} \quad (k = 1,2,\cdots,K)$ の場合、平均情報量は最大値 $H = \log_2 K$ をとる。

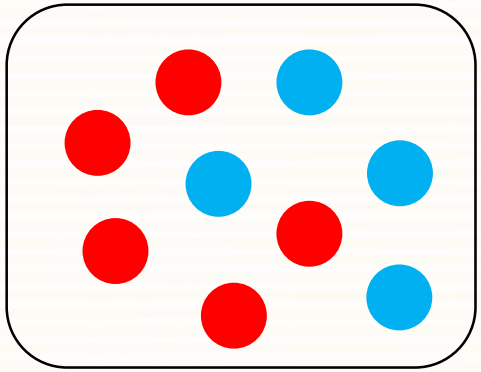
$p_k = 0 \ (k = 1,2,\cdots k-1, k+1, \cdots K)$ , $p_k = 1$の場合、

平均情報量は最小値 $H = 0$ をとる。

乱雑さが増えるほど平均情報量が大きくなることから、平均情報量をエントロピー

と呼ばれる。

# 決定木の学習



エントロピー

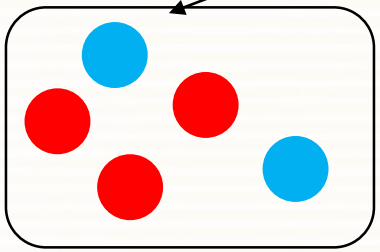
$$H = \sum_{k=1}^{K} p_k \log_2 \frac{1}{p_k}$$

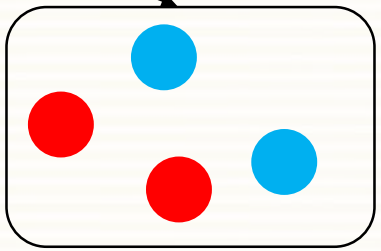
$$= -\sum_{k=1}^{K} p_k \log_2 p_k$$

ここで、$p_k = \frac{N_k}{N}$

yes　判別条件　no

$N$：全データ数、$N_k$：クラス$k$のデータ数





$$H^L = -\sum_{k=1}^{K} p_k^L \log_2 p_k^L$$

$$H^R = -\sum_{k=1}^{K} p_k^R \log_2 p_k^R$$

ここで、$p_k^{L(R)} = \frac{N_k^{L(R)}}{N^{L(R)}}$

$N^{L(R)}$：左(右)集合の全データ数、

$N_k^{L(R)}$：左（右）集合におけるクラス$k$のデータ数

# 決定木の学習

判別前のデータ集合のエントロピー　$H = -\sum_{k=1}^{K} p_k \log_2 p_k$

判別後のデータ集合のエントロピーの重みづけ平均　$\frac{N^L}{N} H^L + \frac{N^R}{N} H^R = -\frac{N^L}{N} \sum_{k=1}^{K} p_k^L \log_2 p_k^L - \frac{N^R}{N} \sum_{k=1}^{K} p_k^R \log_2 p_k^R$

判別前後でのエントロピーの増減（情報利得）　$\Delta H = H - \left( \frac{N^L}{N} H^L + \frac{N^R}{N} H^R \right)$

情報利得が大きい判別条件を見つける問題は、

判別後のデータ集合のエントロピーが最小の判別条件を見つける問題になる。

（∵判別条件を変えても$H$ は変化しない）

# 決定木の学習

2つの商品（商品A,B）のどちらか好きなほうを選択したとき、
ユーザーの性別および年齢情報を収集した。

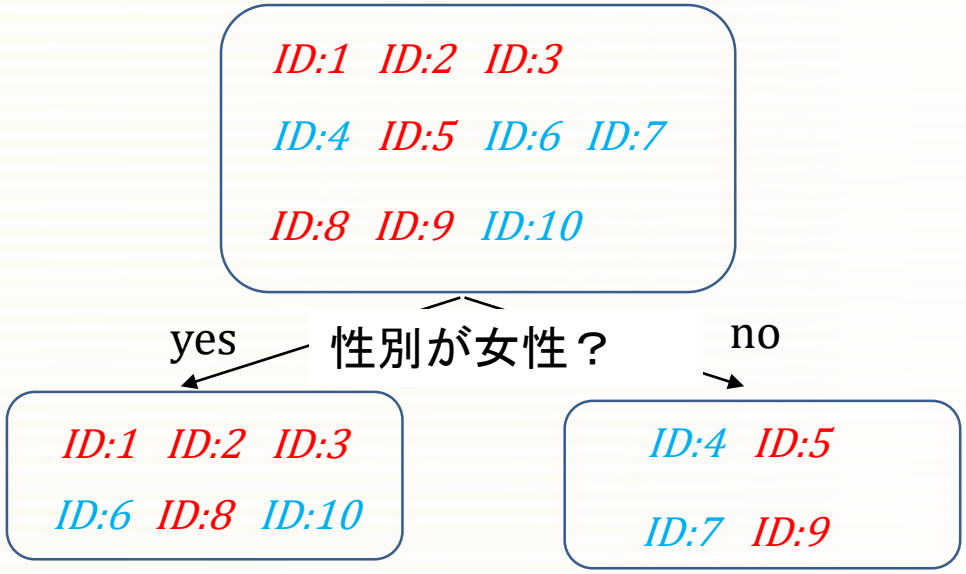
| ID | 性別 | 年齢 | 商品 |
|---|---|---|---|
| 1 | 女性 | 10代 | A |
| 2 | 女性 | 20代 | A |
| 3 | 女性 | 20代 | A |
| 4 | 男性 | 10代 | B |
| 5 | 男性 | 10代 | A |
| 6 | 女性 | 30代 | B |
| 7 | 男性 | 30代 | B |
| 8 | 女性 | 10代 | A |
| 9 | 女性 | 20代 | A |
| 10 | 男性 | 30代 | B |

# 決定木の学習

性別で判別・分類する場合：

10人分のデータのエントロピー：

$$-\frac{6}{10}\log_2\frac{6}{10}-\frac{4}{10}\log_2\frac{4}{10}=0.971$$




「性別が女性？」という条件で判別・分類した場合のエントロピー：

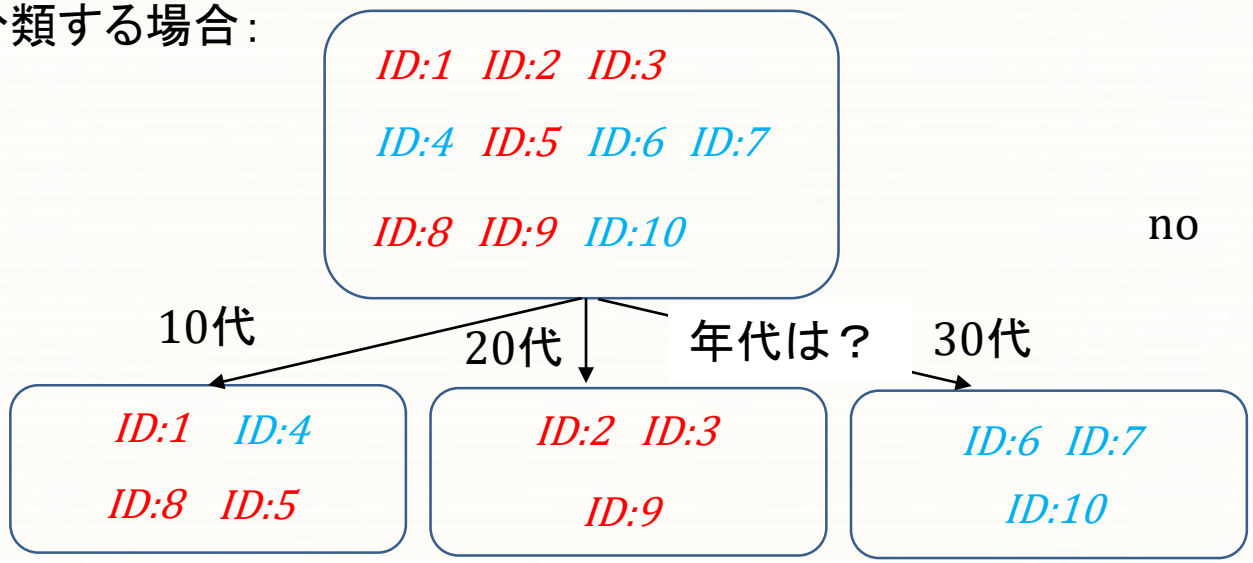
$$\frac{6}{10}\left(-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}\log_2\frac{2}{6}\right)+\frac{4}{10}\left(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4}\right)=0.951$$

情報利得は、$0.971-0.951=0.02$

# 決定木の学習

数式がやや細かいかもしれないので、以降読み飛ばしても構いません

年代で判別・分類する場合：




「年代は？」という条件で判別・分類した場合のエントロピー：

$$\frac{4}{10}\left(-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}\right)+\frac{3}{10}\left(-\frac{3}{3}\log_2\frac{3}{3}\right)+\frac{3}{10}\left(-\frac{3}{3}\log_2\frac{3}{3}\right)=0.325$$

情報利得は、$0.971 - 0.325 = 0.646$

性別判別による情報利得 $0.02$ < 年代による情報利得 $0.646$

したがって、一番の判別条件は年代を利用する

# アンサンブル学習

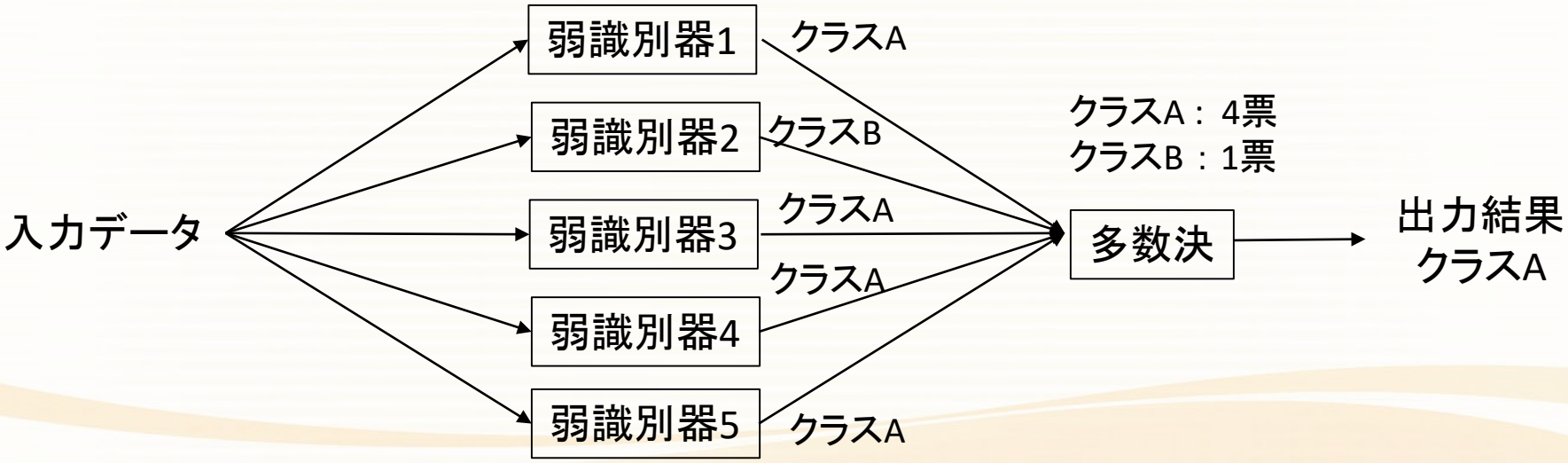
多数の弱識別器の多数決によって、識別性能の向上を図る方法論。

弱識別器（識別性能が低い）は、

・各識別器の計算は単純（計算負荷が小さい）

・ランダムに識別結果を返すよりかは性能がいい

を特徴とする。



# アンサンブル学習

2つの商品（商品A,B）のどちらか好きなほうを選択したとき、
ユーザーの性別および年齢情報を収集した。

| ID | 性別 | 年齢 | 商品 |
|---|---|---|---|
| 1 | 女性 | 11 | A |
| 2 | 女性 | 23 | A |
| 3 | 女性 | 24 | A |
| 4 | 男性 | 12 | B |
| 5 | 男性 | 15 | A |
| 6 | 女性 | 33 | B |
| 7 | 男性 | 37 | B |
| 8 | 女性 | 17 | A |
| 9 | 女性 | 21 | A |
| 10 | 男性 | 31 | B |

# アンサンブル学習

収集した情報をもとに、以下のような仮説に基づく弱識別器を設計する。

・弱識別器１
　女性かつ25才以下なら商品Aを選択
　それ以外の場合なら商品Bを選択

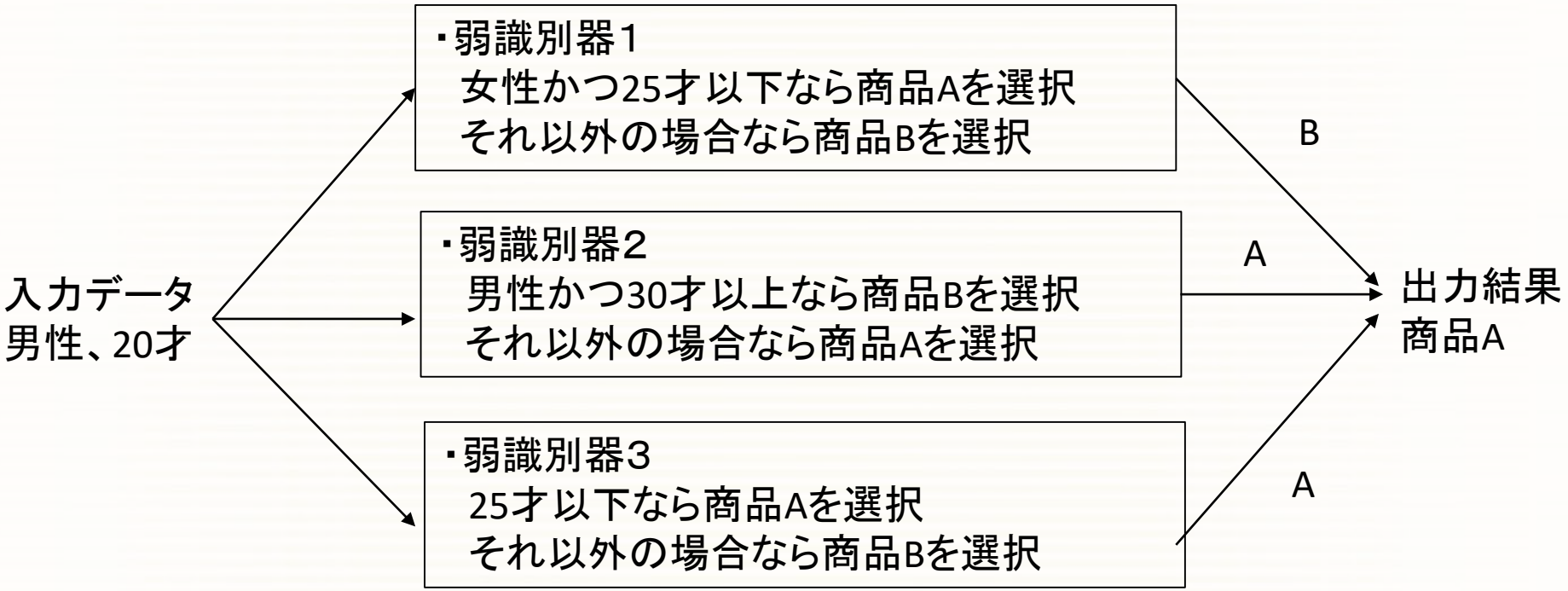
・弱識別器２
　男性かつ30才以上なら商品Bを選択
　それ以外の場合なら商品Aを選択

・弱識別器３
　25才以下なら商品Aを選択
　それ以外の場合なら商品Bを選択

男性、20才のユーザーはどちらの商品を選択するだろうか？

# アンサンブル学習

各弱識別器で、男性、20才のユーザーがどちらを選択するか考える。
その結果を多数決で統合して最終結果を計算する。




多数決からこのユーザーは商品Aを選択するであろうと予測する。

# ブースティング

アンサンブル学習の一種であり、その学習手順は以下のようになります。

【学習】
第１番目の弱識別器を作成する。
第１番目の弱識別器にて誤識別したデータに重みを置いて、第２番目の弱識別器を作成する。
第２番目の弱識別器にて誤識別したデータに重みを置いて、第３番目の弱識別器を作成する。

この手続きを以下繰り返す

第T-1番目の弱識別器にて誤識別したデータに重みを置いて、第T番目の弱識別器を作成する。

【識別】
作成した弱識別器の結果を統合して、入力データに対する識別結果を計算する。

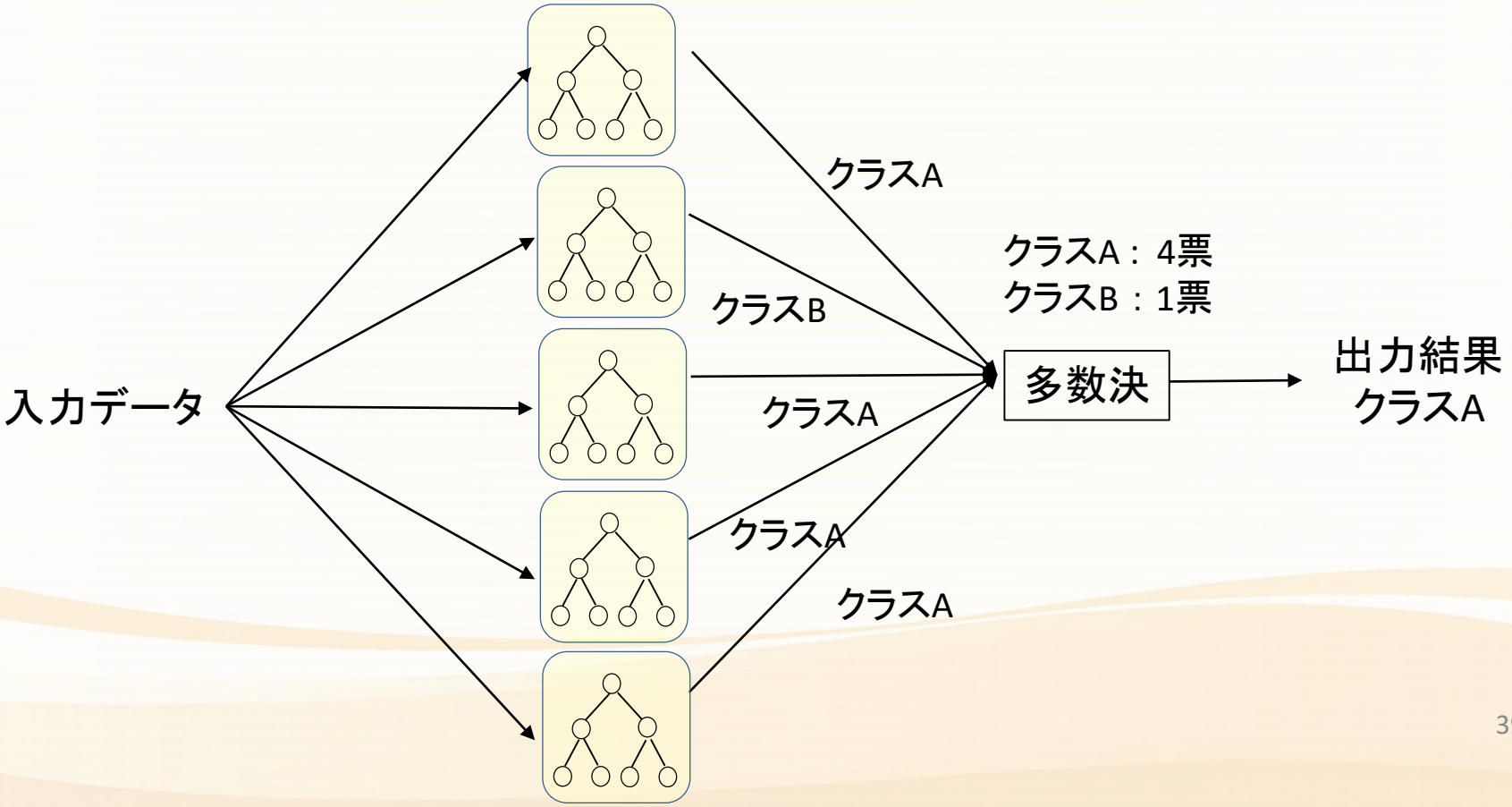
# ランダムフォレスト

アンサンブル学習における弱識別器に決定木を用いる方法。

学習データから無作為に選ばれたデータセットを用いて決定木を構築する。

別に無作為に選ばれたデータセットを用いて決定木を構築する。

これを繰り返して、複数個の決定木を統合した識別器を設計する。



# データサイエンス応用コース
# （変分ベイズ）

高野　渉
大阪大学

# 変分法

関数を変数に持つ関数を汎関数という。

関数 $y = f(x)$ を変数に持つ汎関数を $I[y]$ と記す。

汎関数は、独立変数 $x$ 、関数 $\mathrm{y}(x)$、およびその導関数 $y'(x)$ を変数に持つ関数 $F(x, y, y')$ の積分として表されることが多い。

$$I[y] = \int_{x_0}^{x_1} F(x, y(x), y'(x))dx$$

積分の形で与えられる汎関数を特に積分汎関数とよぶ。

$I[y]$ を最小もしくは最大にする関数 $\mathrm{y}(x)$ を求める問題を変分問題と呼び、それを解く方法を変分法という。

# 変分問題

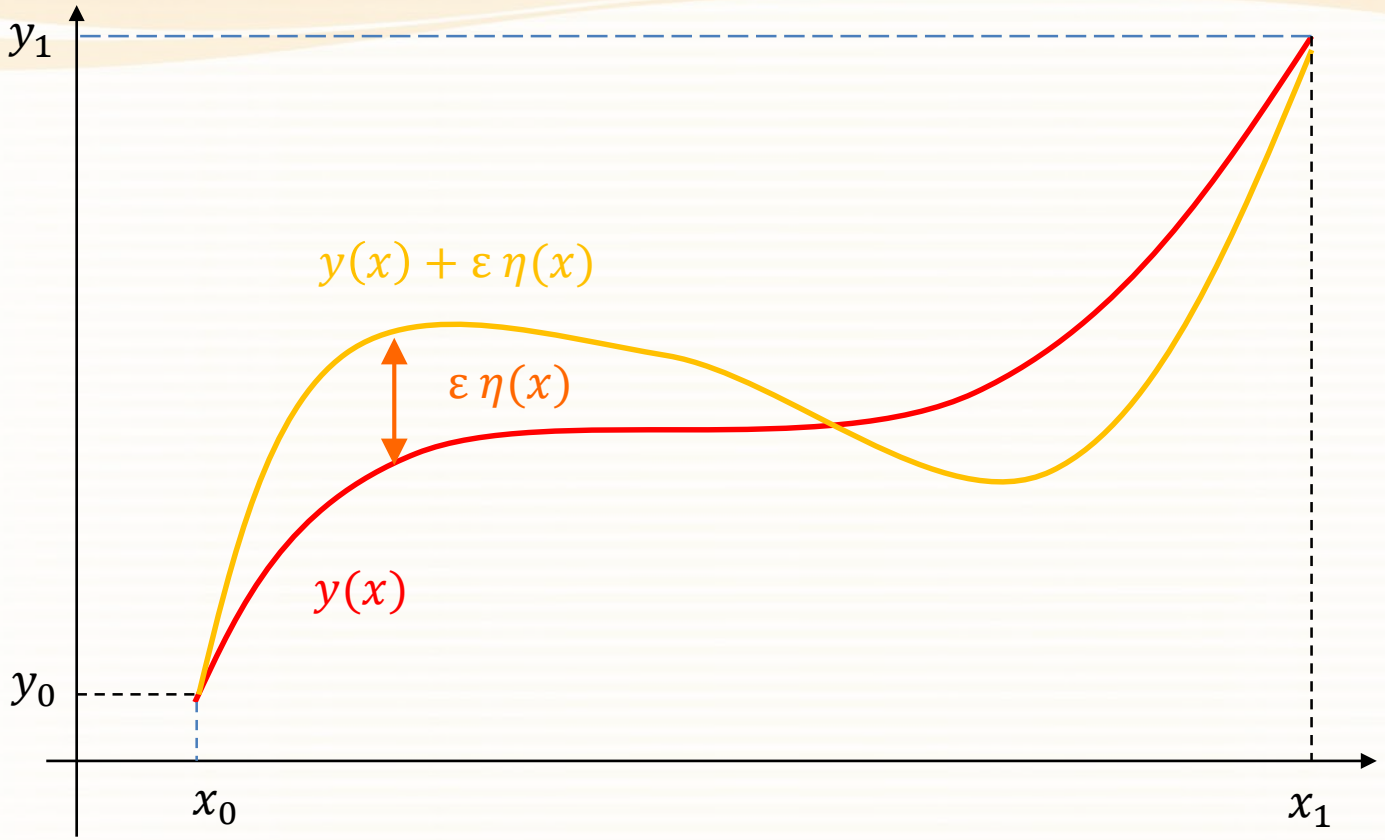
変分問題の典型的なものに、関数 $\mathrm{y} = \mathrm{y}(x)$ を両端$x = x_0, x_1$ における $\mathrm{y}(x)$ の値を固定するという条件のもとで、積分汎関数を最小もしくは最大にする問題がある。

$$I[y] = \int_{x_0}^{x_1} F(x, y(x), y'(x))dx$$

を最小もしくは最大にする $\mathrm{y}(x)$ をもとめよ。
ただし、以下の境界条件を満足するものとする。

$\mathrm{y}(x_0) = y_0,\quad \mathrm{y}(x_1) = y_1$ （$y_0, y_1$ は与えられている）

# オイラー･ラグランジュ方程式




最適な関数 $\mathrm{y}=\mathrm{y}(x)$

これを境界条件のもとで少しだけずらした関数 $y_\varepsilon=\mathrm{y}(x)+\varepsilon\,\eta(x)$

ここで、$\varepsilon$は微小パラメータ、$\eta(x)$は任意関数

境界条件を満たすことから、$\eta(x_0)=\eta(x_1)=0$ である。

# オイラー・ラグランジュ方程式

$I[y]$ に $y_\varepsilon$ を代入すると

$$I[y_\varepsilon] = \int_{x_0}^{x_1} F(x, y_\varepsilon(x), y_\varepsilon'(x))dx$$

$$= \tilde{I}(\varepsilon) \quad （\varepsilon の関数とみなす）$$

$\varepsilon = 0$ のときに $\tilde{I}(\varepsilon)$ は最小となる。
すなわち、一次の最適性条件を満足することになる。

$$\left.\frac{d\tilde{I}(\varepsilon)}{d\varepsilon}\right|_{\varepsilon = 0} = 0$$

# オイラー・ラグランジュ方程式

$$\frac{d\tilde{I}(\varepsilon)}{d\varepsilon} = \frac{d}{d\varepsilon}\int_{x_0}^{x_1} F(x, y_\varepsilon(x), {y_\varepsilon}'(x))dx$$

$$= \int_{x_0}^{x_1} \left( \frac{\partial F}{\partial y_\varepsilon}\frac{\partial y_\varepsilon}{\partial \varepsilon} + \frac{\partial F}{\partial {y'}_\varepsilon}\frac{\partial {y'}_\varepsilon}{\partial \varepsilon} \right) dx$$

$$\begin{cases} y_\varepsilon(x) = y(x) + \varepsilon\,\eta(x) \\ {y'}_\varepsilon(x) = y'(x) + \varepsilon\,\eta'(x) \end{cases}$$

なので、上式は以下となる

$$\frac{d\tilde{I}(\varepsilon)}{d\varepsilon} = \int_{x_0}^{x_1} \left( \frac{\partial F}{\partial y_\varepsilon}\eta(x) + \frac{\partial F}{\partial {y'}_\varepsilon}\eta'(x) \right) dx$$

# オイラー・ラグランジュ方程式

$\varepsilon = 0$ とおくと、

$$\frac{d\tilde{I}(0)}{d\varepsilon} = \int_{x_0}^{x_1} \left( \frac{\partial F}{\partial y}\eta(x) + \frac{\partial F}{\partial y'}\eta'(x) \right) dx = 0$$

第2項を式変形する。

$$\int_{x_0}^{x_1} \frac{\partial F}{\partial y'}\eta'(x)dx = \left[ \frac{\partial F}{\partial y'}\eta(x) \right]_{x_0}^{x_1} - \int_{x_0}^{x_1} \frac{d}{dx}\left( \frac{\partial F}{\partial y'} \right) \eta(x)\, dx$$

これを上式に代入すると、

# オイラー・ラグランジュ方程式

$$\frac{d\tilde{I}(0)}{d\varepsilon} = \left[\frac{\partial F}{\partial y'}\eta(x)\right]_{x_0}^{x_1} + \int_{x_0}^{x_1}\left[\frac{\partial F}{\partial y}\eta(x) - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right)\eta(x)\right] dx$$

$\eta(x_0) = \eta(x_1) = 0$ なので、
この部分はゼロとなる。

$$= \int_{x_0}^{x_1}\left[\frac{\partial F}{\partial y} - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right)\right]\eta(x)\, dx$$

$$= 0$$

# オイラー・ラグランジュ方程式

任意の関数 $\eta(x)$に対して上式がなりたつので、
解 $y(x)$ は以下の関係式を満足しなければならない。

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right) = 0$$

これをオイラー・ラグランジュ方程式と呼ぶ。
この解を汎関数の停留関数と呼ぶ。

関数が汎関数を最小もしくは最大にするには、$y(x)$ が
停留関数、すなわちオイラー・ラグランジュ方程式の解で
なければならない。

# 変分ベイズ法

観測データ$X = \{x_1, x_2, \cdots, x_N\}$の周辺尤度$P(X)$を最大にする隠れ変数$z$, パラメータ$\theta$の分布を求める。

$$\ln P(X) = \ln \int P(X, z, \theta)\, dz\, d\theta$$

$$= \ln \int q(z, \theta) \frac{P(X, z, \theta)}{q(z, \theta)}\, dz\, d\theta$$

Jensenの不等式より

$$\geq \int q(z, \theta) \ln \frac{P(X, z, \theta)}{q(z, \theta)}\, dz\, d\theta$$

$$= L(q(z, \theta)) \quad \text{対数尤度下限}$$

# 変分ベイズ法

$$\ln P(X) - L\big(q(z,\theta)\big)$$

$$= \ln P(X) - \int q(z,\theta) \ln \frac{P(X,z,\theta)}{q(z,\theta)}\, dz\, d\theta$$

$$= \ln P(X) - \int q(z,\theta) \ln \frac{P(X)P(z,\theta|X)}{q(z,\theta)}\, dz\, d\theta$$

$$= \ln P(X) - \boxed{\int q(z,\theta) \ln P(X)\; dz\, d\theta} \quad = \ln P(X)$$

$$- \int q(z,\theta) \ln \frac{P(z,\theta|X)}{q(z,\theta)}\, dz\, d\theta$$

$$= \int q(z,\theta) \ln \frac{q(z,\theta)}{P(z,\theta|X)}\, dz\, d\theta$$

# 変分ベイズ法

$$= KL\big(q(z,\theta)||P(z,\theta|X)\big)$$

これは、分布$q(z,\theta)$と分布$P(z,\theta|X)$の非類似度を表すKullback Leibler 情報量である。

$$\ln P(X) = L\big(q(z,\theta)\big) + KL\big(q(z,\theta)||P(z,\theta|X)\big)$$

周辺対数尤度は、対数尤度下限とKullback Leibler 情報量の和である。$q(z,\theta)$を変化させても周辺対数尤度は変化しない。対数尤度下限とKullback Leibler 情報量の比率が変化する。

Kullback Leibler 情報量を最小化する分布$q(z,\theta)$を推定することと、対数尤度下限を最大化する$q(z,\theta)$を推定することは等価

# 変分ベイズ法

$$\underset{q(z,\theta)}{\text{maximize}}\ L\big(q(z,\theta)\big)$$

$$\left[\underset{q(z,\theta)}{\text{maximize}} \int q(z,\theta)\ln\frac{P(X,z,\theta)}{q(z,\theta)}\,dz\,d\theta\right]$$

隠れ変数 $z$ とパラメータ $\theta$ が独立と仮定すると、$q(z,\theta)=q(z)q(\theta)$

$$\underset{q(z)q(\theta)}{\text{maximize}} \int q(z)q(\theta)\ln\frac{P(X,z,\theta)}{q(z)q(\theta)}\,dz\,d\theta$$

# 変分ベイズ法

隠れ変数 $z$ の分布 $q(z)$ に的を絞って、

$$\underset{q(z)}{\text{maximize}} \int q(z)q(\theta) \ln \frac{P(X,z,\theta)}{q(z)q(\theta)} \, d\theta \, dz$$

これは以下の汎関数とみなすことができる。

$$\int F(z, q(z)) \, dz$$

$$F(z, q(z)) = \int q(z)q(\theta) \ln \frac{P(X,z,\theta)}{q(z)q(\theta)} \, d\theta$$

# Jensenの不等式(補足)

【補足】 Jensenの不等式

関数$f(x)$は凸関数とする.

$\lambda_1 + \lambda_2 + \cdots + \lambda_n = 1$ を満足する $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$に対して

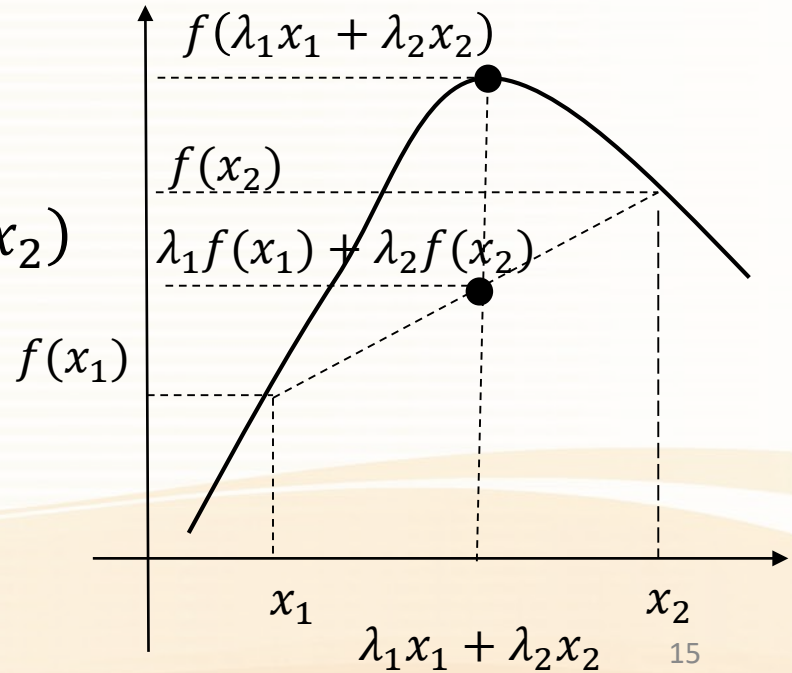
$$\lambda_1 f(x_1) + \lambda_2 f(x_2) + \cdots + \lambda_n f(x_n) \leq f(\lambda_1 x_1 + \lambda_2 x_2 + \cdots + \lambda_n x_n)$$

が成立する.

∵ $n = 2$ の場合, 凸関数の定義より

$$\lambda_1 f(x_1) + \lambda_2 f(x_2) \leq f(\lambda_1 x_1 + \lambda_2 x_2)$$

が成り立つ.



# Jensenの不等式(補足)

$n = k$ の場合,

$$\lambda_1 f(x_1) + \lambda_2 f(x_2) + \cdots + \lambda_k f(x_k) \leq f(\lambda_1 x_1 + \lambda_2 x_2 + \cdots + \lambda_k x_k)$$

が成り立つとする.

$n = k + 1$ の場合,

$$\text{左辺} = \lambda_1 f(x_1) + \lambda_2 f(x_2) + \cdots + \lambda_k f(x_k) + \lambda_{k+1} f(x_{k+1})$$

$$= (\lambda_1 + \lambda_2 + \cdots + \lambda_k)$$

$$\times \left\{ \frac{\lambda_1}{\lambda_1 + \lambda_2 + \cdots + \lambda_k} f(x_1) + \cdots + \frac{\lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k} f(x_k) \right\}$$

$$+ \lambda_{k+1} f(x_{k+1})$$

# Jensenの不等式（補足）

$$= (\lambda_1 + \lambda_2 + \cdots + \lambda_k)\big(\alpha_1 f(x_1) + \cdots + \alpha_k f(x_k)\big) + \lambda_{k+1} f(x_{k+1})$$

ただし，
$\alpha_1 + \alpha_2 + \cdots + \alpha_k = 1$ ，$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$ を満たす．

$n = k$ の場合，

$$\alpha_1 f(x_1) + \alpha_2 f(x_2) + \cdots + \alpha_k f(x_k) \leq f(\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k)$$

満たすのであるから，

$$\text{左辺} \leq (\lambda_1 + \lambda_2 + \cdots + \lambda_k)\big(f(\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k)\big) + \lambda_{k+1} f(x_{k+1})$$

$n = 2$ の場合の関係式を適用すると

# Jensenの不等式（補足）

左辺$\leq f\left((\lambda_1+\lambda_2+\cdots+\lambda_k)(\alpha_1x_1+\alpha_2x_2+\cdots+\alpha_kx_k)+\lambda_{k+1}x_{k+1}\right)$

$$(\lambda_1+\lambda_2+\cdots+\lambda_k)\alpha_i=(\lambda_1+\lambda_2+\cdots+\lambda_k)\left(\frac{\lambda_i}{\lambda_1+\lambda_2+\cdots+\lambda_k}\right)$$
$$=\lambda_i$$

左辺$\leq f(\lambda_1x_1+\lambda_2x_2+\cdots+\lambda_kx_k+\lambda_{k+1}x_{k+1})$

$n=k+1$の場合も成り立つから，数学的帰納法により，Jensenの不等式が成り立つことがわかる.

# 変分ベイズ法

この積分汎関数の最大化は、変分問題であるので、
最適分布 $q(z)$ は、以下のオイラー・ラグランジュ方程式を
満足することになる。

$$\frac{\partial F}{\partial q(z)} - \frac{d}{dz}\left(\frac{\partial F}{\partial q'(z)}\right) = 0$$

$F(z, q(z))$は$q'(z)$を陽に含んでいないので、

$$\frac{\partial F}{\partial q(z)} = 0$$

この方程式を解けばいい。

# 変分ベイズ法

$$\frac{\partial F}{\partial q(z)} = \frac{\partial}{\partial q(z)} \int q(z)q(\theta) \ln \frac{P(X,z,\theta)}{q(z)q(\theta)} \, d\theta$$

$$= -\frac{\partial}{\partial q(z)} \int q(z)q(\theta) \ln \frac{q(z)q(\theta)}{P(X,z,\theta)} \, d\theta$$

$$= -\int q(\theta) \ln \frac{q(z)q(\theta)}{P(X,z,\theta)} \, d\theta$$

$$-\int q(z)q(\theta) \frac{P(X,z,\theta)}{q(z)q(\theta)} \frac{q(\theta)}{P(X,z,\theta)} \, d\theta$$

# 変分ベイズ法

$$= -\int q(\theta) \ln q(z)\ d\theta - \int q(\theta) \ln \frac{q(\theta)}{P(X, z, \theta)} d\theta - \int q(\theta) d\theta$$

$$= -\ln q(z) - \int q(\theta) \ln \frac{q(\theta)}{P(X, z, \theta)} d\theta - 1$$

$$= 0$$

したがって、$q(z)$は以下となる

$$\ln q(z) = \int q(\theta) \ln \frac{P(X, z, \theta)}{q(\theta)} d\theta - 1$$

# 変分ベイズ法

$$q(z) = exp\left( \int q(\theta) \ln \frac{P(X, z, \theta)}{q(\theta)} d\theta \right) \exp(-1)$$

$$q(z) \propto exp\left( \int q(\theta) \ln \frac{P(X, z, \theta)}{q(\theta)} d\theta \right)$$

同様にパラメータの分布$q(\theta)$は以下となる

$$q(\theta) \propto exp\left( \int q(z) \ln \frac{P(X, z, \theta)}{q(z)} dz \right)$$

# 変分ベイズ法

アルゴリズム:

Step1 : $q(z), q(\theta)$ を初期化する

Step2 : $q(\theta)$ を使って、$q(z)$を更新する

Step3 : $q(z)$ を使って、$q(\theta)$を更新する

Step2とStep3を交互に繰り返す。

# データサイエンス応用コース
# ニューラルネットワーク I･II

（教師あり学習 /教師なし学習）

下川　和郎
大阪大学

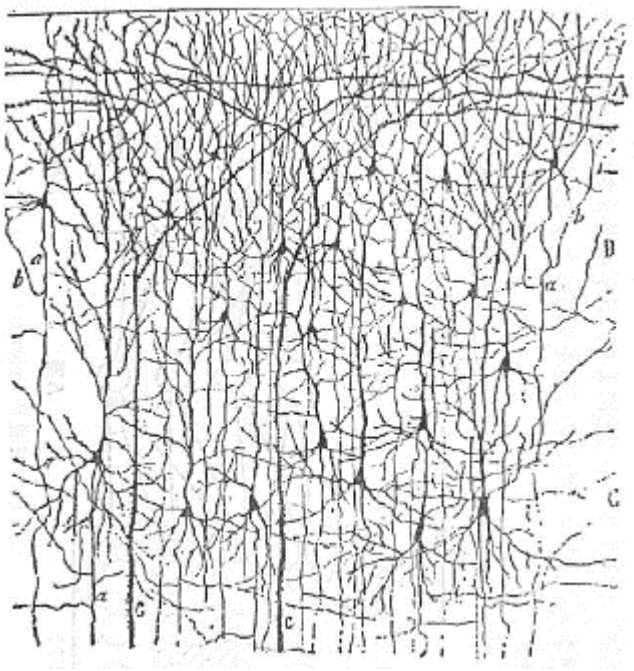
# ニューラルネットワークの構造

脳の神経回路網

Golgi 染色によって、脳の神経回路網が多数の神経細胞の結合からできていることが明確になり

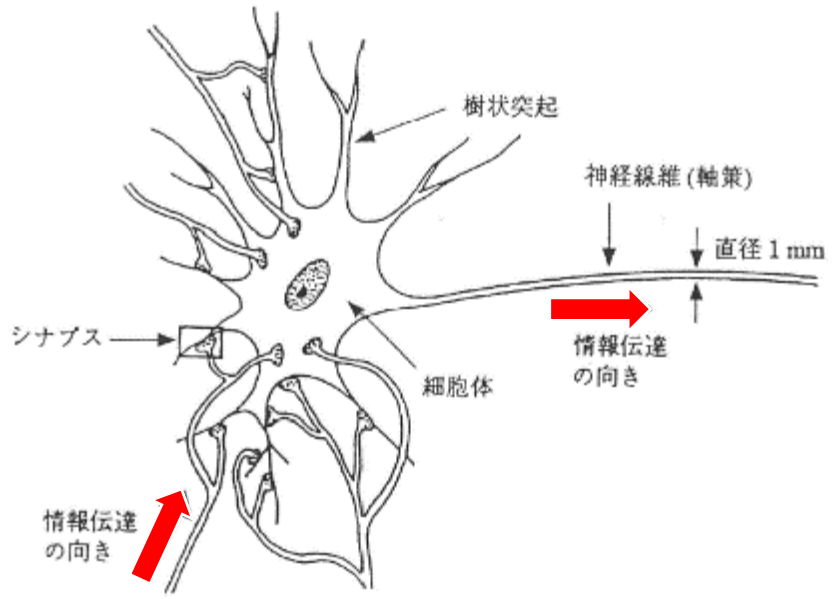
Golgi と Cajal が共にノーベル賞を受賞（1906）.

Golgi ：神経線維説
Cajal：ニューロン説

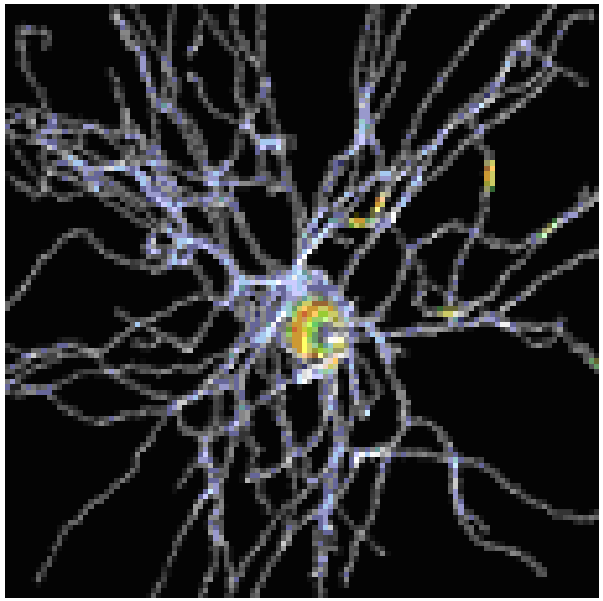


S.R.Cajal (1911)

# ニューラルネットワークの構造




神経細胞の実際



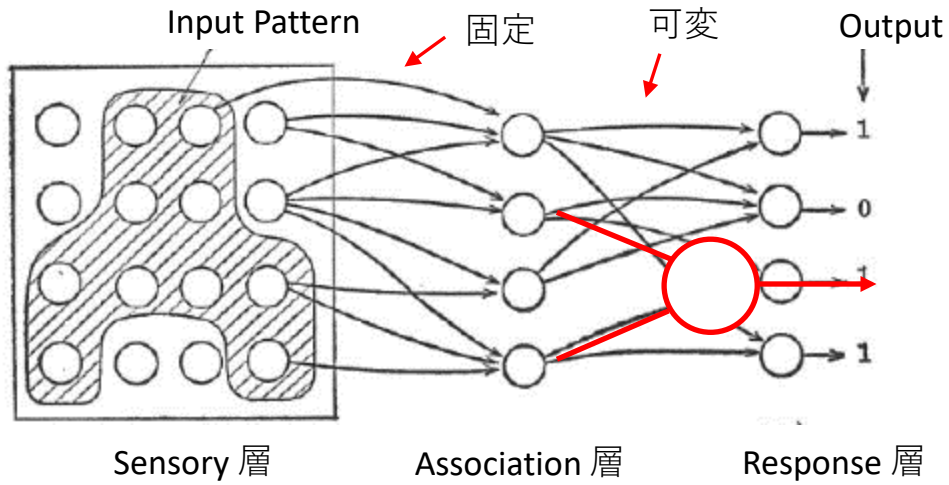
実際の神経細胞に計算上の電気的活動をあてはめた例

「神経システムの大規模シミュレーションを目指して」村木茂, 下川和郎 Bit 2001 3月号　共立出版 より

# パーセプトロン

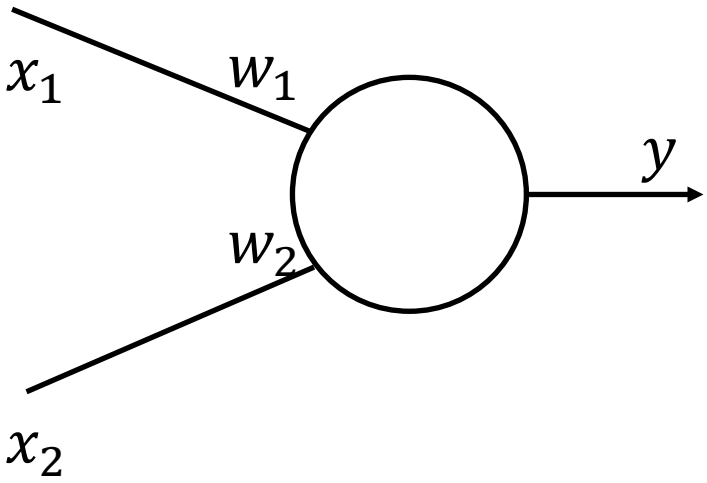
パーセプトロン
Rosenblatt (1961)

オリジナルは三層構造だが、学習できる場所は出力層の手前だけなので単層パーセプトロンとも呼ばれる.




多層化により能力が向上.
Rumelhart, McClelland & the PDP Research Group (1961)

# パーセプトロン




$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

AND

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

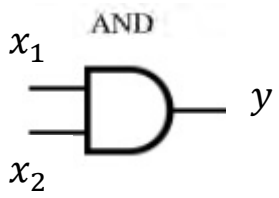
# パーセプトロン

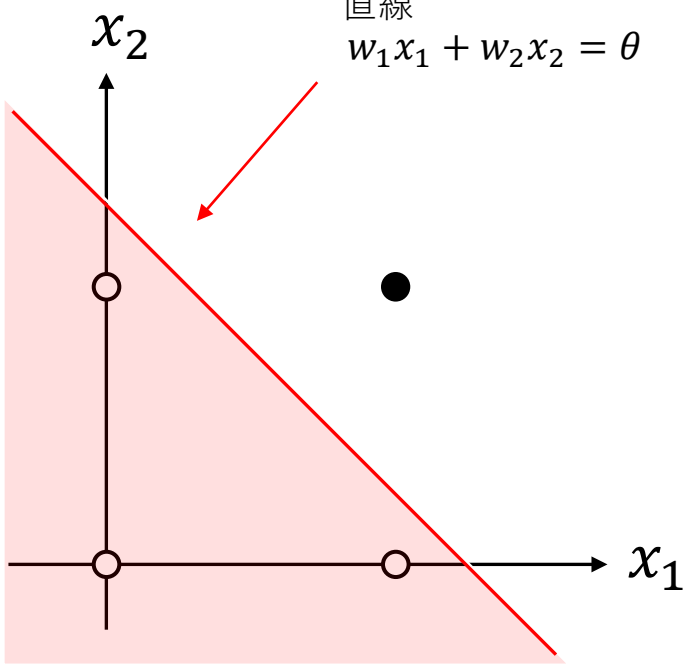
例えば $\begin{cases} w_1 = 1.25 \\ w_2 = 1.25 \\ \theta = 1.5 \end{cases}$

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

AND

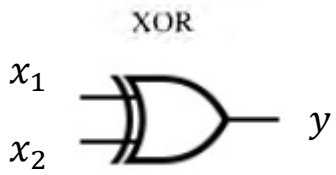
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |



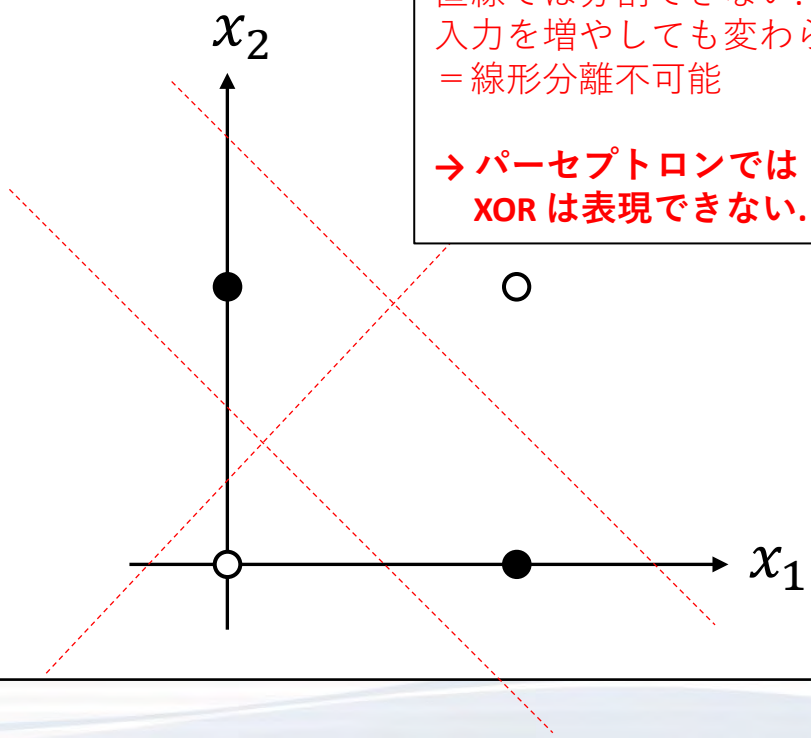

直線
$w_1x_1 + w_2x_2 = \theta$

# パーセプトロン




XOR

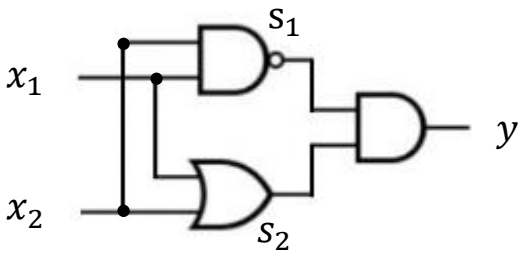
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |




直線では分割できない.
入力を増やしても変わらない.
＝線形分離不可能

**→ パーセプトロンでは XOR は表現できない.**

# 論理素子を組み合わせる




三種の論理素子を組み合わせる事で必要となる XOR が出力される.

これらの接続を多数行って様々な機能を実現することが可能.

必要な出力.

| $x_1$ | $x_2$ | NAND $s_1$ | OR $s_2$ | AND |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |

# 組み合わせの発想

- 組み合わせて他の論理回路を作ることが可能
  - ex. NAND 回路 4 個で XOR
  - ex. NOR 回路 5 個で XOR
  - （構成は動作速度、消費電力、実装面積、利用環境での信頼性で決まる）
- 論理回路の組み合わせでコンピュータを作ることが可能
- パーセプトロンの組み合わせでコンピュータを作ることも

# アポロ計画で使われたコンピュータ

NOR


| $x_1$ | $x_2$ | OR | NOR $y$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |

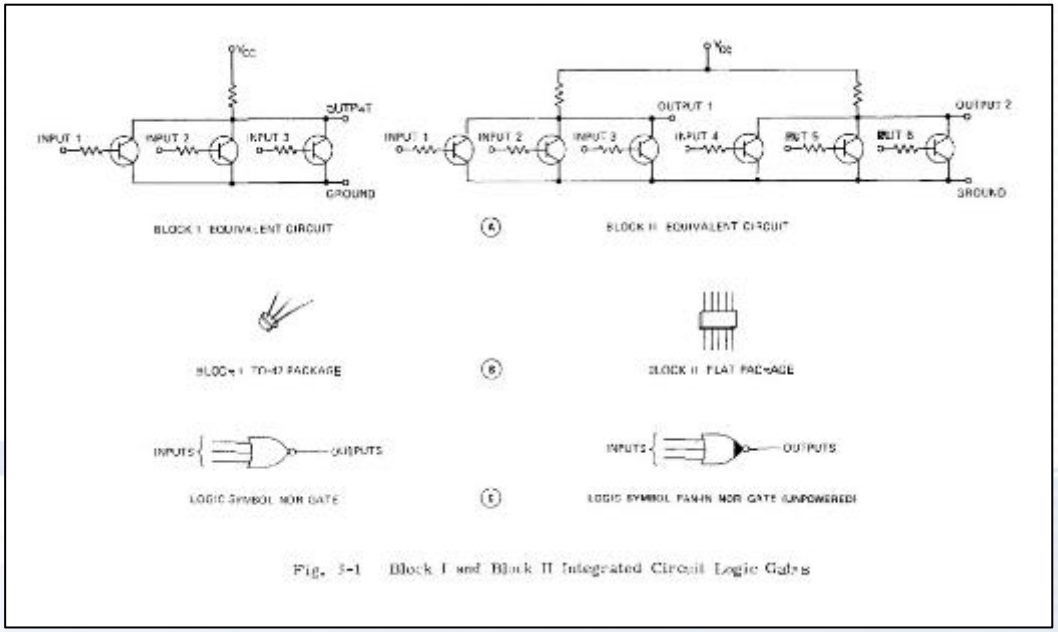


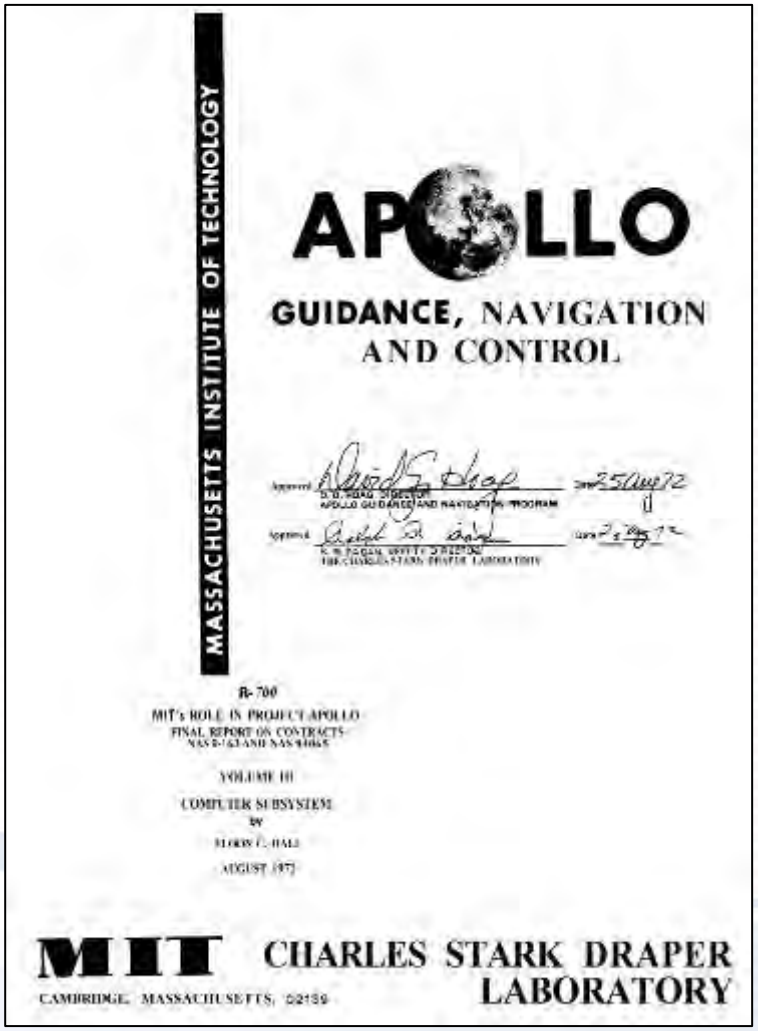

Fig. 3-1 Block I and Block II Integrated Circuit Logic Gates

全て 3 入力 NOR 回路の
組み合わせで作られた.



NASA Office of Logic Design より http://klabs.org/history/ech/agc_schematics/

# アポロ計画で使われたコンピュータ


MASSACHUSETTS INSTITUTE OF TECHNOLOGY

APOLLO

GUIDANCE, NAVIGATION AND CONTROL

R-700

MIT's ROLE IN PROJECT APOLLO

FINAL REPORT ON CONTRACTS NAS9-153 AND NAS9-1065

VOLUME III

COMPUTER SUBSYSTEM

BY

AUGUST 1972

MIT CHARLES STARK DRAPER LABORATORY

CAMBRIDGE, MASSACHUSETTS 02139

DSKY

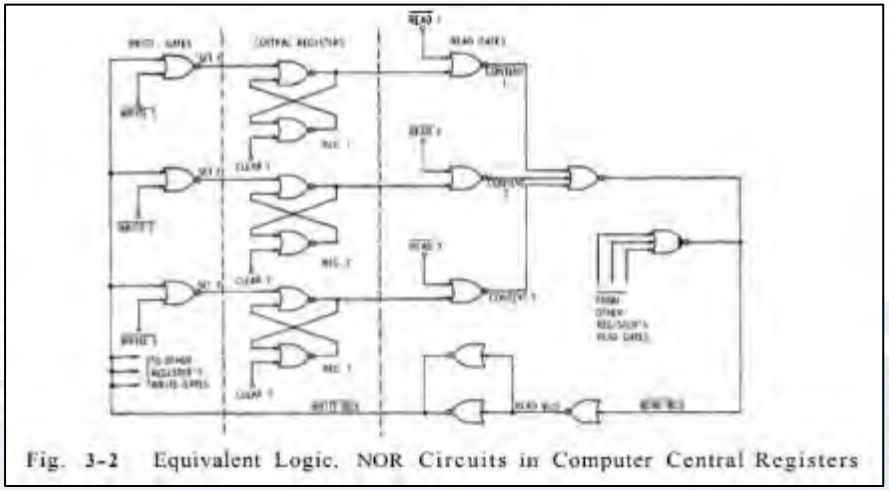
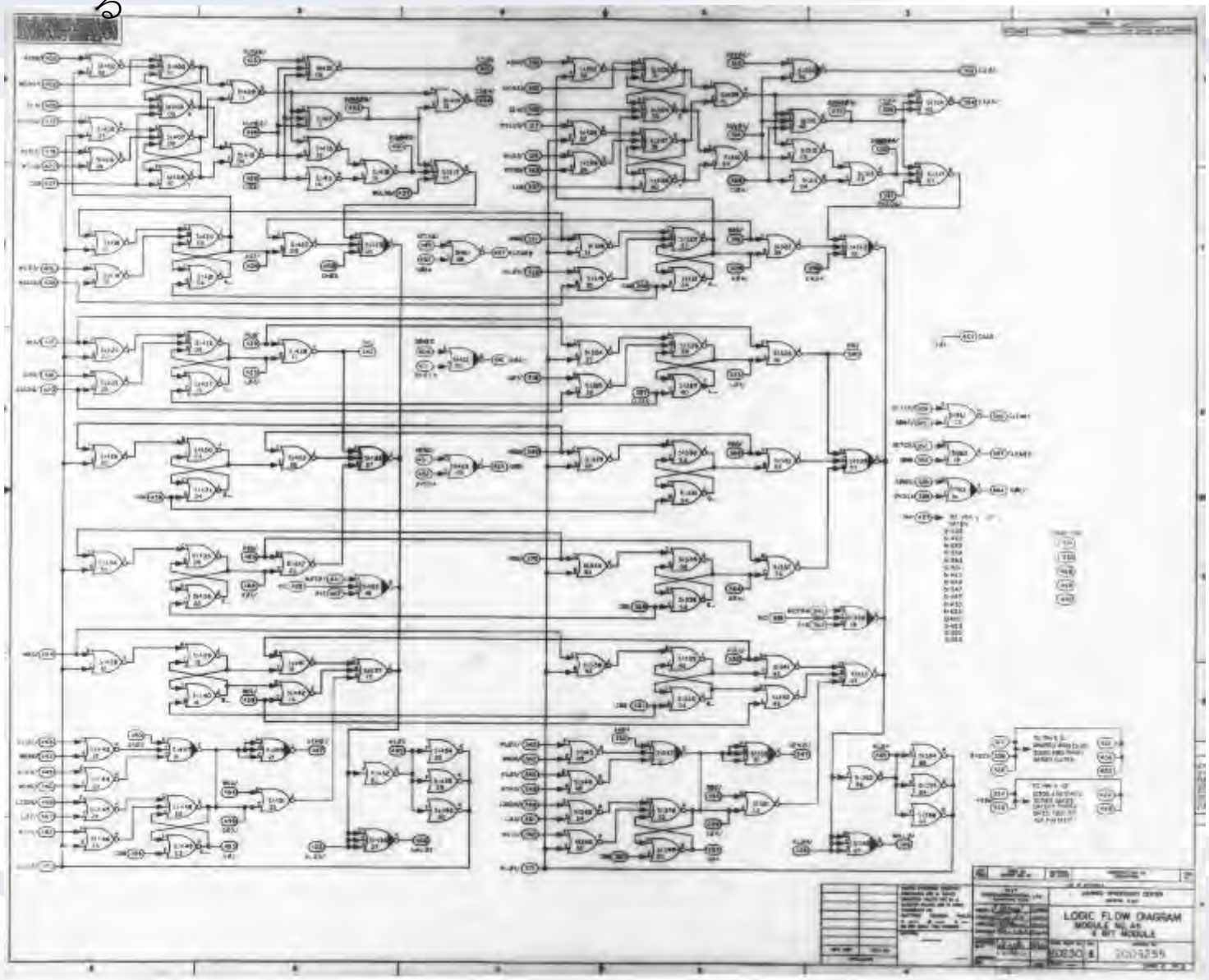


Fig. 3-2 Equivalent Logic, NOR Circuits in Computer Central Registers



Apollo 11 mission's 40th Anniversary: One large step for open source code... より

回路全体は 3 入力 NOR 回路の組み合わせで作成されている



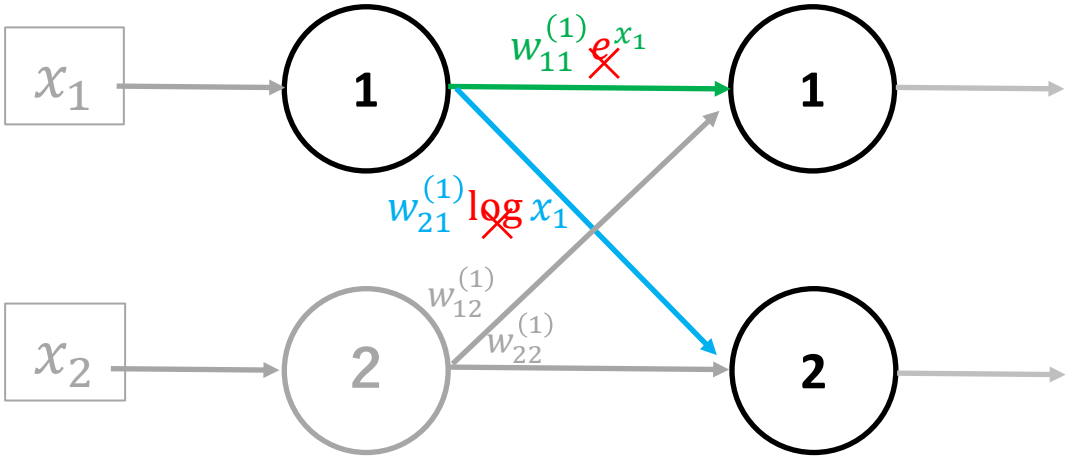

NASA Office of Logic Design より　http://klabs.org/history/ech/agc_schematics/

# NNモデルのポイント（1）

素子間の情報伝達は線形： ex. $w_{11}^{(1)}x_1, w_{21}^{(1)}x_1$

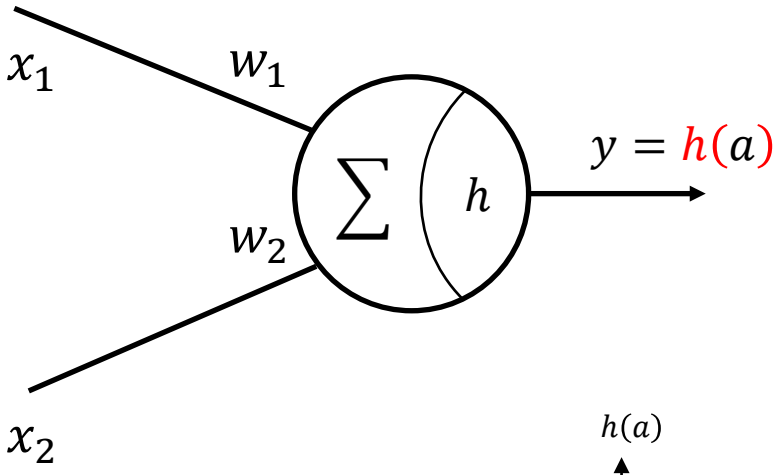
情報伝達には非線形関数を用いない. ex. $\left(w_{11}^{(1)}\cancel{e^{x_1}}, w_{21}^{(1)}\cancel{\log} x_1\right)$

→ モデルの前提条件. 非線形を許すと
1 層でも XOR を実現することができそう
だが、自由度が高過ぎて議論が難しくなる.

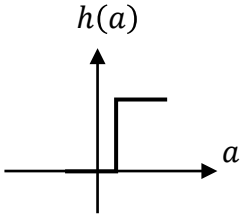


# NNモデルのポイント（2）

活性化関数は非線形関数 $h(\ )$ を用いる.




$$\begin{cases} a = b + w_1 x_1 + w_2 x_2 \\ y = h(a) \end{cases}$$



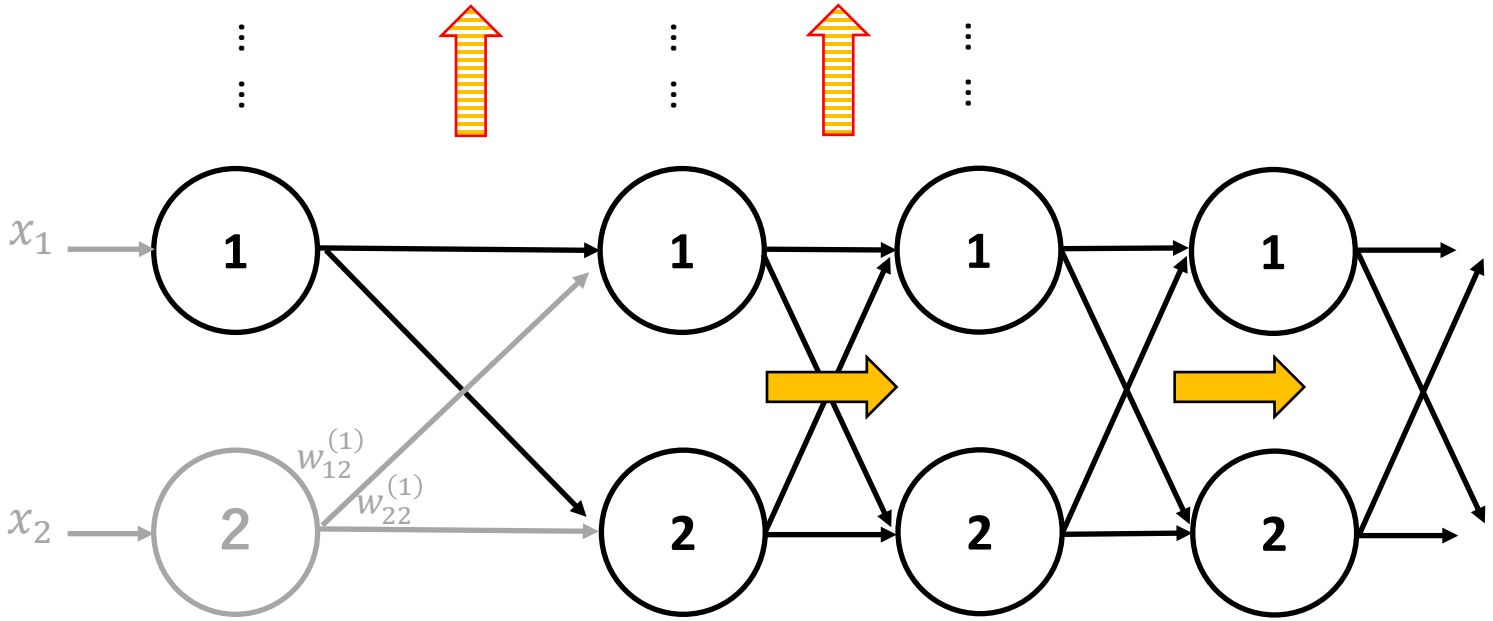

活性化関数に非線形関数を用いることで、多層にした時に線形分離不可能な例題（XOR) が解けるようになる.

パーセプトロンではステップ関数を使っていたので多層化によってXOR 例題を解くことができた.

# NNモデルのポイント（3）

## NN モデルの発展の方向性.

2 層のパーセプトロンによって任意の関数が表現可能。 しかし NN モデルは単純な仕組みのまま層を増やす方向で複雑化に対応していく.

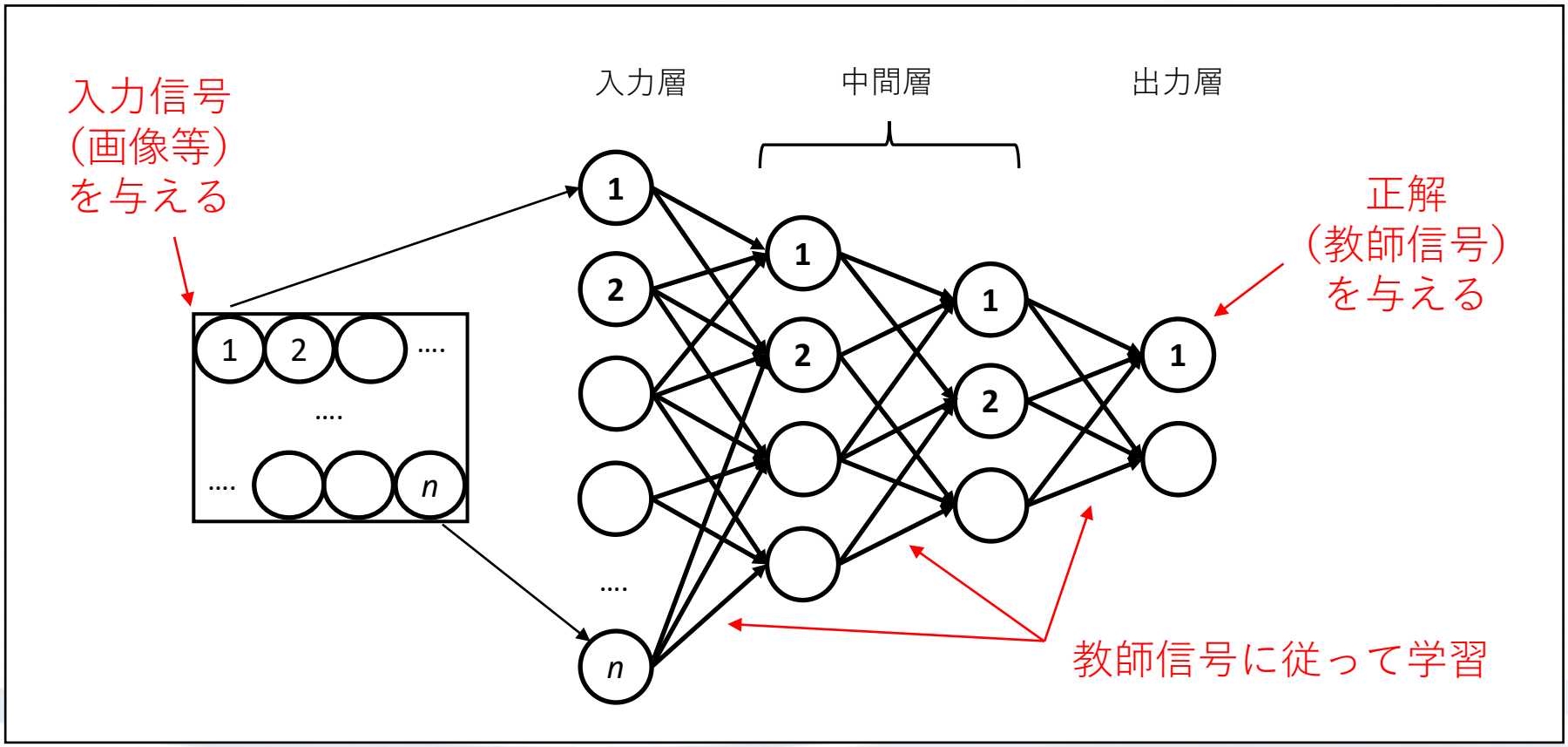


# 教師あり学習と教師なし学習

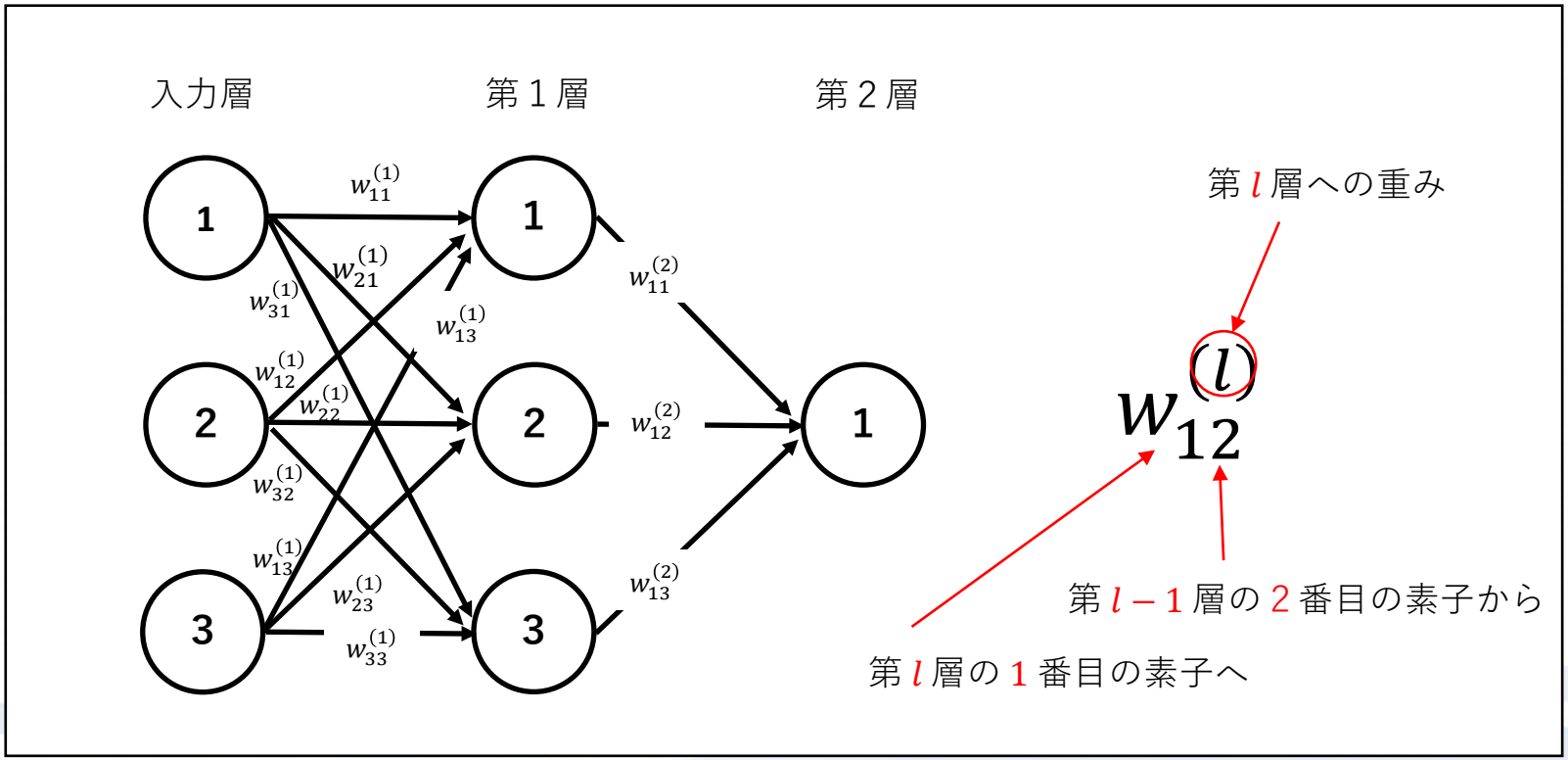
- 教師あり学習
  - ラベルデータを用いて学習
  - ラベルに基づいてデータ分類を行う
  - 未知のデータがどのグループに属するかを導出
- 教師なし学習
  - ラベルなしデータを用いる
  - データが内包している特徴量を自動学習
  - クラスタリング、オートエンコーダー、事前学習

# 教師あり学習

# ニューラルネットワーク： 教師あり学習

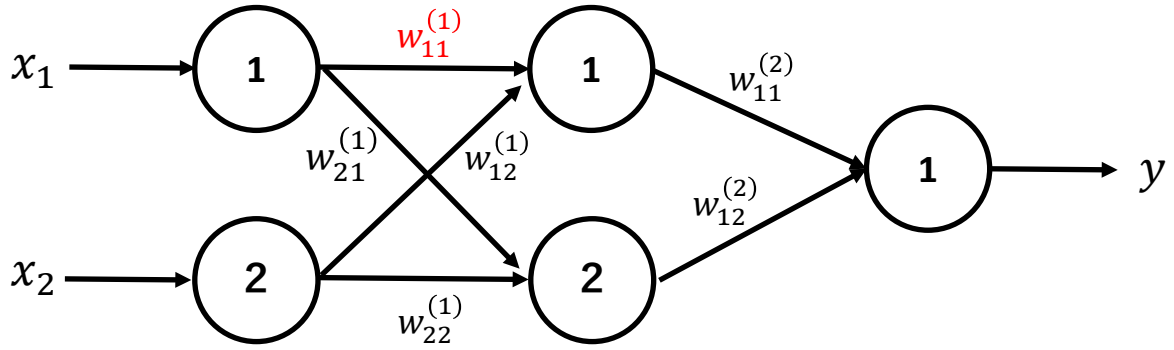


# NW層・素子間結合の記述



# 学習（勾配法）

簡単のため三層の簡単なネットワークで学習する場合を考える




$$\boldsymbol{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \end{bmatrix} \qquad \boldsymbol{b}^{(2)} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$

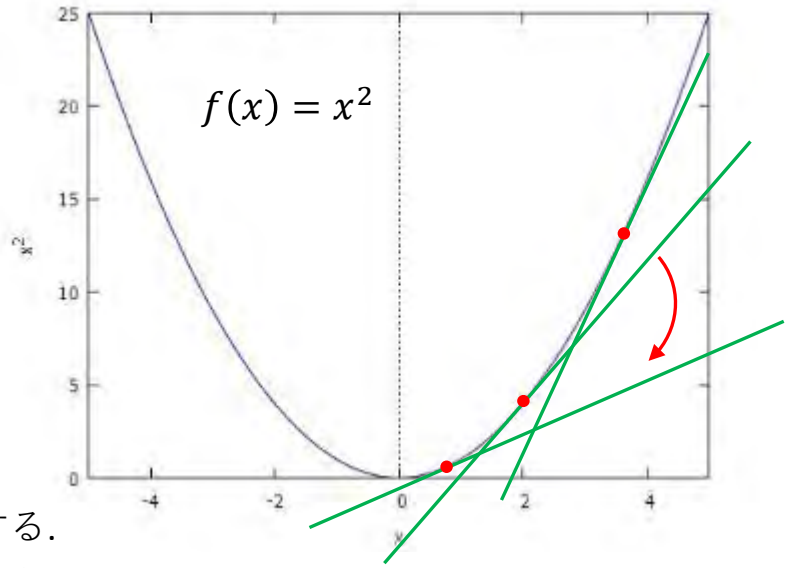
この NW で学習するパラメータは 9 個

試しに最初に $w_{11}^{(1)}$ について計算してみる.

# 学習（勾配法）

勾配が正なら負（負なら正）の方向へ
移動することで、最適値に近づく.

$$x := x - \eta \frac{\partial f(x)}{\partial x}$$




接線の傾きが正であれば $x$ を小さくする.
負であれば $x$ を大きくする.

このときの変化の刻み幅 $\eta$ は課題に応じて調整が必要.

イータ

# 学習（勾配法）

勾配降下法の式に従えば、$w, b$ の更新規則は以下のように記述できる.

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{\partial E}{\partial w_{ij}^{(l)}} \qquad b_{ij}^{(l)} := b_{ij}^{(l)} - \eta \frac{\partial E}{\partial b_{i}^{(l)}}$$

$$E = \frac{1}{2}\sum_{k=1}^{n}\left(y_k - f(\boldsymbol{x}_k)\right)^2$$

$E$ : 損失関数（誤差の評価関数：二乗誤差）

$f(\boldsymbol{x})$ : 出力層での最終出力

$y$ : 正解ラベル

$n$ : 学習データ数　　$k$ : $k$ 番目の学習データ

# 学習（勾配法）

入力層から出力層へ（ベクトル表記）

$f(\boldsymbol{x})$：出力層での最終出力、 $\boldsymbol{h}(\boldsymbol{x})$：活性化関数

$$\boldsymbol{x}^{(0)} = \boldsymbol{x}_k$$ 入力層

$$\boldsymbol{x}^{(1)} = \boldsymbol{h}^{(1)}\left(\boldsymbol{w}^{(1)}\boldsymbol{x}^{(0)} + \boldsymbol{b}^{(1)}\right)$$ 第一層

$$\boldsymbol{x}^{(2)} = \boldsymbol{h}^{(2)}\left(\boldsymbol{w}^{(2)}\boldsymbol{x}^{(1)} + \boldsymbol{b}^{(2)}\right)$$ 第二層

$$f(\boldsymbol{x}_k) = \boldsymbol{x}^{(2)}$$ 出力層

# 学習（勾配法）

早速先ほどの $w_{11}^{(1)}$ の計算を行ってみる．

$$w_{11}^{(1)} := w_{11}^{(1)} - \eta \frac{\partial E}{\partial w_{11}^{(1)}}$$

更新規則

$$\frac{\partial E}{\partial w_{11}^{(1)}} = \frac{\partial}{\partial w_{11}^{(1)}} \left( \frac{1}{2} \sum_{k=1}^{n} \left( y_k - f(x_k) \right)^2 \right)$$

$$= \frac{1}{2} \cdot \frac{\partial}{\partial w_{11}^{(1)}} \sum_{k=1}^{n} \left( y_k - f(x_k) \right)^2$$

$\boldsymbol{h}^{(2)}$：二層目の活性化関数

$$= \frac{1}{2} \cdot \frac{\partial}{\partial w_{11}^{(1)}} \sum_{k=1}^{n} \left( y_k - \boldsymbol{h}^{(2)} \left( \boldsymbol{w}^{(2)} \cdot \boldsymbol{x}^{(1)} + \boldsymbol{b}^{(2)} \right) \right)^2$$

$\boldsymbol{h}^{(1)}$：一層目の活性化関数

$$= \frac{1}{2} \cdot \frac{\partial}{\partial w_{11}^{(1)}} \sum_{k=1}^{n} \left( y_k - \boldsymbol{h}^{(2)} \left( \boldsymbol{w}^{(2)} \cdot \boldsymbol{h}^{(1)} \left( \boldsymbol{w}^{(1)} \boldsymbol{x}^{(0)} + \boldsymbol{b}^{(1)} \right) + \boldsymbol{b}^{(2)} \right) \right)^2$$

一層目の重み補正値の計算をするためには二層目の計算も必要になる

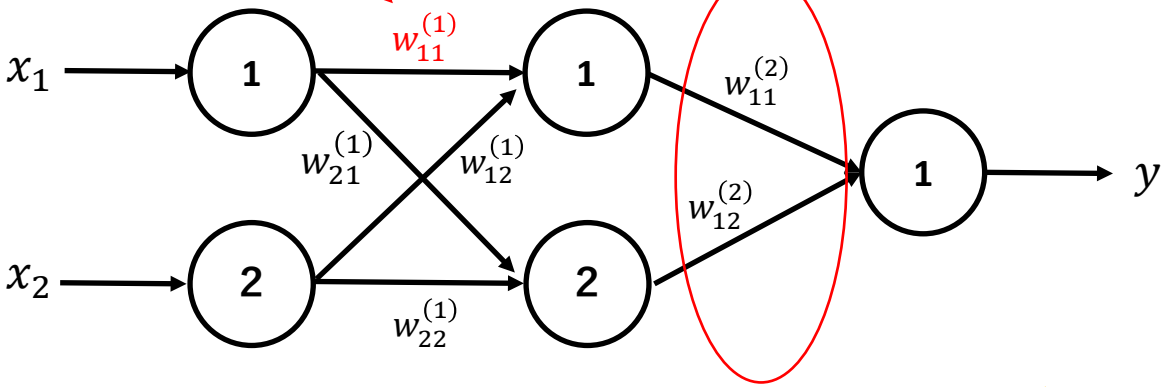
→ 出力層から順に計算すれば楽　→ 誤差逆伝播法

# 誤差逆伝播法

話を要約すると

これを計算するためには

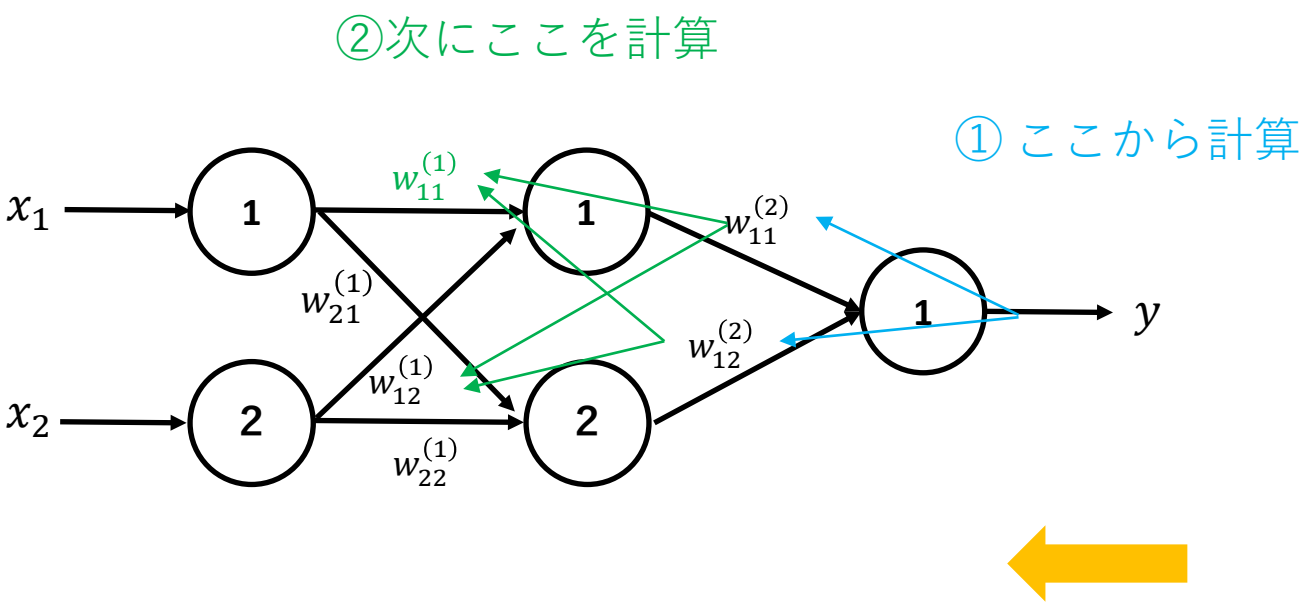
これらの計算結果が必要になる.




出力層から順に計算すれば楽

# 誤差逆伝播法

話を要約すると



# 誤差逆伝播法（結果のみ）

更新式全体は

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{\partial E}{\partial w_{ij}^{(l)}} = w_{ij}^{(l)} - \eta \cdot \delta_i{}^{(l)} x_j^{(l-1)}$$

重みの更新式

$$b_{ij}^{(l)} := b_{ij}^{(l)} - \eta \frac{\partial E}{\partial b_i^{(l)}} = b_{ij}^{(l)} - \eta \cdot \delta_i{}^{(l)}$$

バイアスの更新式

$$\delta_i{}^{(L)} = \left(h^{(L)}\left(z_i^{(L)}\right) - y_k\right) \cdot h'^{(L)}\left(z_i^{(L)}\right)$$

出力層の $\delta$

$$\delta_i{}^{(l)} = h'^{(l)}\left(z_i^{(l)}\right) \cdot \sum_{r=1}^{m^{(l+1)}} \left(\delta_r{}^{(l+1)} \cdot w_{ri}^{(l+1)}\right)$$

中間層の $\delta$

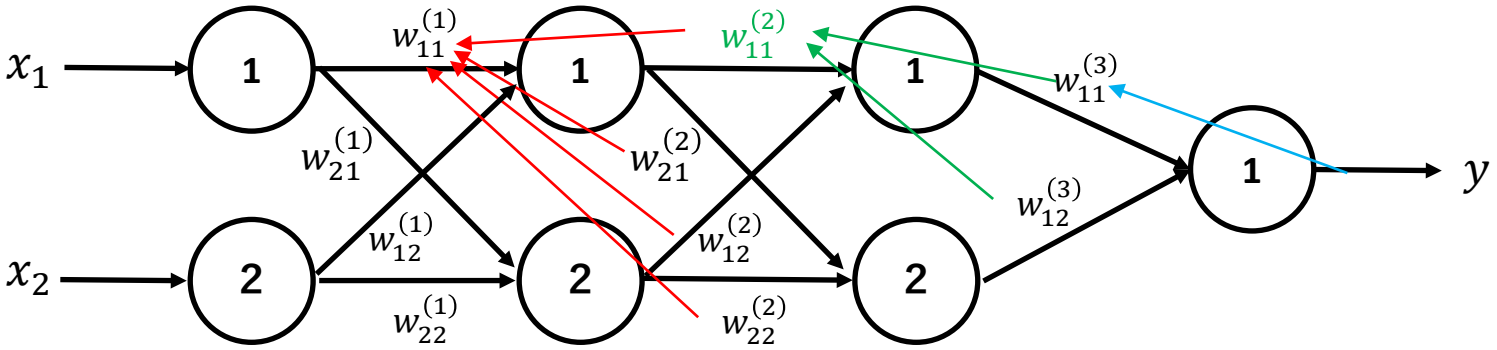
活性化関数の微分形．活性化関数は微分可能な関数である必要がある．
この値が小さくなってくると、入力に近い層の補正値ほど少なくなる．

# 誤差逆伝播法の問題点

## 勾配消失問題

計算ステップが進むにつれ補正量が少なくなり次第に学習が進まなくなってくる．　多層になると入力に近い層は学習が進まない．

原因は中間層の $\delta$ の値の減少　　$\delta_i{}^{(1)} = h'^{(1)} \cdot h'^{(2)} \cdot h'^{(3)} \cdot h'^{(4)} \ldots \cdot h'^{(n)}$

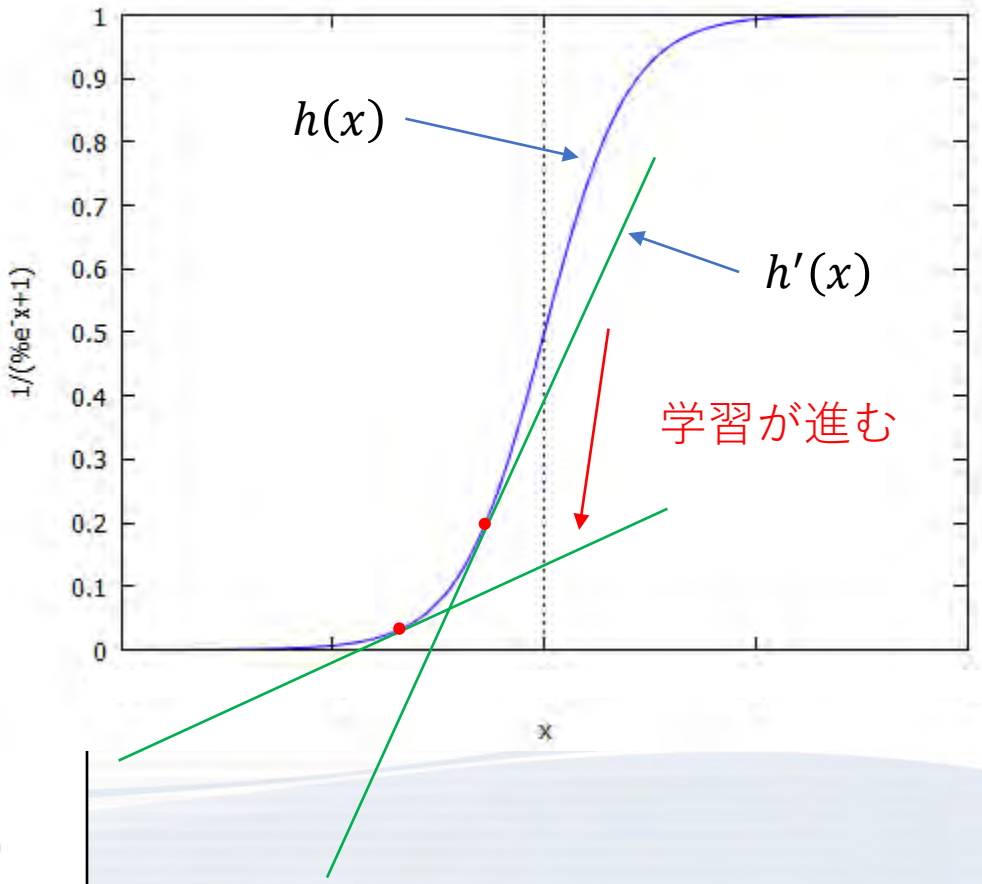



第二次 NN ブーム終焉の原因になったとも言われる．

# 勾配消失問題

活性化関数
（ここでは Sigmoid）

$$h(x) = \frac{1}{1 + e^{-x}}$$




微分値は、学習が進むにつれゼロに近づいていく．さらに入力に近い層ではそれらが乗算されてゼロへ近づく．

$h'^{(l)} \rightarrow 0$

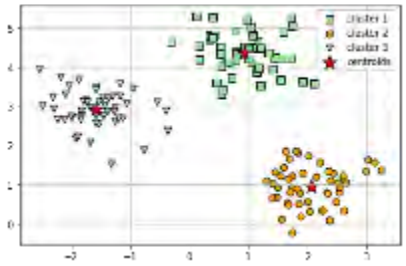
$h'^{(1)} \cdot h'^{(2)} \cdot h'^{(3)} \ldots \cdot h'^{(n)} \rightarrow 0$

# 教師なし学習

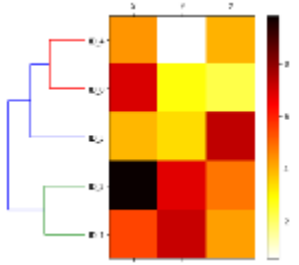
# 教師なし学習

機械学習一般における教師なし学習

*K-means Clustering*

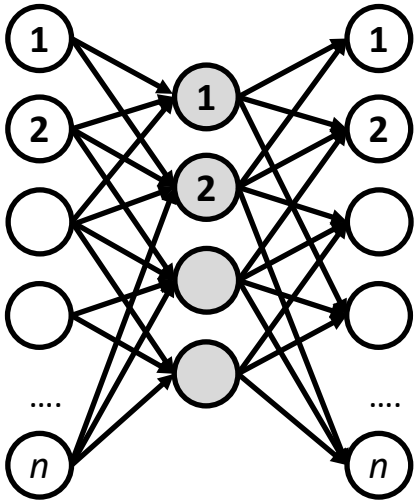


*Hierarchcal Clustering*



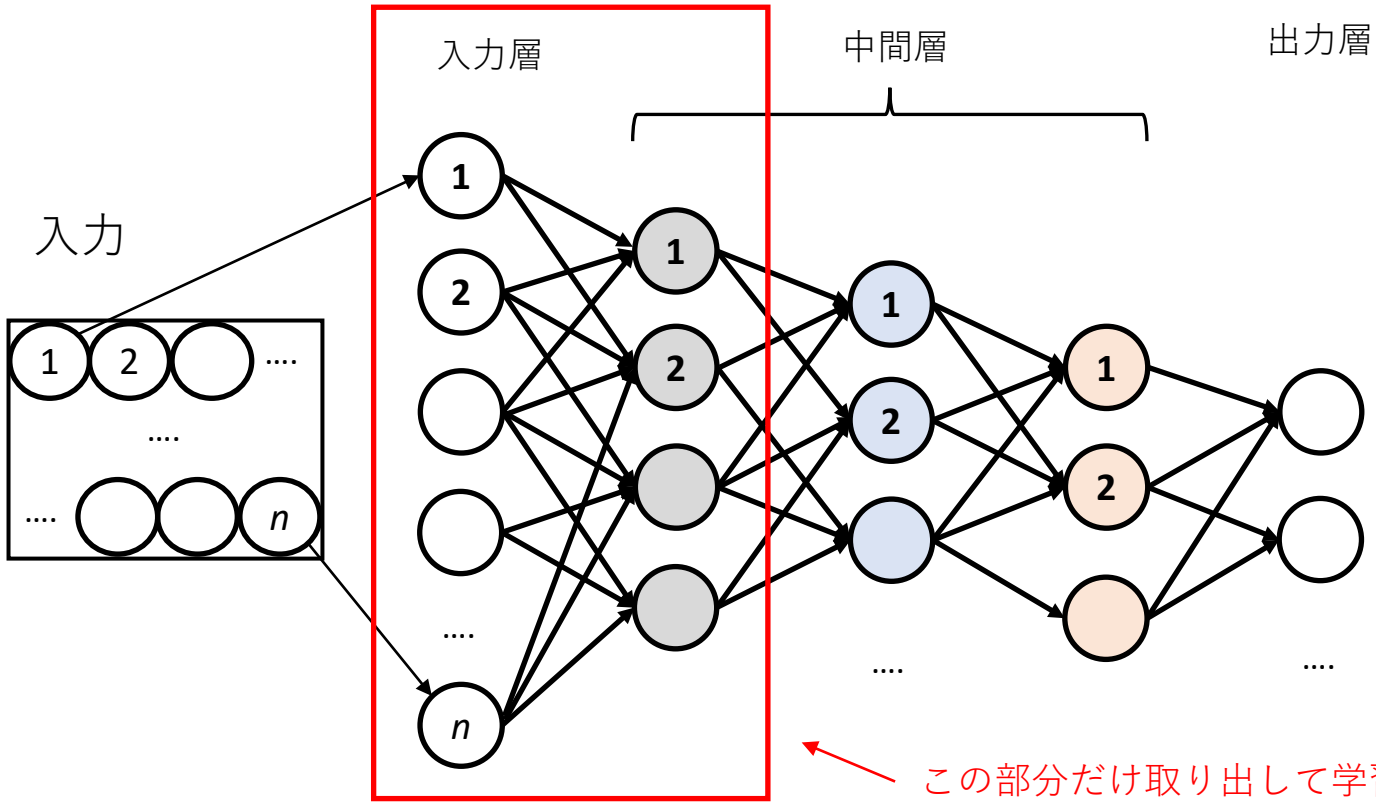
ニューラルネットワークにおける教師なし学習

*AutoEncoder*



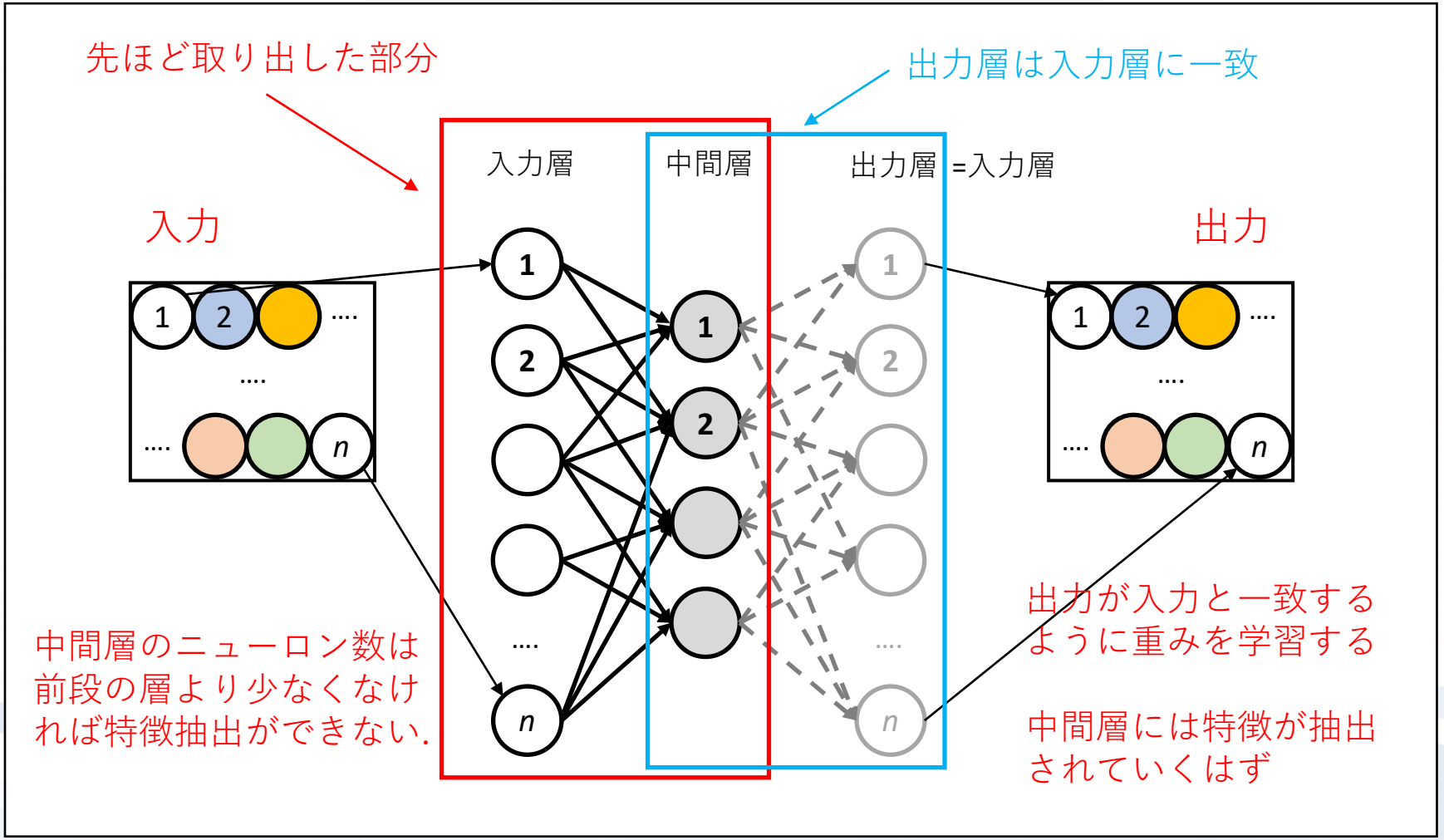
# ニューラルネットワーク：教師なし学習

隠れ層（中間層）の効率的学習



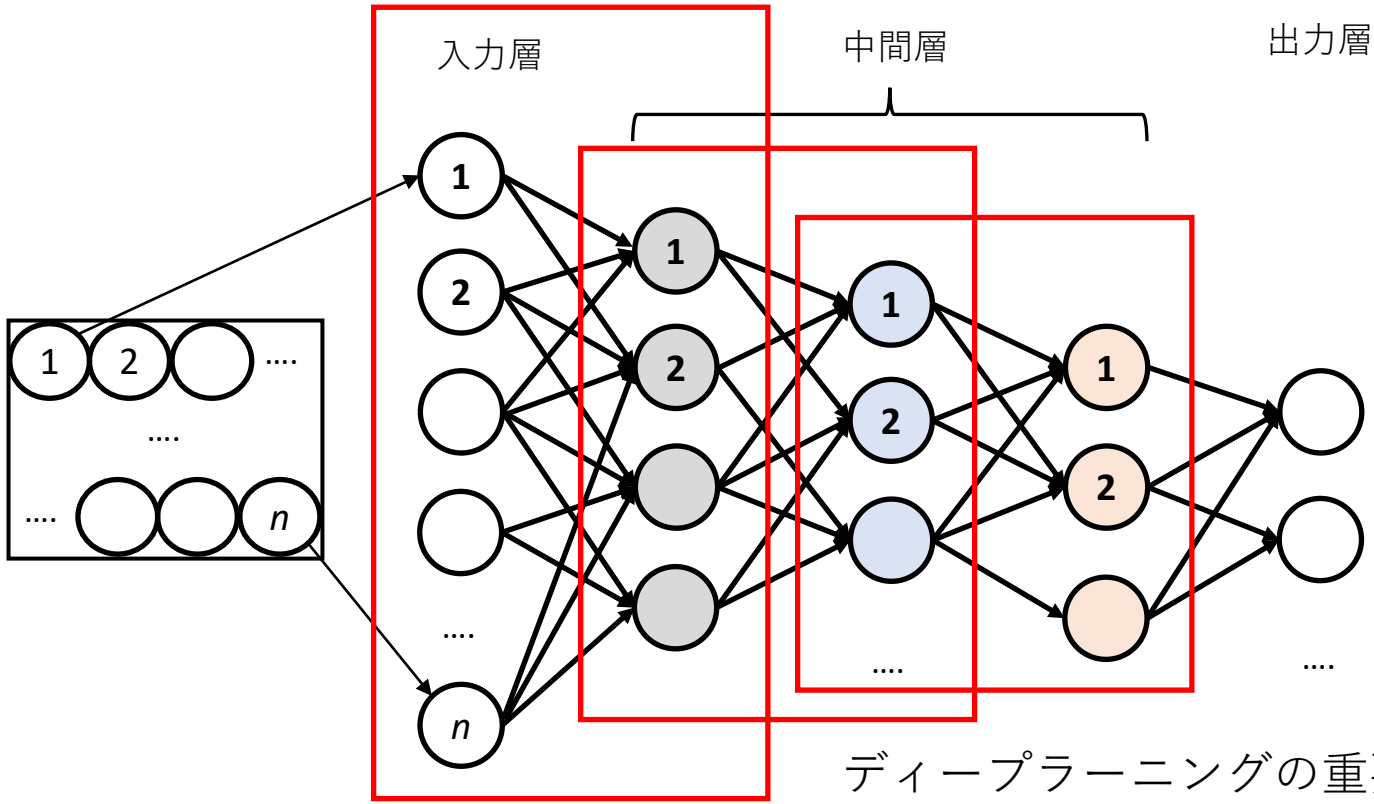

この部分だけ取り出して学習

# ニューラルネットワーク： *AutoEncoder による圧縮*



# ニューラルネットワーク：教師なし学習

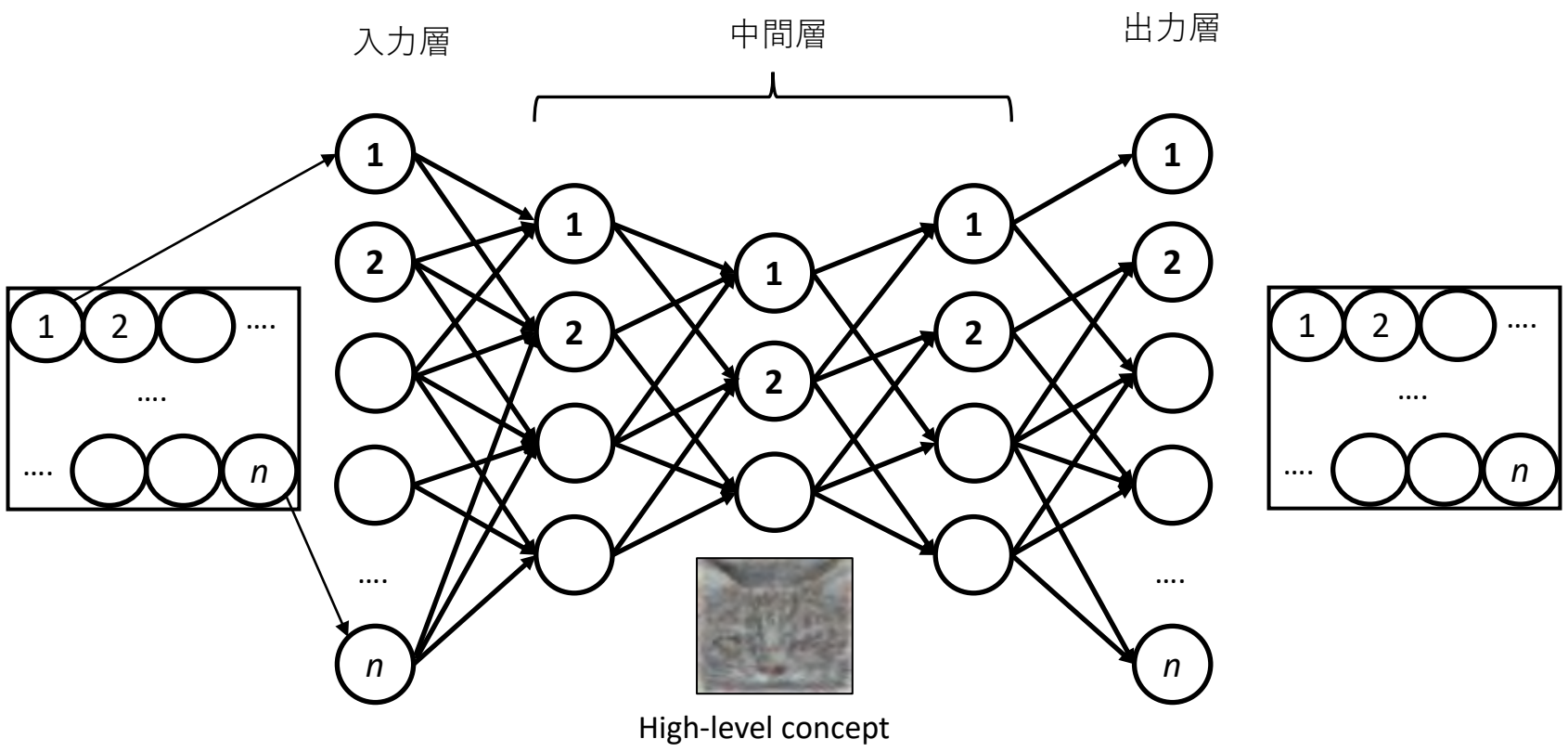
隠れ層（中間層）の効率的学習




ディープラーニングの重要なアイデア

# ニューラルネットワーク：教師なし学習

高次概念の生成に関わる



# 教師なし学習の応用例 特徴分類

- 特徴量の自動抽出
  - 画像の中の特徴抽出を自動化
  - 新しい画像の生成（合成）
- 真偽の判定
  - 異常個所特定
    - 学習済みの特徴抽出器にデータを入力し、その入出力間の差異を調べる。異常の種類の調査
- NN の事前学習
  - NNを学習させる前の値
    - 学習前の各パラメータの値をランダムに振るのではなく、事前にオートエンコーダーで学習した値を用いる.

# データサイエンス応用コース
# ディープラーニング I･II

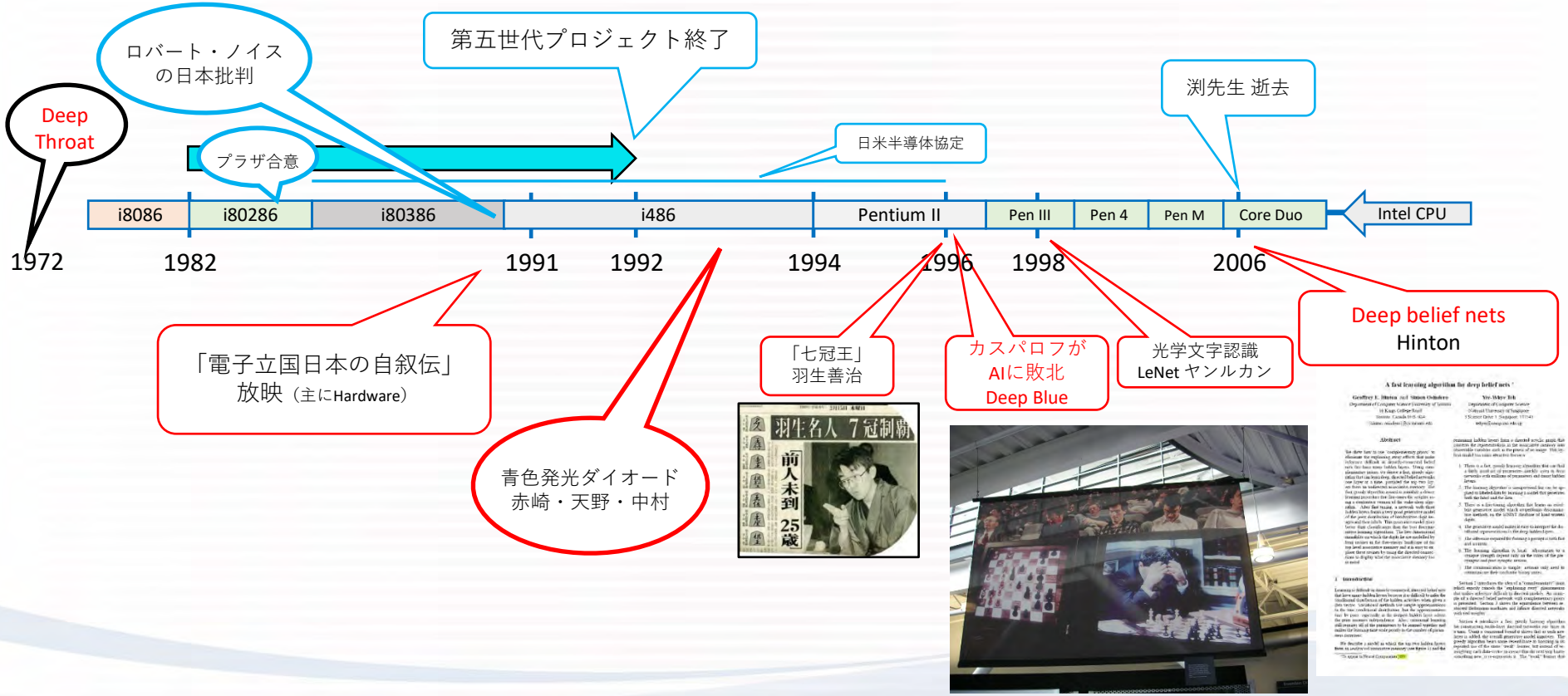
（畳み込み/再帰型）

下川　和郎
大阪大学

# Deep Learning

- いつから出てきた言葉か？
  - AI 関連技術の歴史的経緯
  - Deep の 単語の意味 [形容詞]
    - （奥行きのある）深い
    - （学問・知識など）奥底の計り知れない、洞察力の深い
    - 理解しがたい、難しい
    - 深層の
- 具体的には何を指す？
  - 過去の技術との違いは？
    - 過去のニューラルネットワークとの違い
    - 新しく何ができるようになった？

# 年表：第五世代プロジェクト
# ～




IT に関連する国内研究のネガティブな話題

IT に関連する世界の動き

# 年表：　　～現在

Deep Learning

Deep belief nets
Hinton

ILSVRC 優勝
Hinton

第2回将棋電王戦
ソフト側が
プロ棋士らに勝利

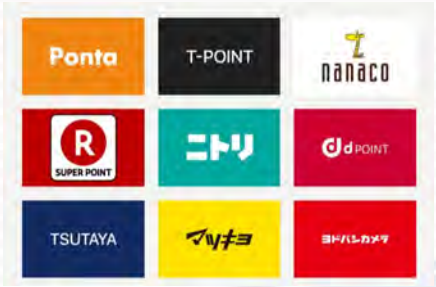
2006　2010　2012　2020

ノイズ除去
オートエンコーダー

Google アルゴリズムが
猫を自動認識


https://googleblog.blogspot.com/2012/06/using-large-scale-brain-simulations-for.html
より

ビッグデータの集積と利用が加速し始める




ポイントカード

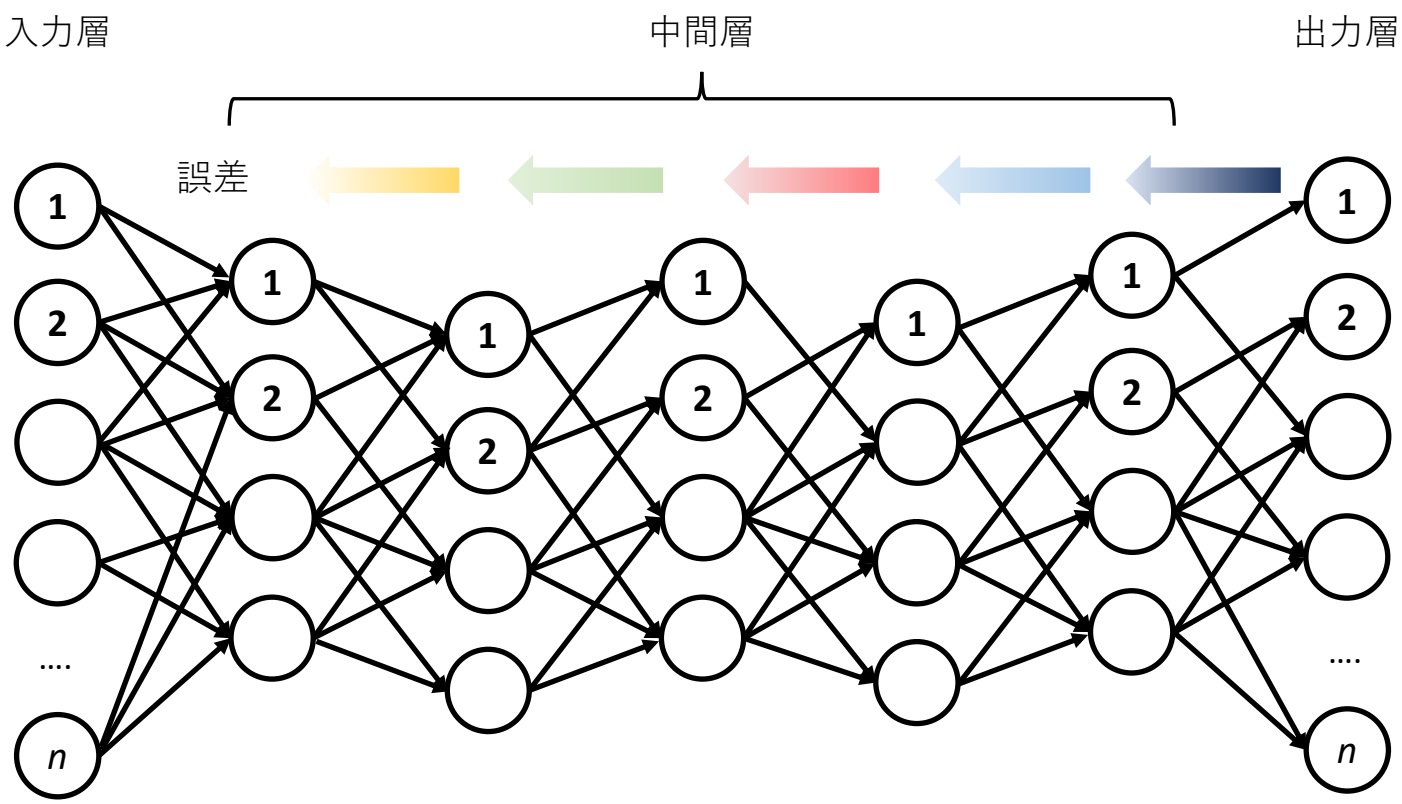

インターネット決済

# Deep Learning： 定義（曖昧）

- 三つ以上の層を持つニューラルネットワーク（NN）
  - ニューロン数の増加
  - 層やニューロン間でのより複雑な接続
  - 訓練に利用可能な計算能力の爆発的増加
  - 自動的な特徴量抽出

- 以下のようなニューラルネットワーク
  - 教師なしの事前訓練済みネットワーク
  - 畳み込みニューラルネットワーク （Convolutional NN）
  - リカレントニューラルネットワーク （Recurrent NN）
  - リカーシブニューラルネットワーク （Recursive NN）

# Deep Learning : 定義の一つ

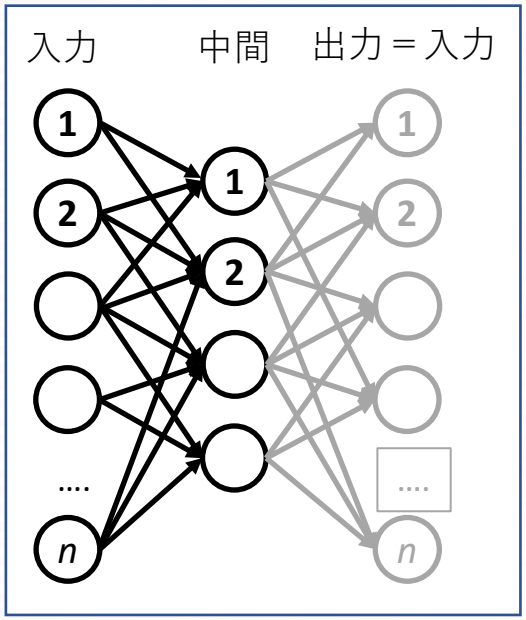


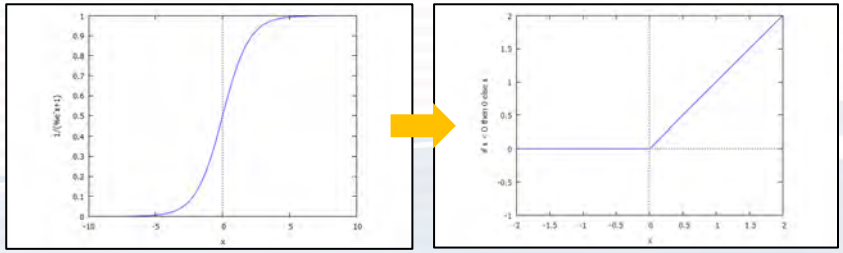

層が深い NW 構造.... 学習できなくなる？ 性能が上がる？

# 多層化における性能向上

- オートエンコーダー
  - 入力（前段）を教師信号とした学習によって各層間の最適化を行う.
- 活性化関数
  - 学習が進行しても勾配消失が起こりにくい関数を利用.

  ex. Sigmoid → ReLU



Sigmoid　　ReLU

# その他の性能向上

- SGD
  - 局所最適解に陥らない方法
    - ランダムに選んだ訓練データの一部で重み $\boldsymbol{W}$ を求める.
    - 訓練データの和だけ繰り返す.
    - → 初期値が毎回異なるため局所解にとらわれにくい.

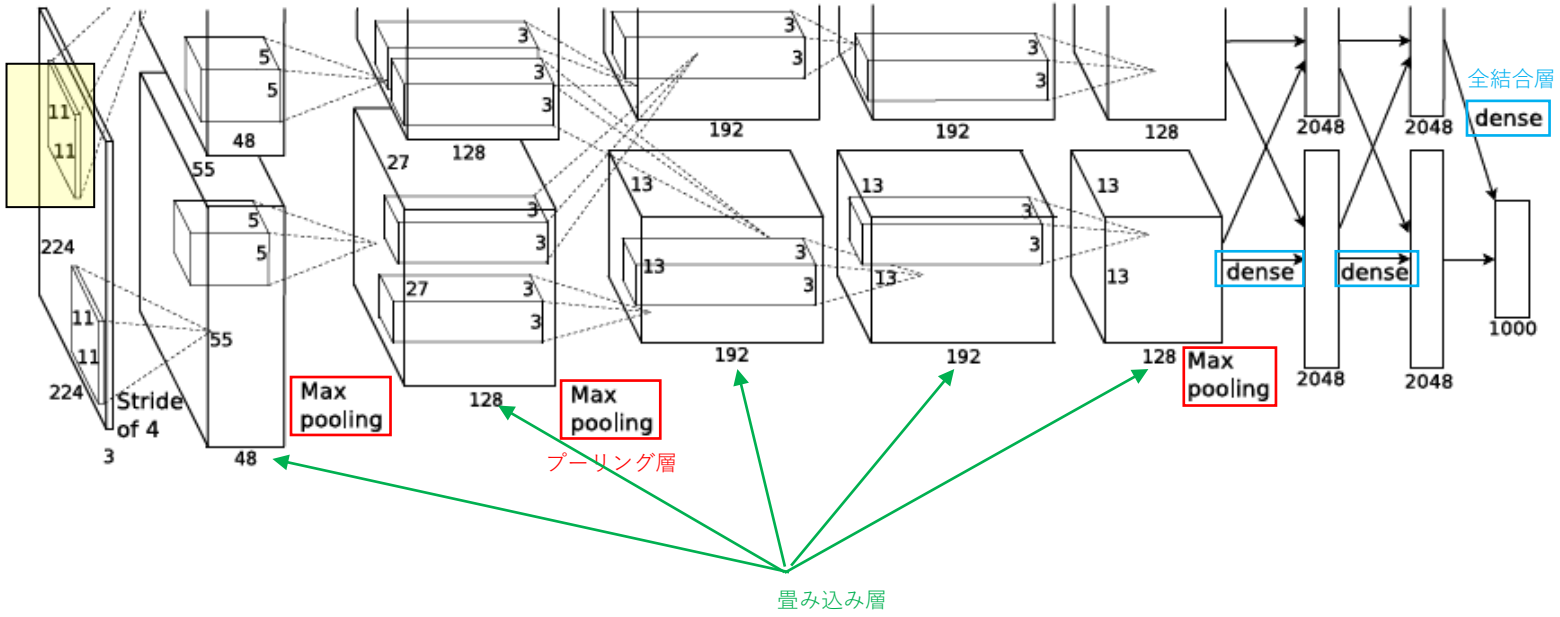
- ミニバッチ法
  - 訓練データを小分けして実施する
    訓練一回分 → 「イテレーション」
  - 小分けした訓練データを一巡する
    → 「エポック」
  - 小分けの仕方を変えて複数エポック
    を実施する.

# 畳み込み
# ニューラルネットワーク

# 畳み込みニューラルネットワーク：CNN

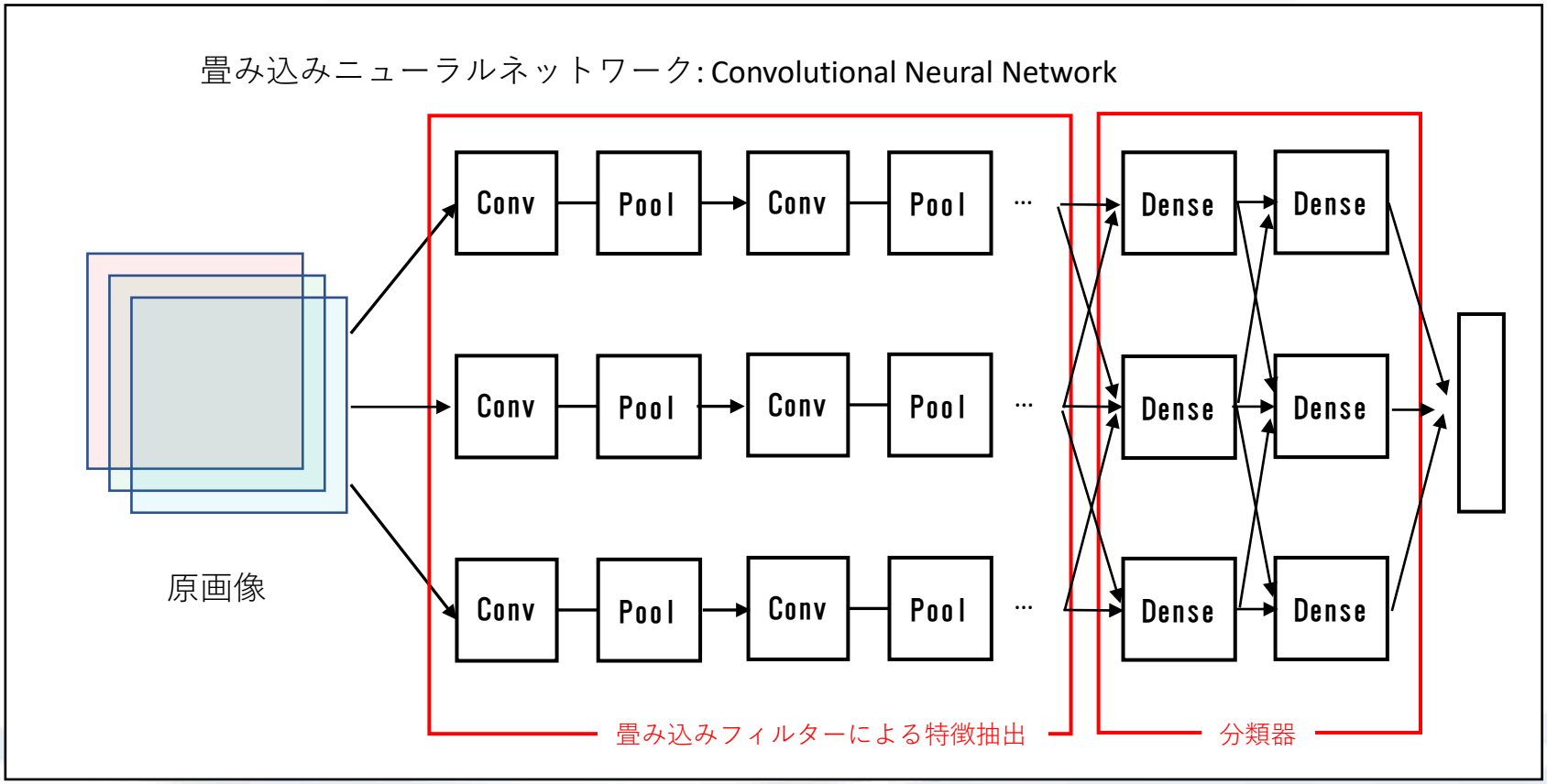
- 空間的な画像認識に向いた手法
- LeNet, Alex Net などの文字/画像認識においても使われた.
- 特定の層の組み合わせで利用される.
    - 畳み込み層
    - プーリング層
    - 全結合層

# Alex Net の構造

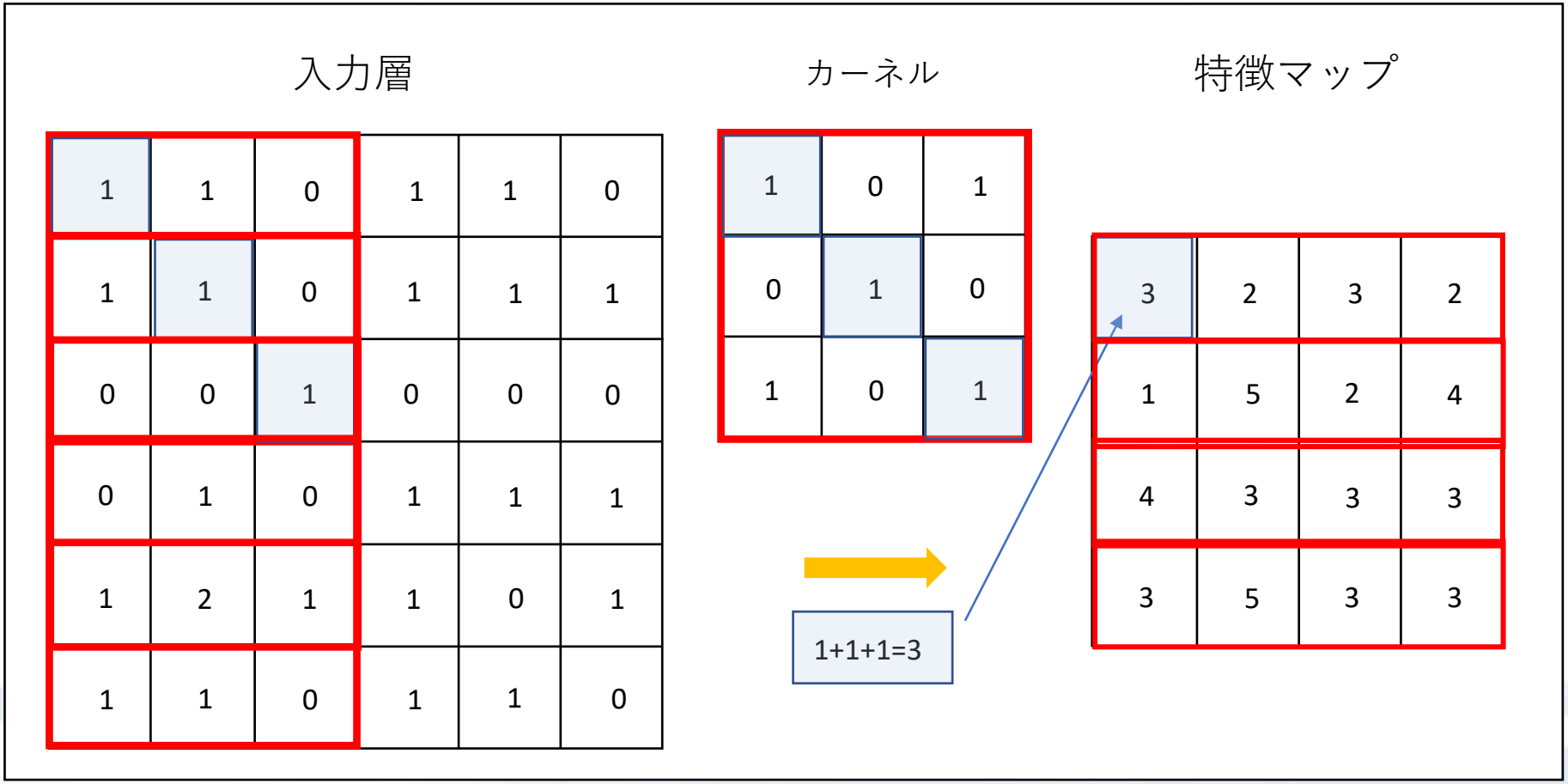



Hinton (NIPS 2012)
ImageNet Classification with Deep ConvolutionalNeural Networks より

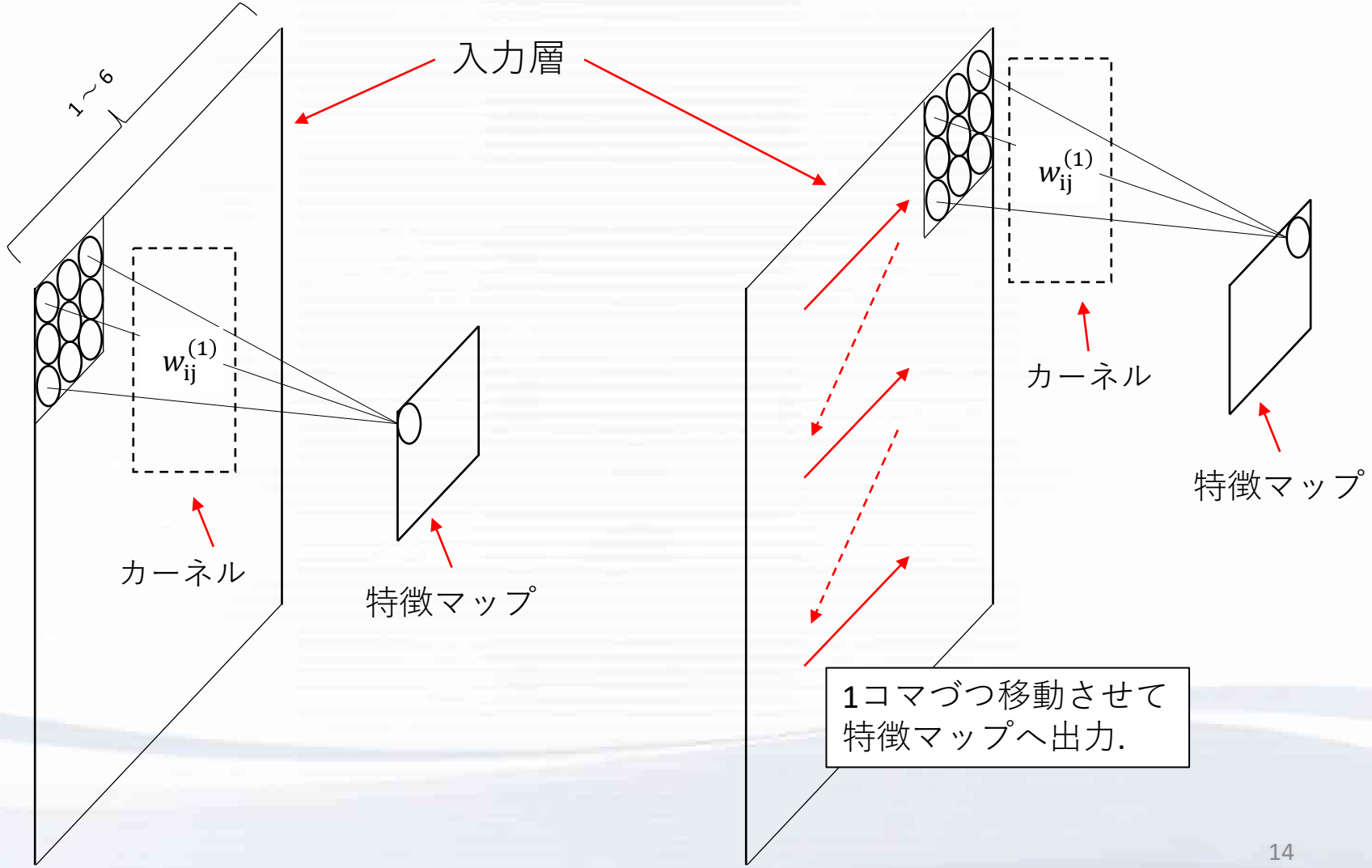
# 畳み込みニューラルネットワーク



# 畳み込み（演算）は何をしている？

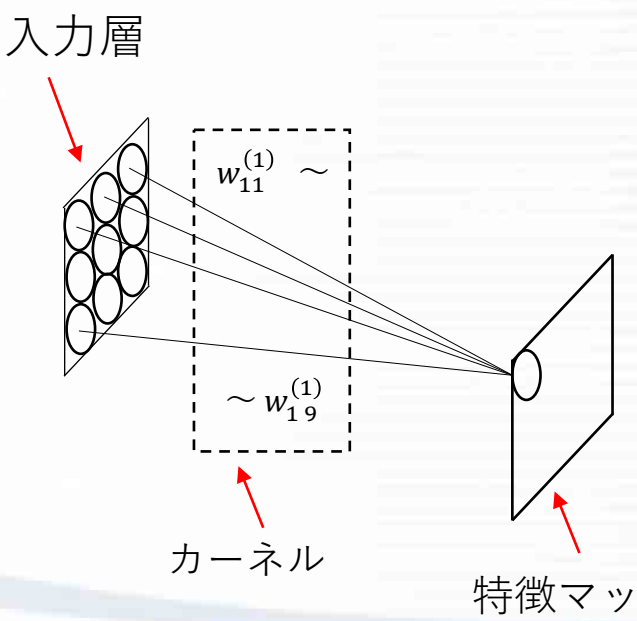


# 畳み込み（演算）3D 表示

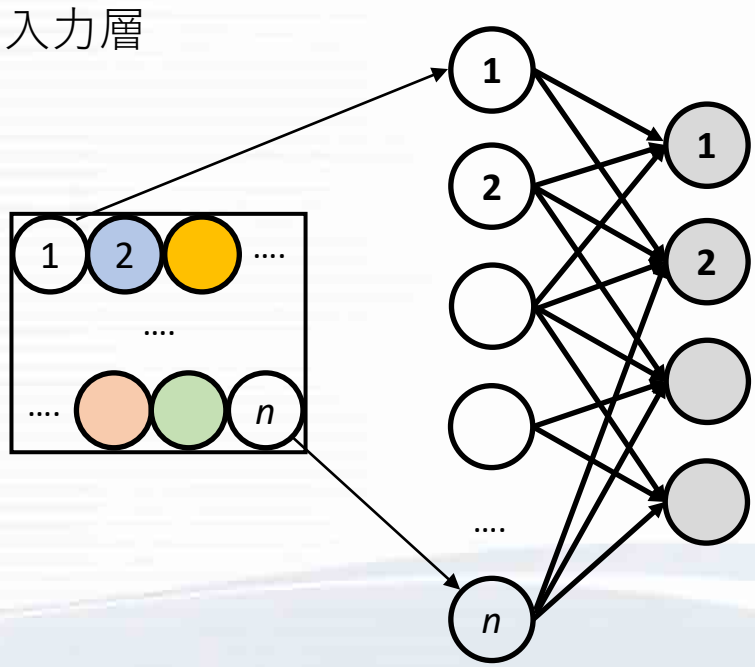


# 畳み込みと全結合

畳み込み：二次元空間上の関係が維持される.




全結合：二次元空間上の関係は考慮されない.



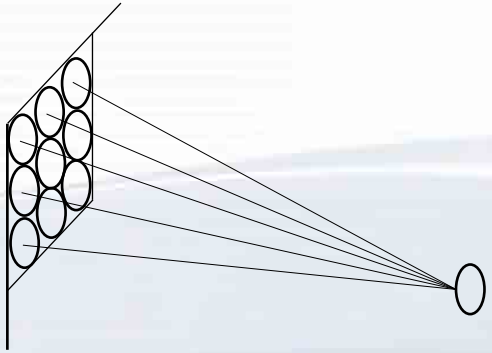
# 畳み込み演算における学習

- Pooling によって出力層と繋がらなくなったユニットについては逆伝播の計算は行わない.

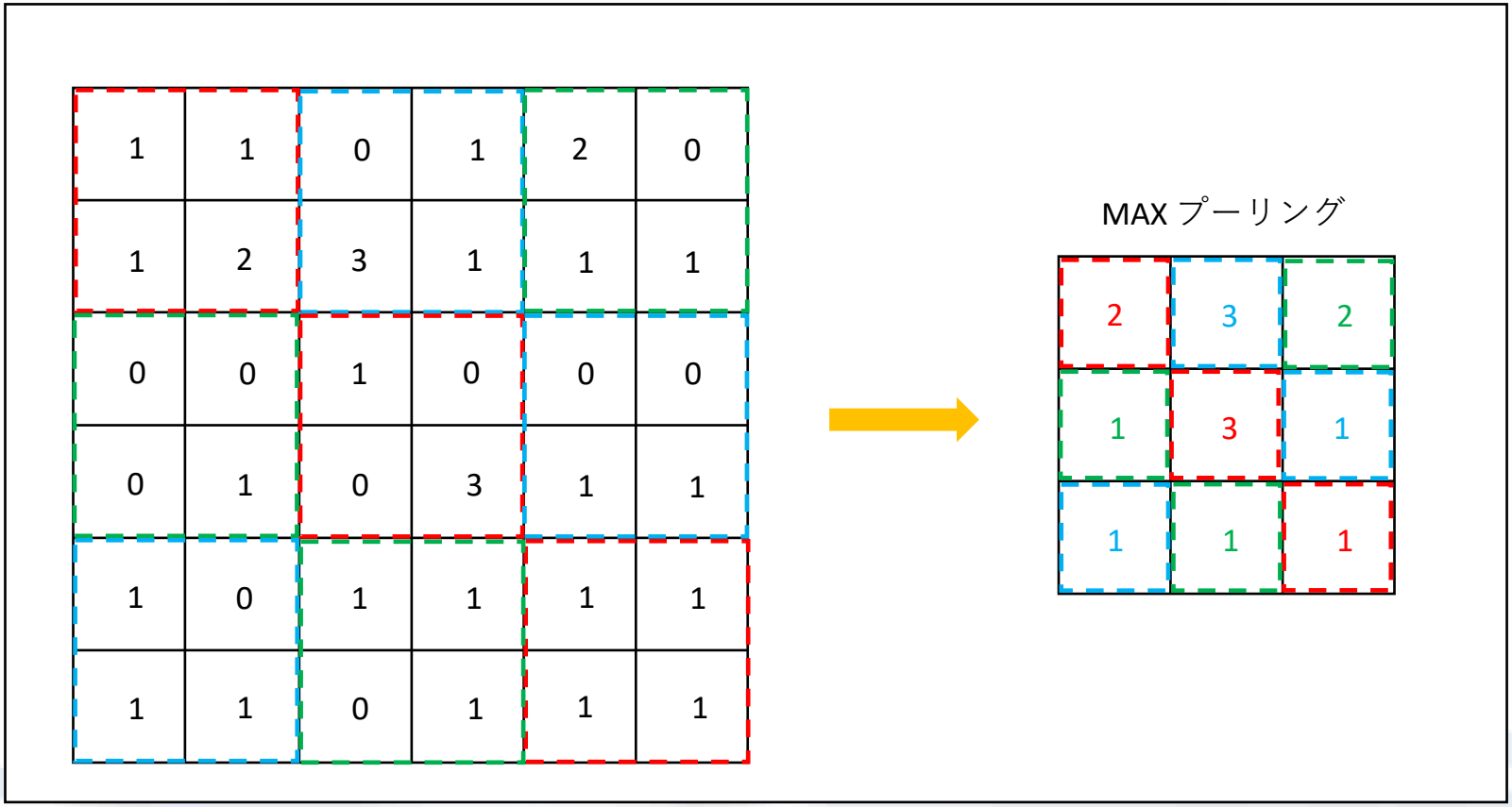
- 畳み込み層に接続される畳み込み層
  - 出力層側の畳み込み層の各フィルタ分の計算が必要

- 全結合層に接続される畳み込み層
  - 全結合層側の各素子分の計算が必要

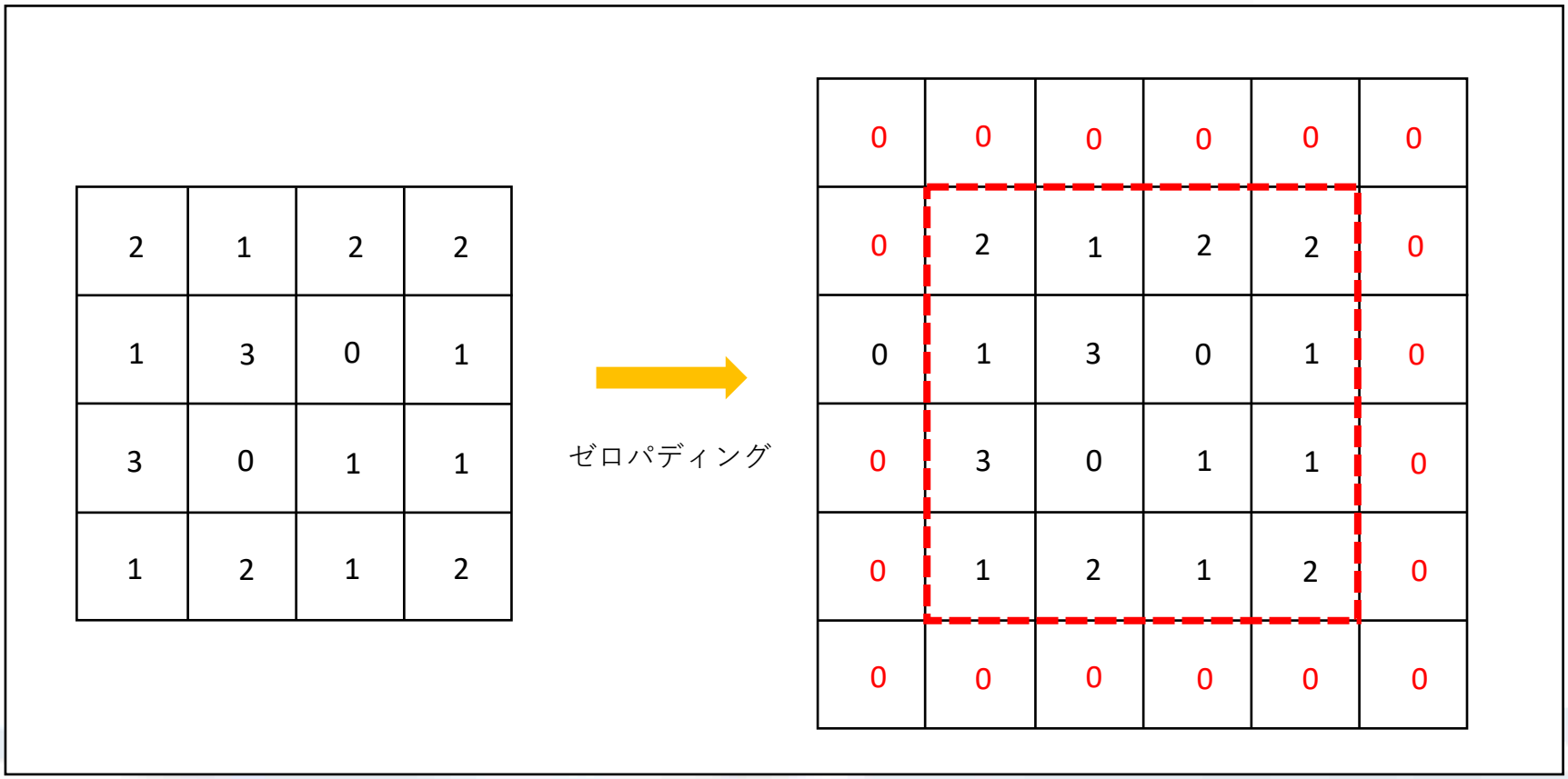
※ユニット：特徴マップにおける
一つの素子から入力層へ繋がる
一まとまりの結線



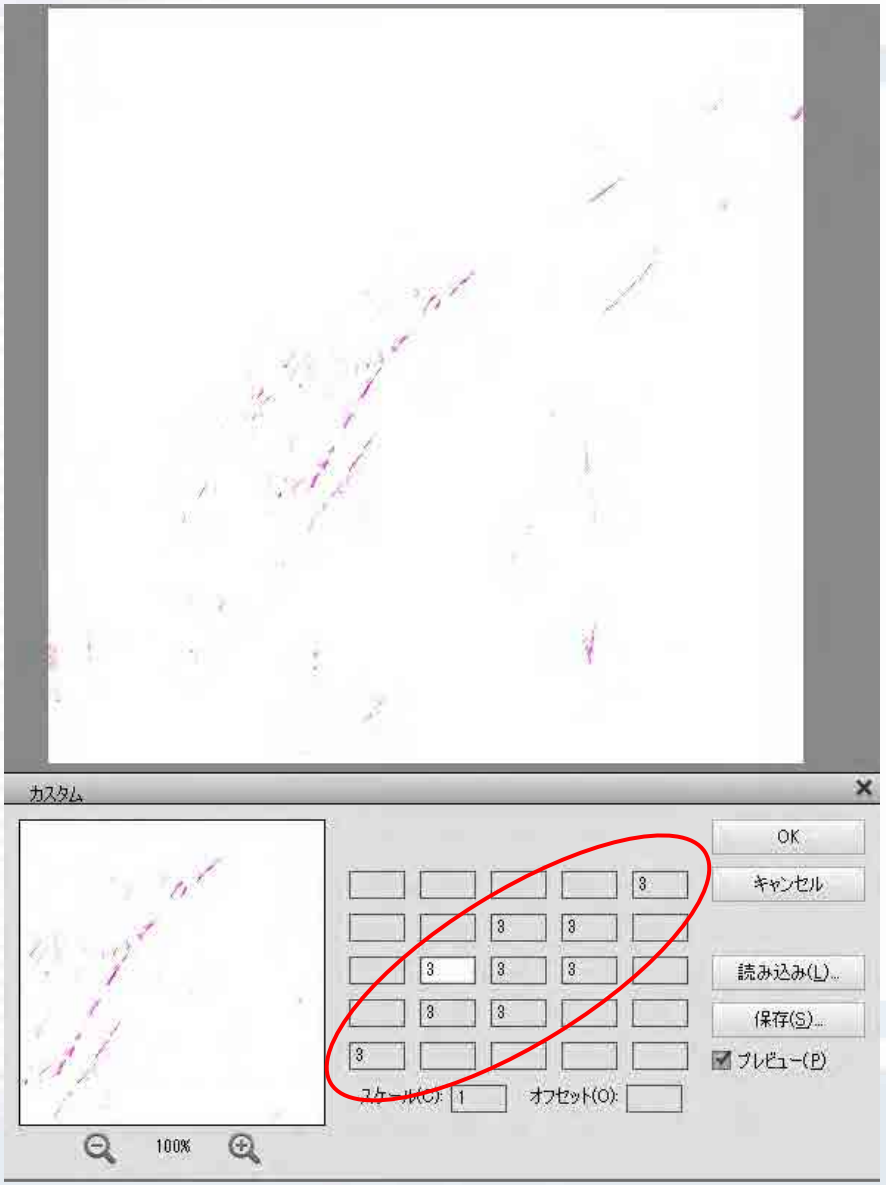
# 畳み込みNN: プーリング



# 畳み込みNN: パディング

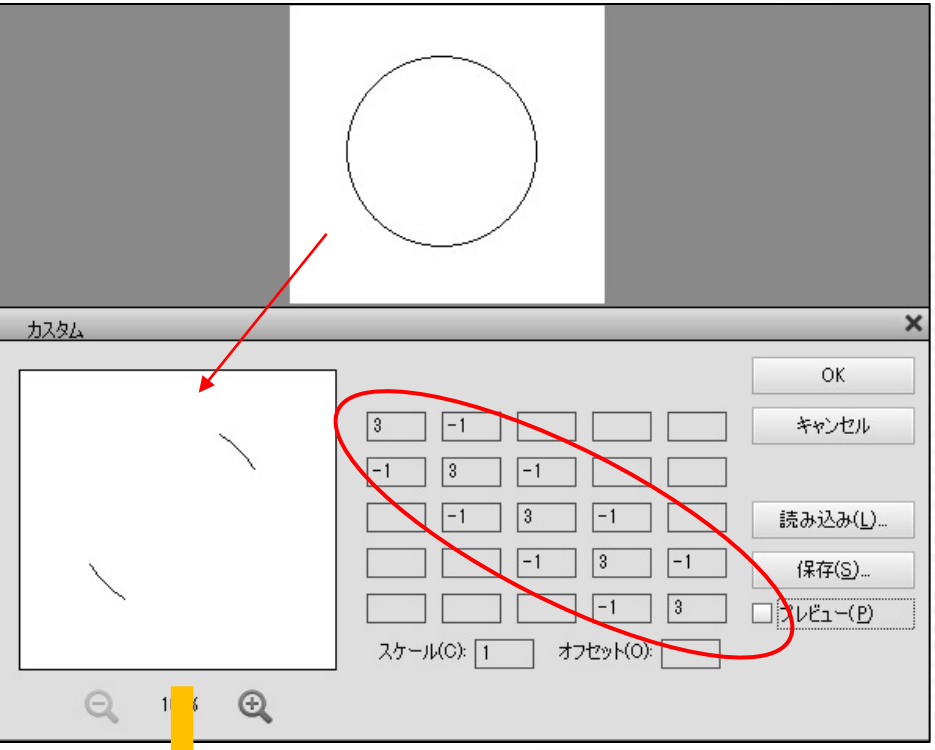


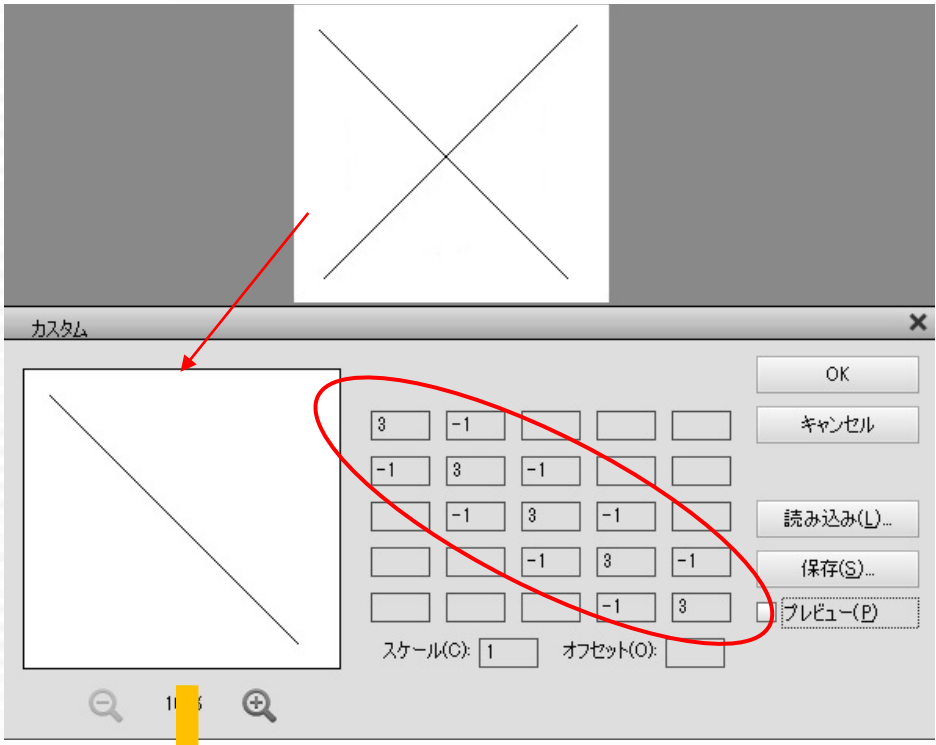

畳み込みフィルターによる演算結果

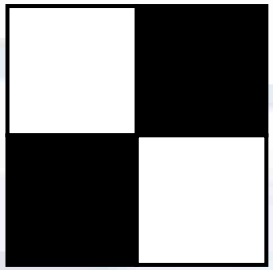



斜線型のカーネルによって
画像の中の斜線部分を
取り出すことができる.

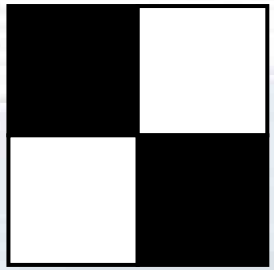
# 畳み込み NN による〇×判定器



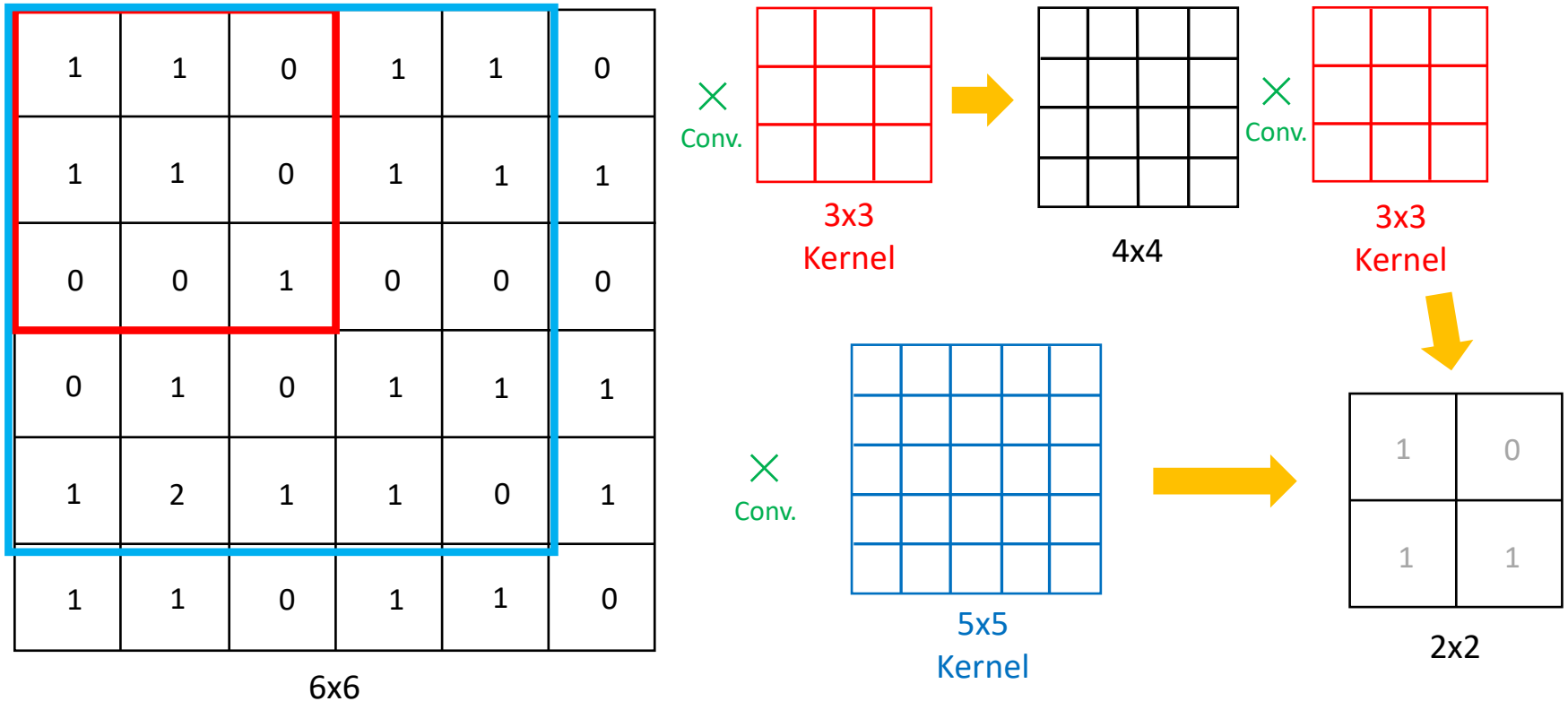



プーリング後の
2x2 特徴マップ



カーネルと特徴マップの組み合わせによって〇×判定が可能になる.
判定は全結合層に任せる.

# 畳み込みNN: 多層化の影響



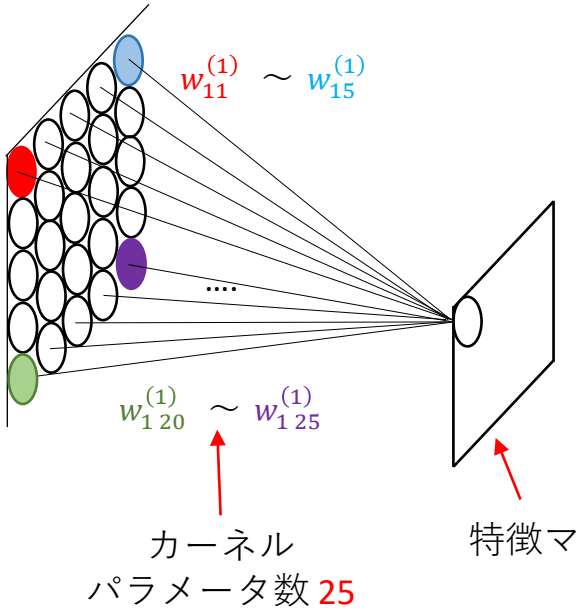

5x5の畳み込み 1 回と 3x3 の畳み込みを 2 回は同じような処理だが

# 畳み込みNN: 多層化

## 5x5 畳み込み

パラメータ数 25

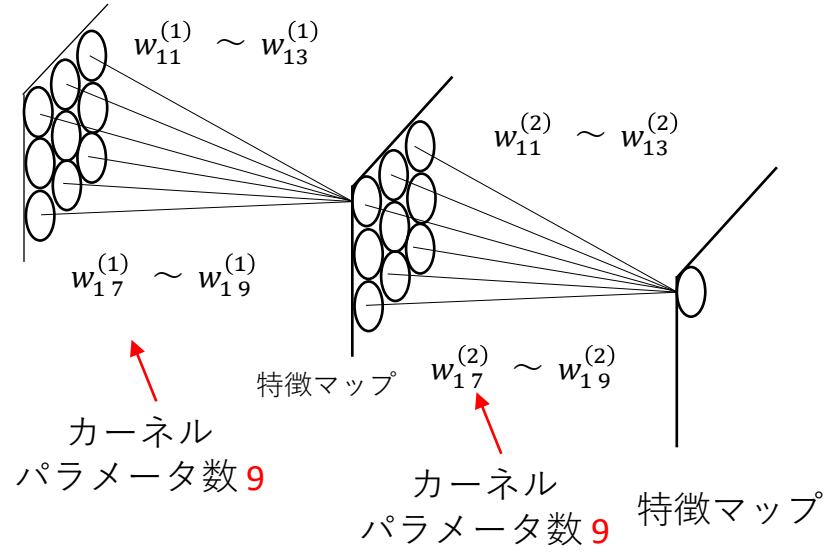
入力層




## 3x3 畳み込み 2 回

パラメータ数 9x2 = 18

入力層




より少ないパラメータで系を記述できた方が良いモデルになる.

# Alex Net（2012）による学習結果

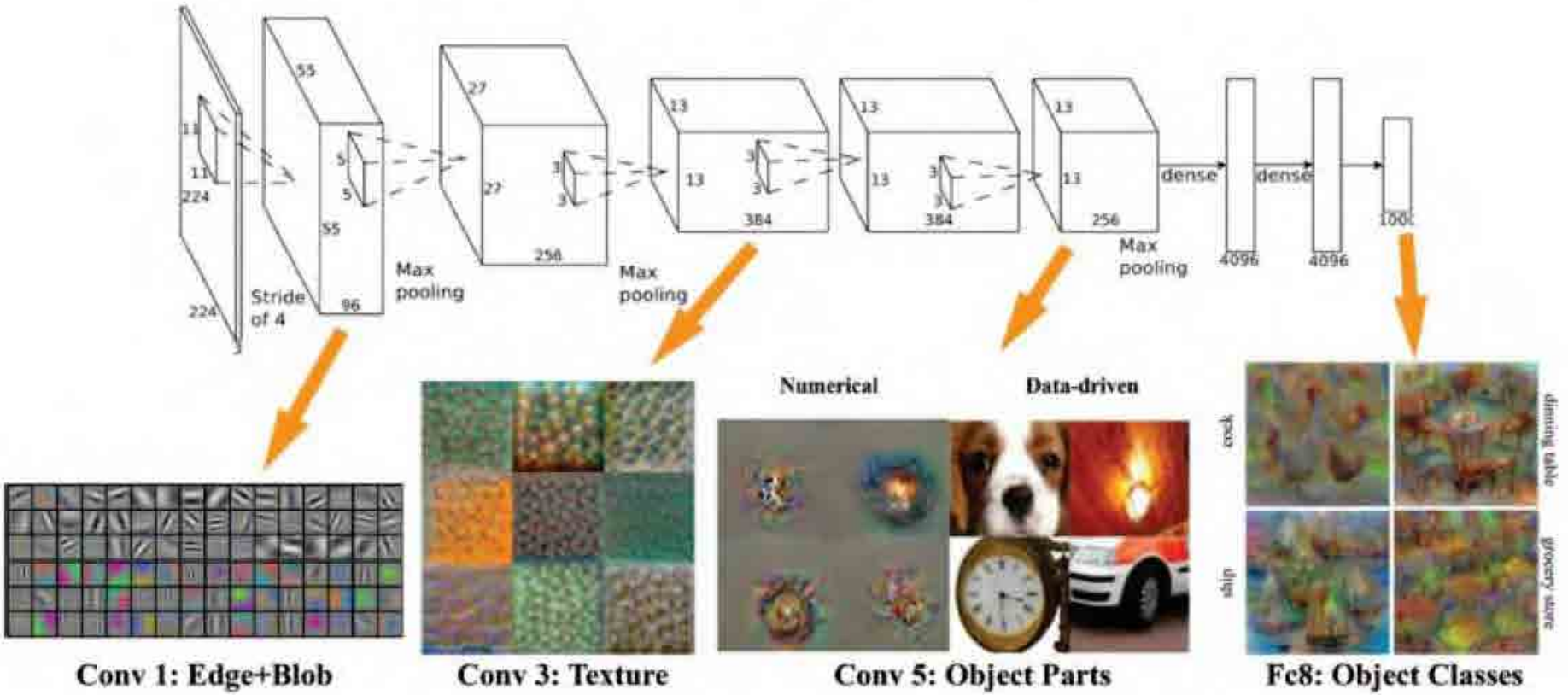



Figure 3.2: Alexnet, an example of convolutional deep learning architecture trained for the ImageNet Large Scale Visual Recognition Competition (ILSVRC), and the role the layers play in detecting features. More in details, the network is build with a series of convolutional layers followed by feed forward layers in the final positions. [25]



Alex Krizhevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks" (2012) より

# LeNet（1998）の構造

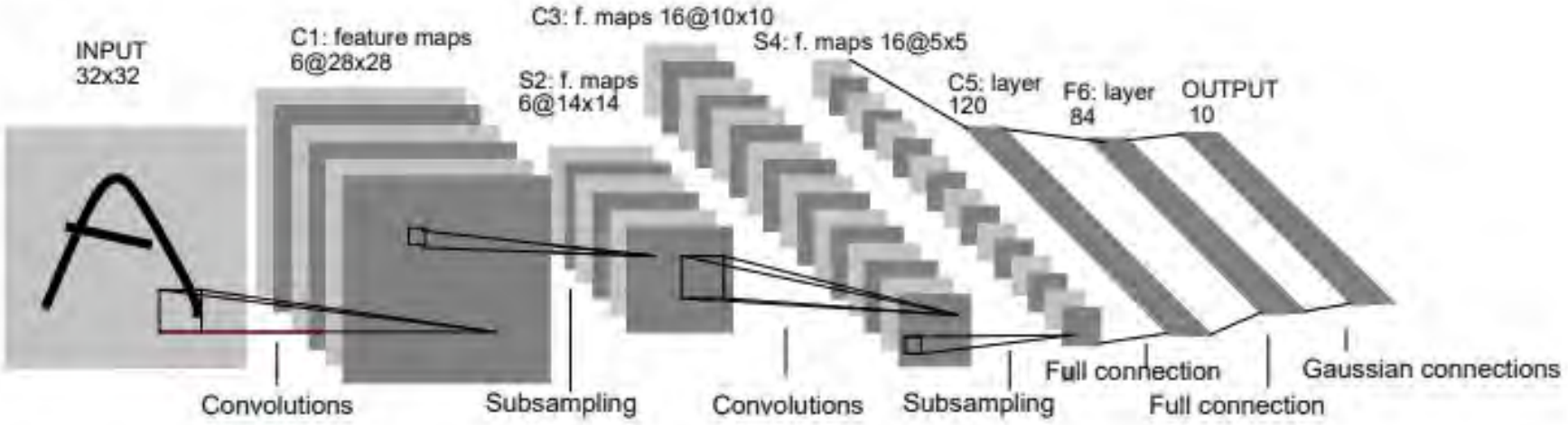



**Fig. 1.** Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Yann LeCun et al. "Object Recognition with Gradient-Based Learning" (1998) より

# データサイエンス応用コース
# ディープラーニング I･II

（畳み込み/再帰型）
後半

下川　和郎

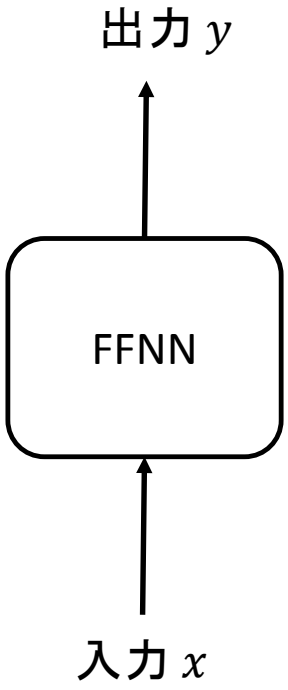
大阪大学

# 再帰型
# ニューラルネットワーク

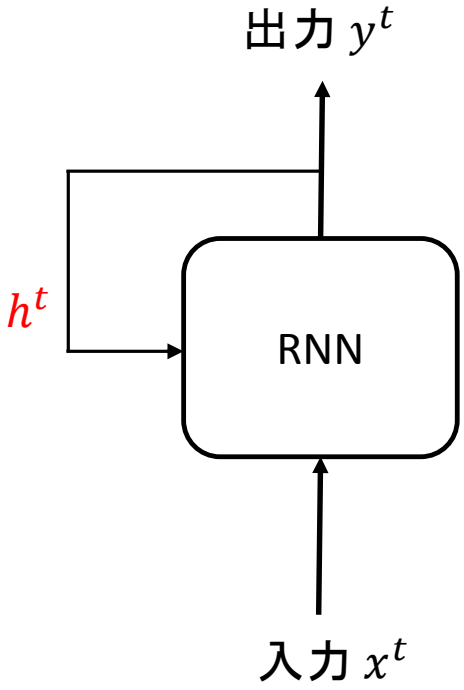
# 再帰型ニューラルネットワーク（RNN）

- Recurrent Neural Network
  - FFNN, CNN との違い
    - 過去の記憶を持っているかいないか
  - Simple RNN
    - 単純. 10 Timestep が実用範囲
  - LSTM（Long Short-Term Memory 1997）
    - 勾配消失を防ぐ構造. 1,000 Timestep 対応可.
  - GRU（Gated Recurrent Unit 2014）
    - LSTM の計算コスト簡素化

# RNN: FFNN との違い

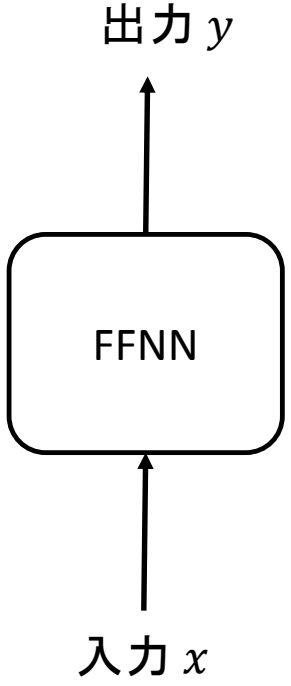



$$y = f(Wx + b)$$




$$y^t = h^t = tanh(Wx^t + Wh^{t-1} + b)$$

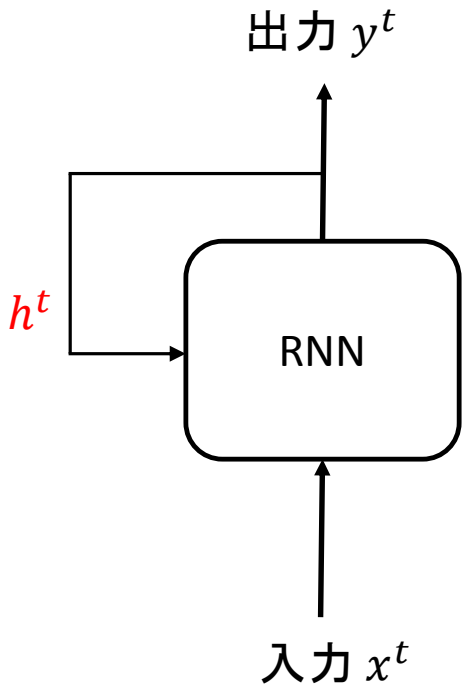
# RNN: FFNNとの違い




You say Goodbye , and I say Hello.

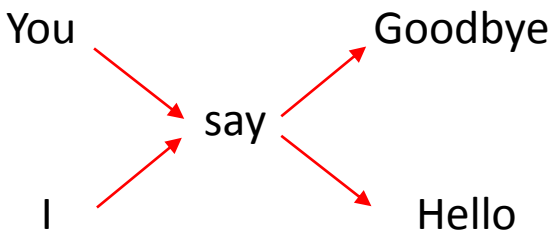
| say | Goodbye | and | I | say | Goodbye ? |
|---|---|---|---|---|---|
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| You | say | Goodbye | and | I | say |

# RNN: FFNN との違い



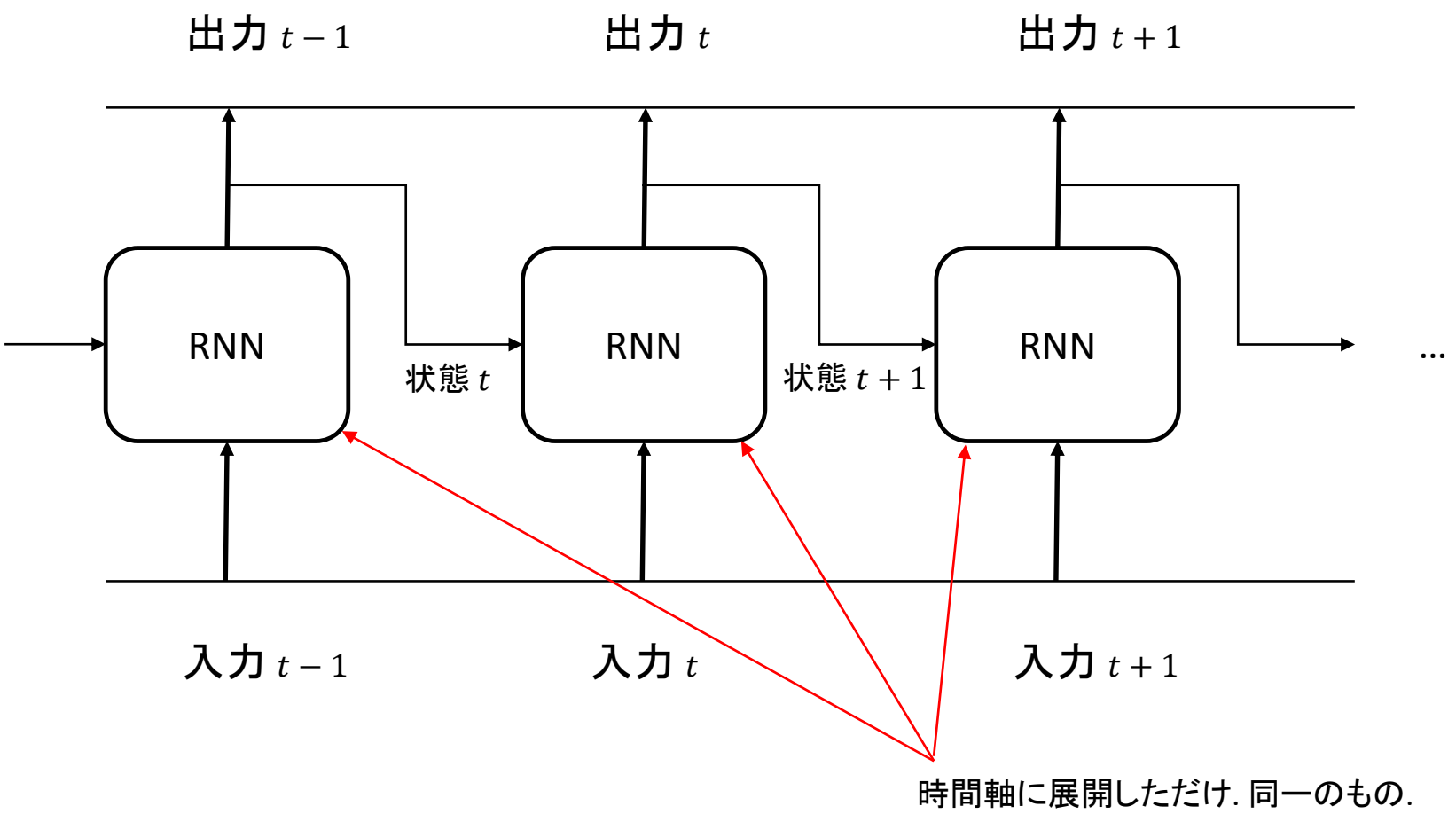

You say Goodbye , and I say Hello.

RNN：

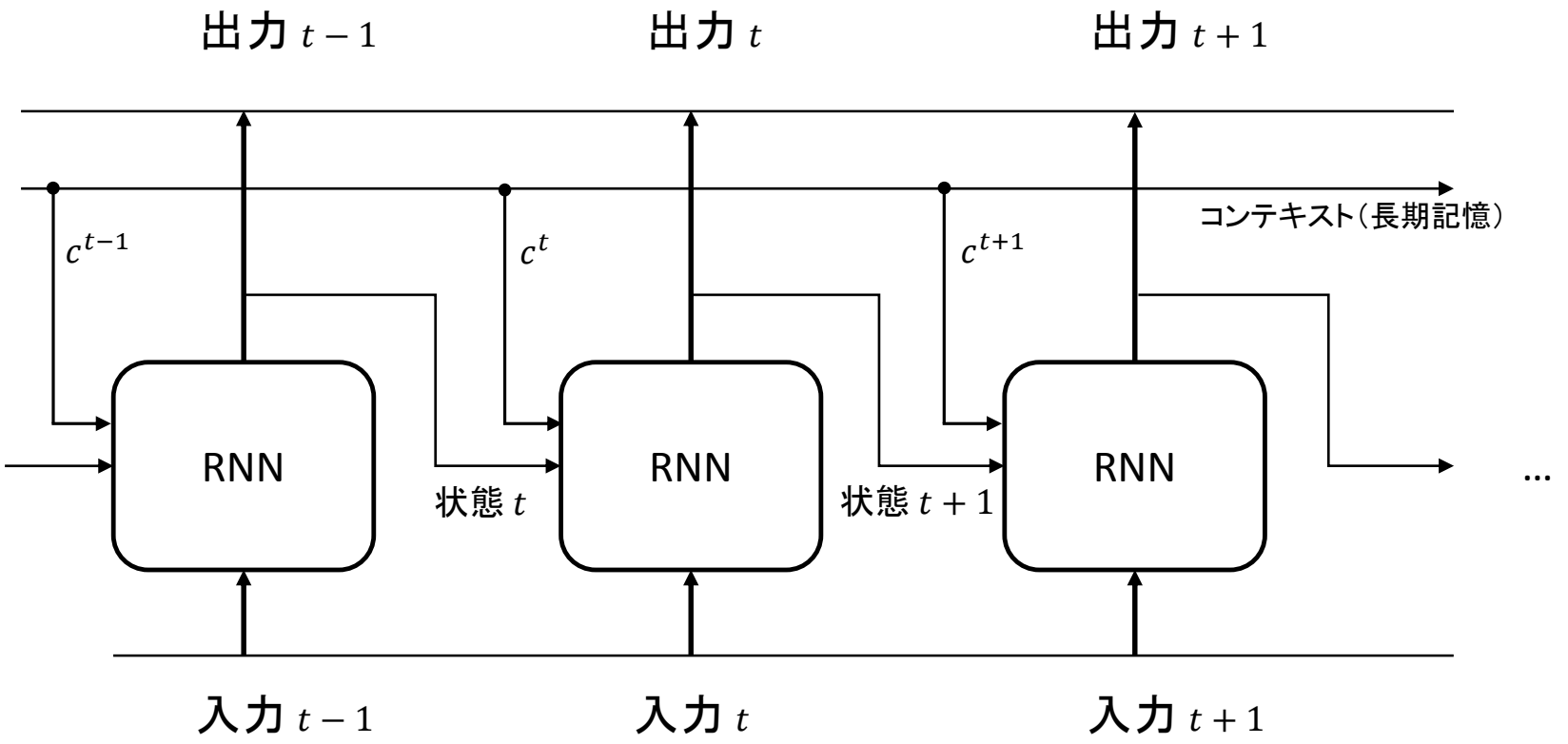



say の前の単語も考慮して推測することができる.

# RNN: FFNN との違い



# LSTM: RNN との違い

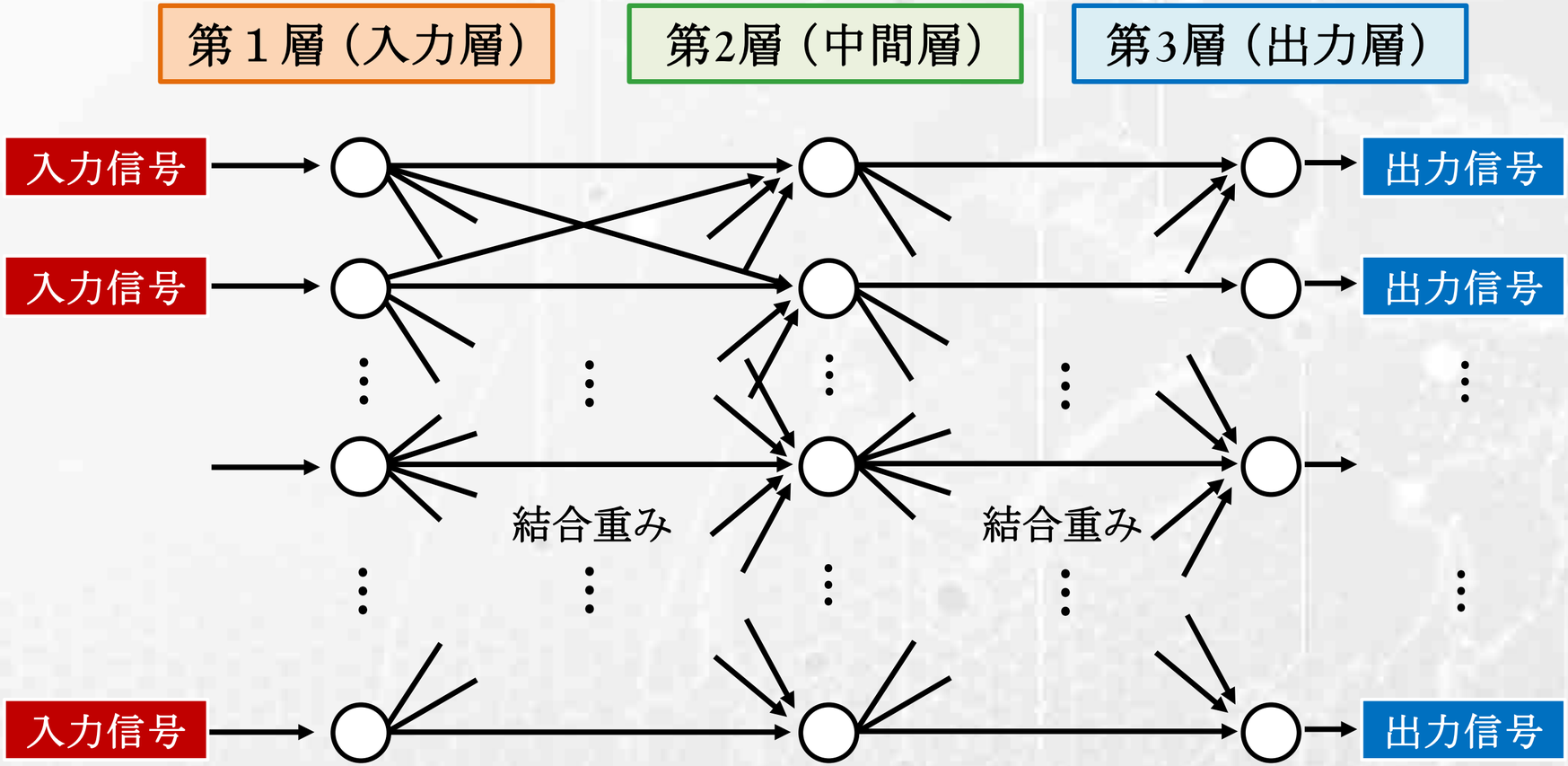



新たに長期記憶、短期記憶の概念を持たせている.

# 深層ニューラルネットの応用研究

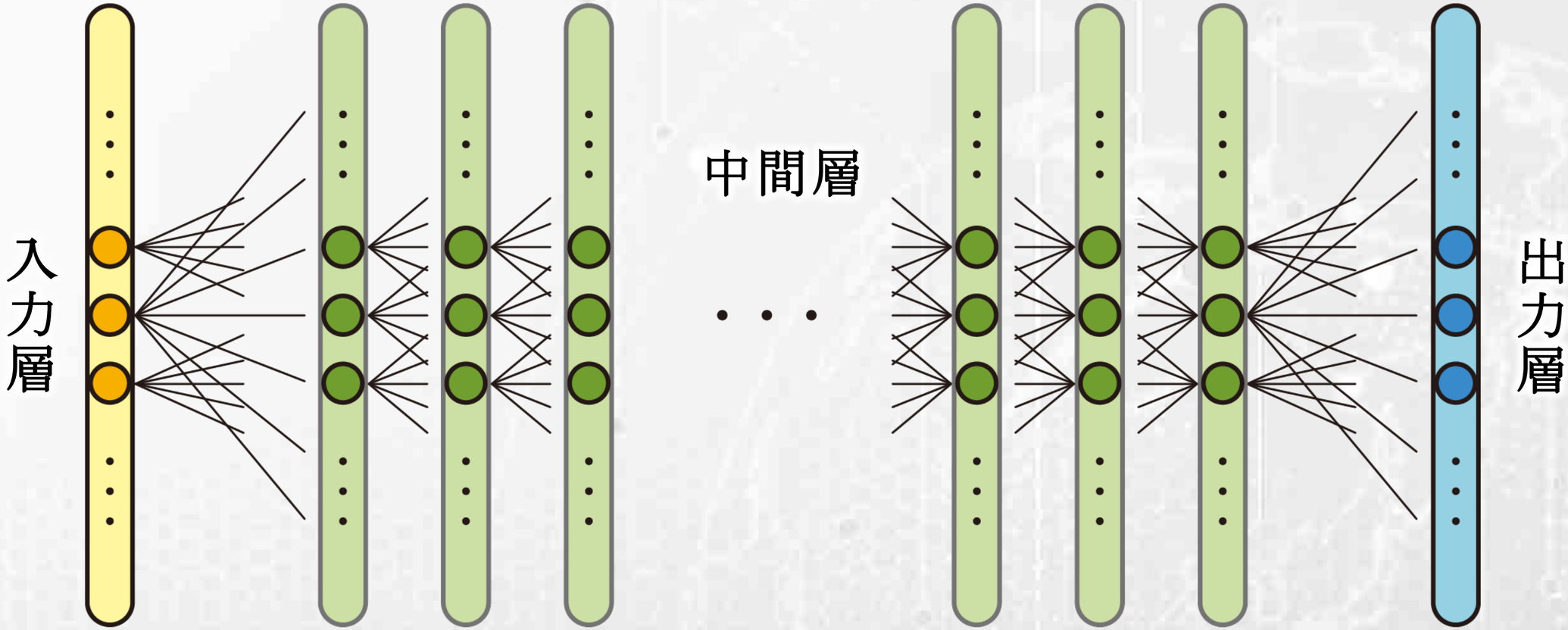
# 入力・中間・出力層の階層化

## 3層構造ニューラルネット

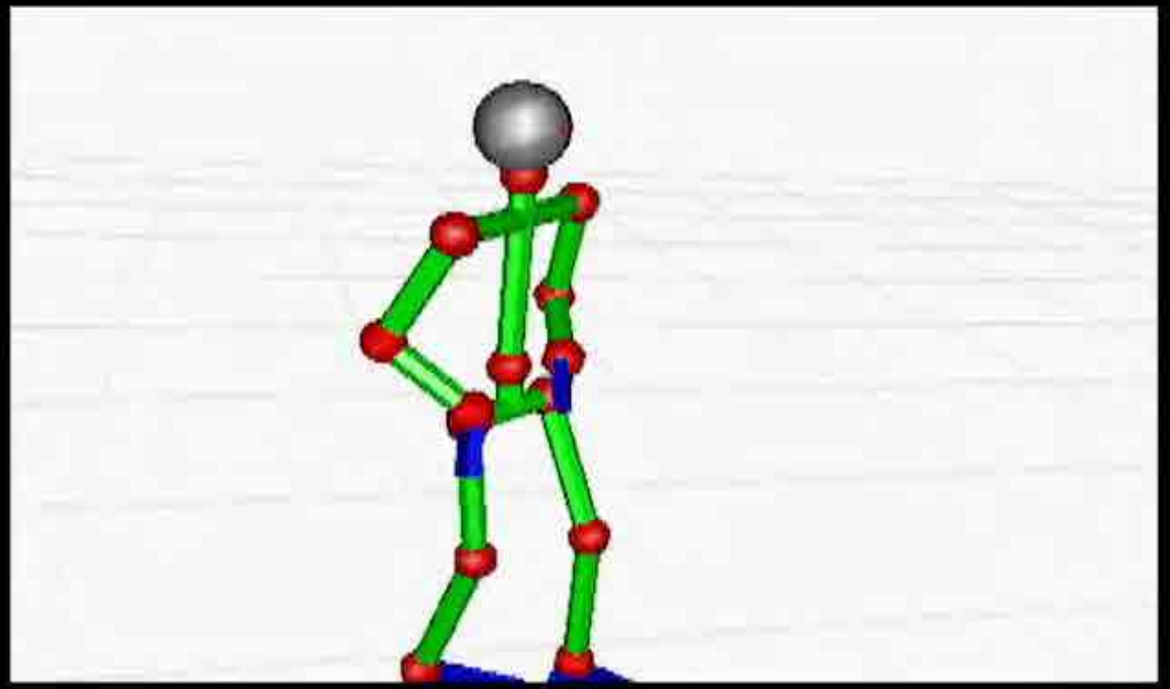


# 深層ニューラルネットワーク

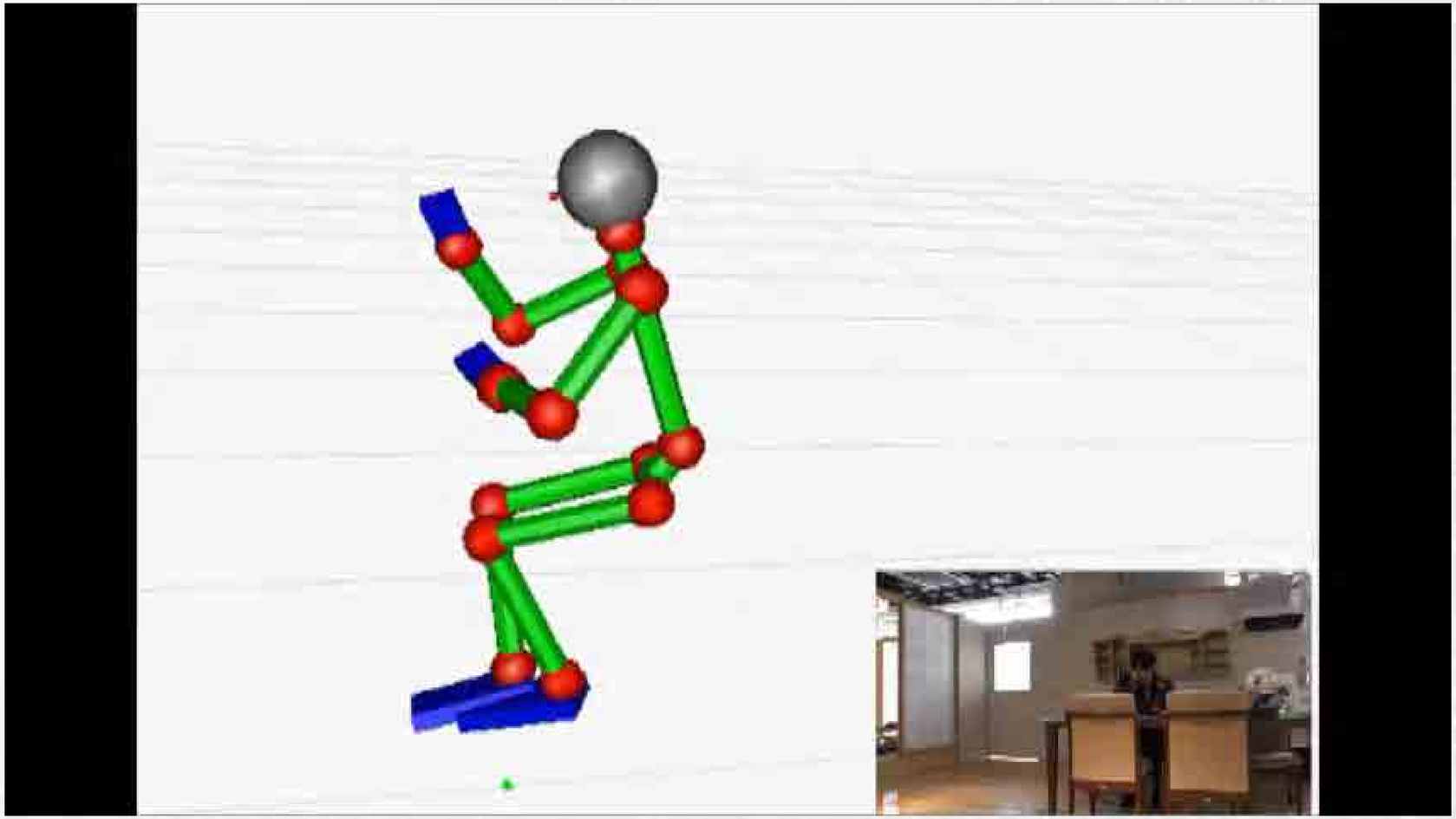
## 深層ニューラルネット



# 身体運動ビッグデータ



# 身体運動ビッグデータ



# 身体運動ビッグデータ


【youtube】

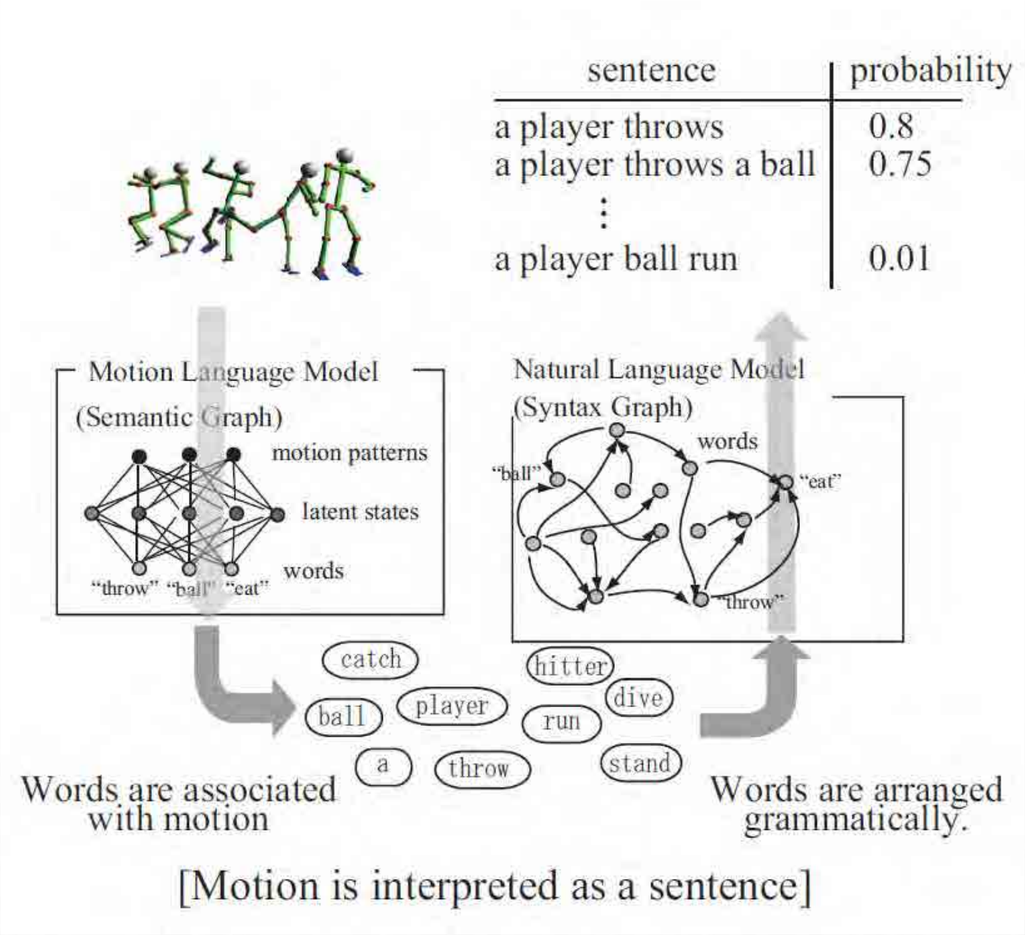
# 身体運動のアノテーション

## クラウドソーシング

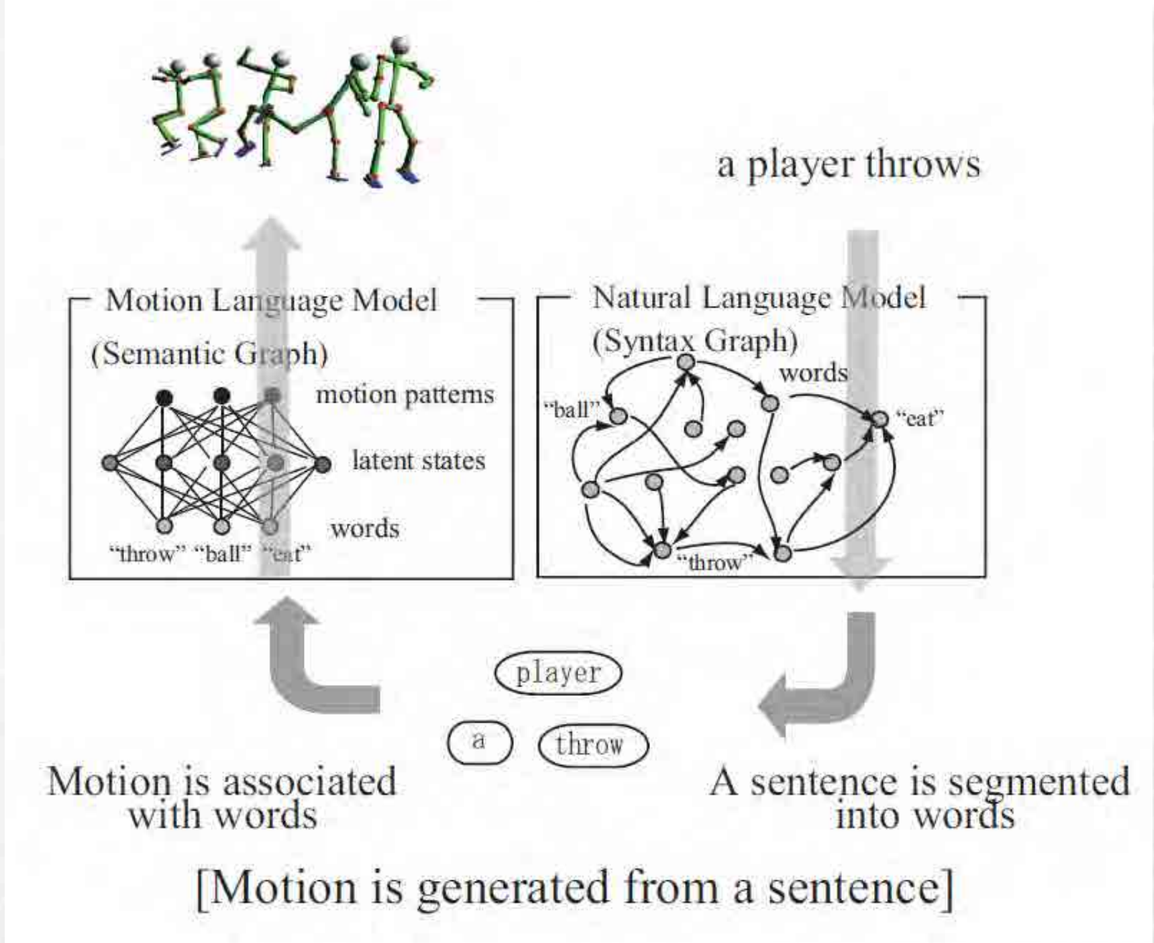
- **群衆（crowd）と**
  業務委託（sourcing）を
  組み合わせた造語

- 不特定多数の人が協力して
  タスクを遂行する枠組み

# 身体運動の言語化知能

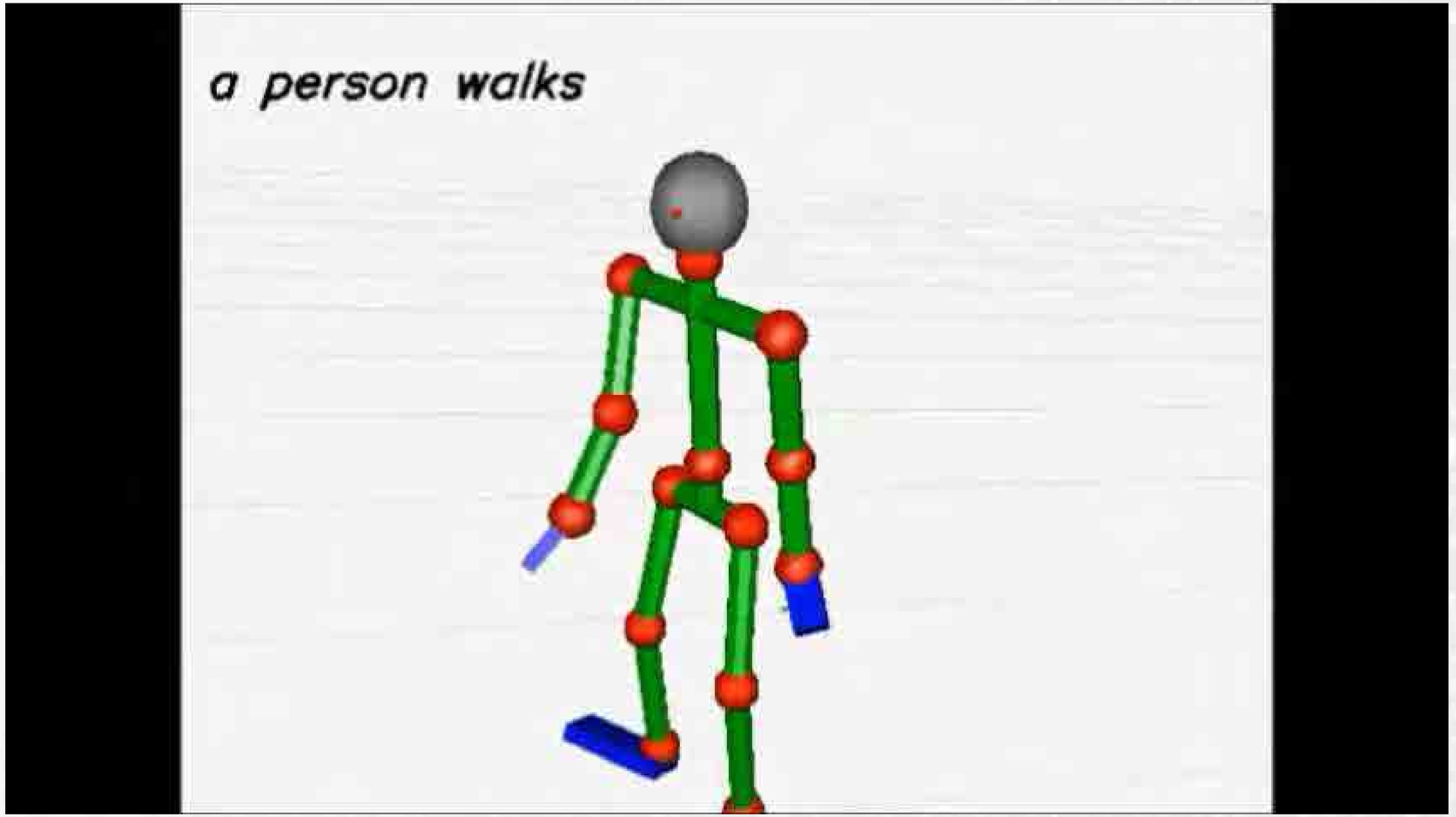



【運動から言語】

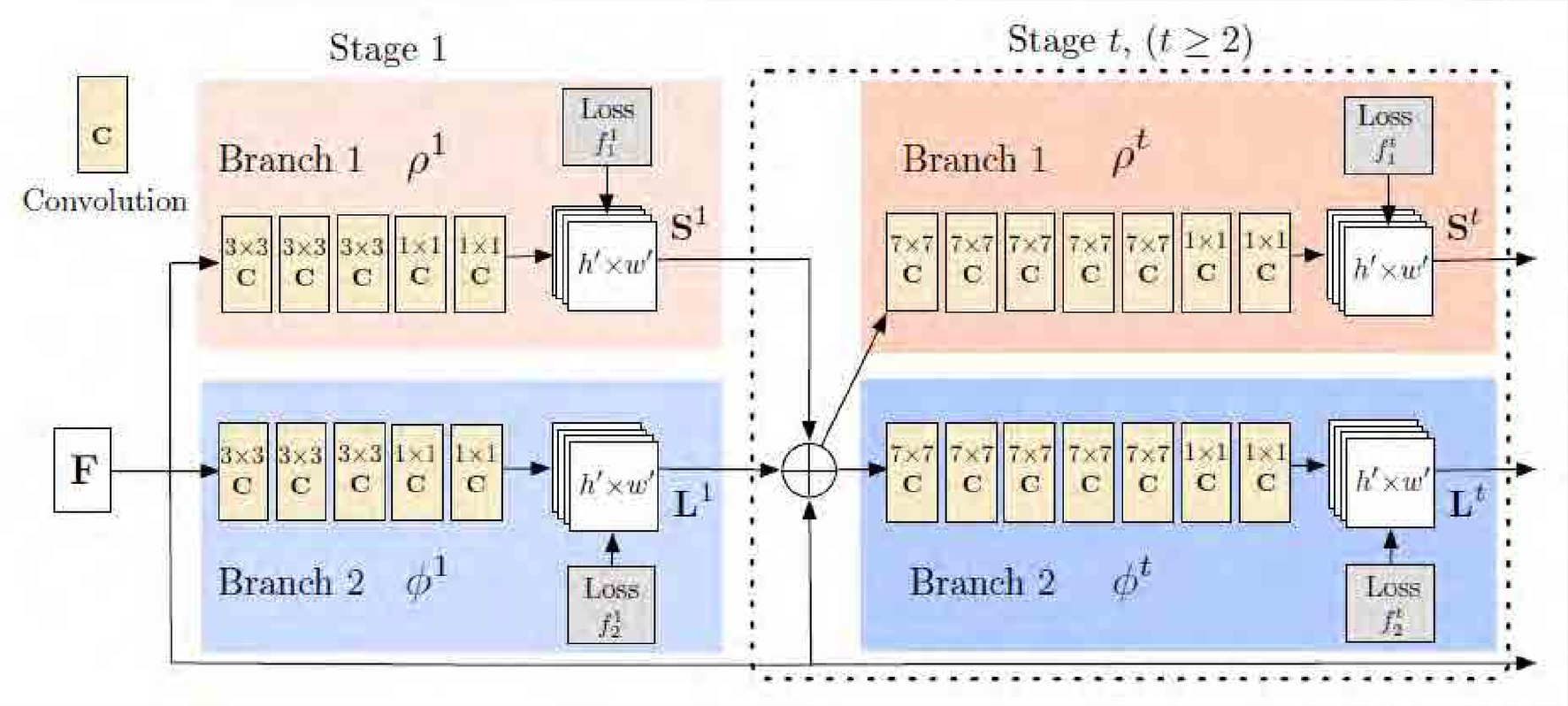



【言語から運動】

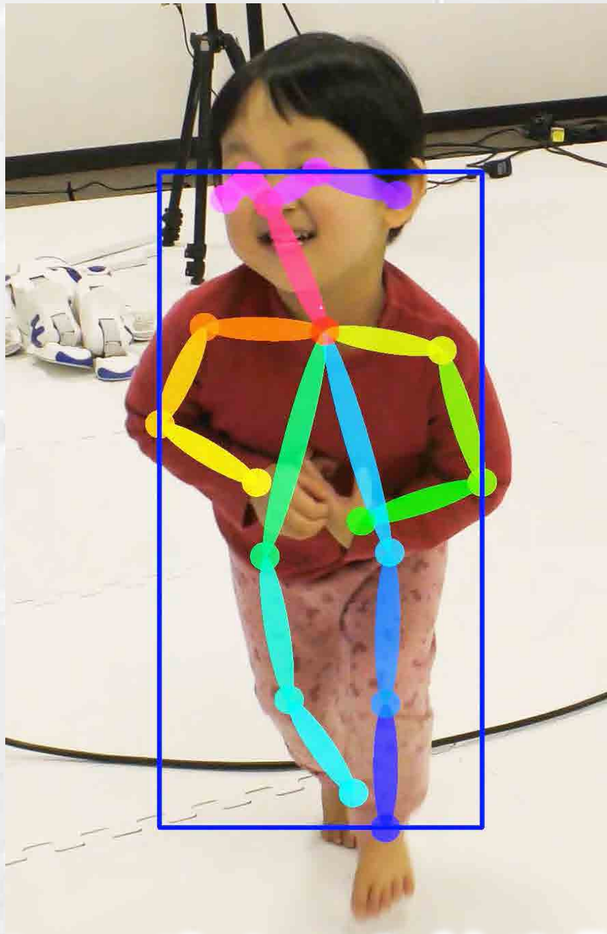
# 身体運動から文章生成



# 画像から2次元姿勢推定

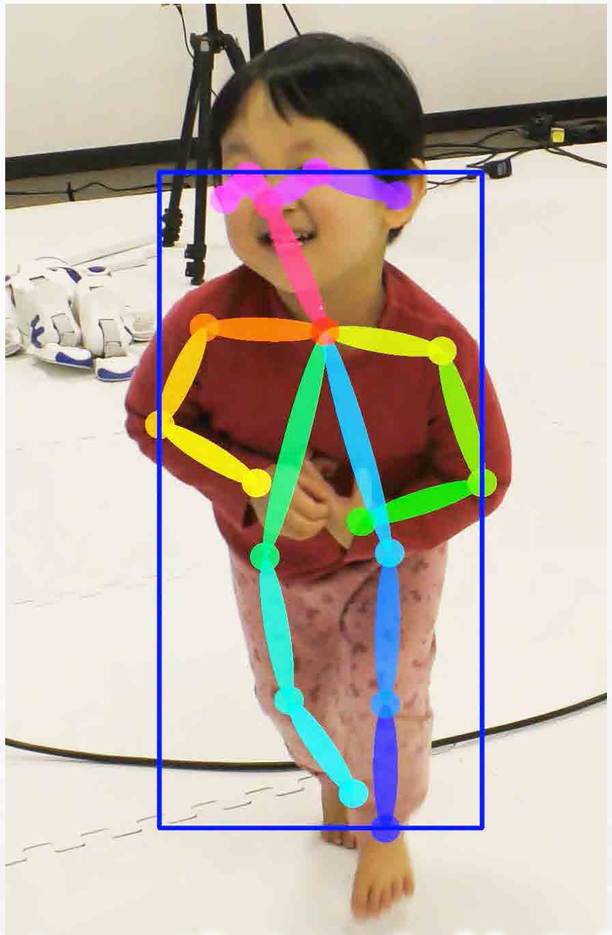



Z.Cao et.al. “Realtime multi-person 2d pose estimation using part affinity fields”



【キーポイント抽出】

# 2次元姿勢の特徴表現

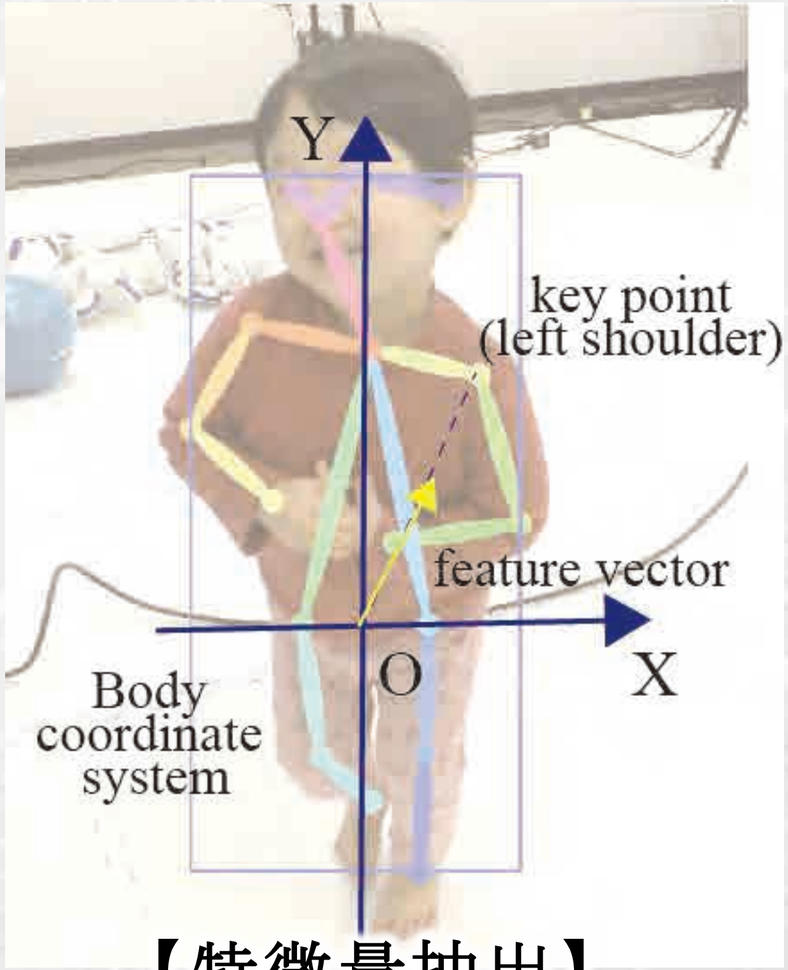


【キーポイント抽出】

身体座標系の設定

身体座標系における
キーポイント２次元位置

2次元位置を連結した
特徴ベクトル

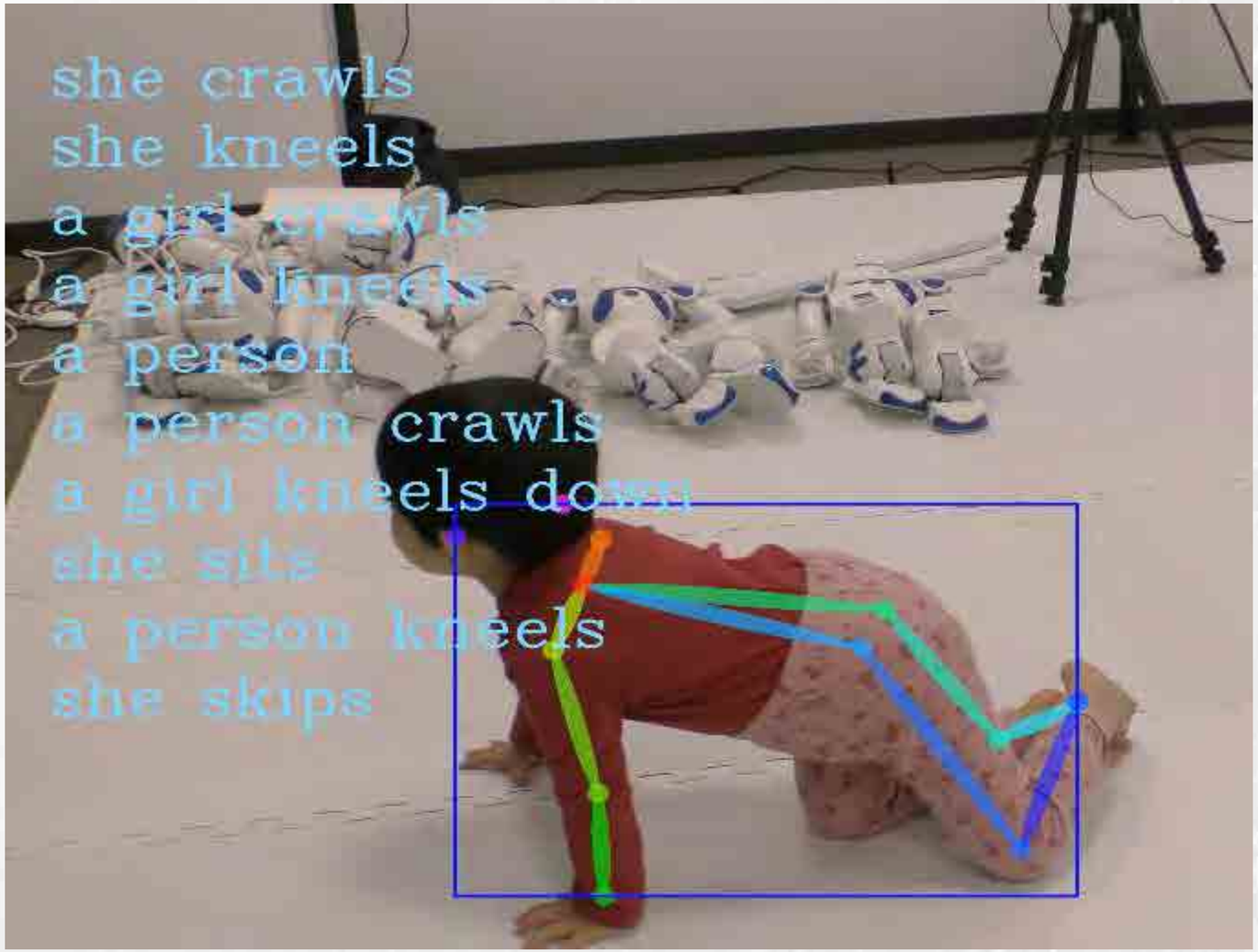



【特徴量抽出】

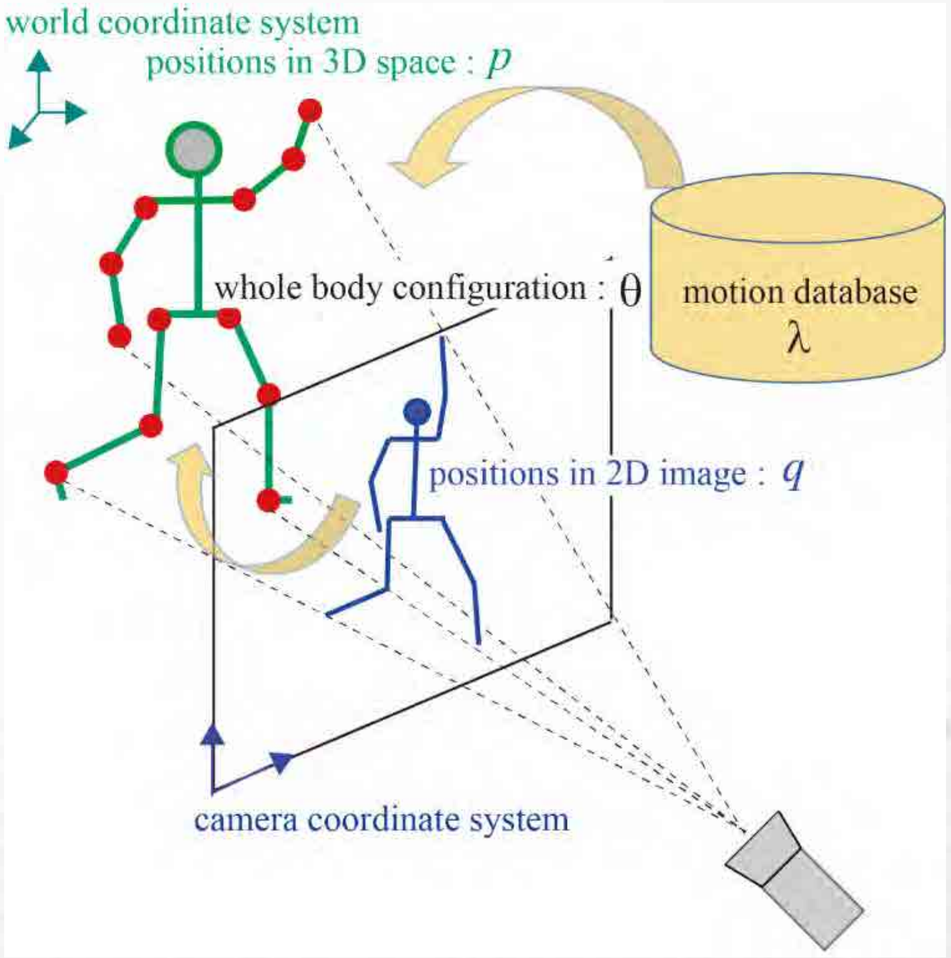
# 介護日誌の自動化


she rolls a wheelchair
she is in a wheelchair
a woman stops
a woman rolls a wheelchair
a woman is in a wheelchair
a woman is wheelchair-mobile
she changes and goes back
she is in a chair
she rolls a wheelchair slowly
she bumps into a woman
TOYOTA

# 保育日誌の自動化


she crawls
she kneels
a girl crawls
a girl kneels
a person
a person crawls
a girl kneels down
she sits
a person kneels
she skips

# 2次元姿勢から3次元身体運動の復元



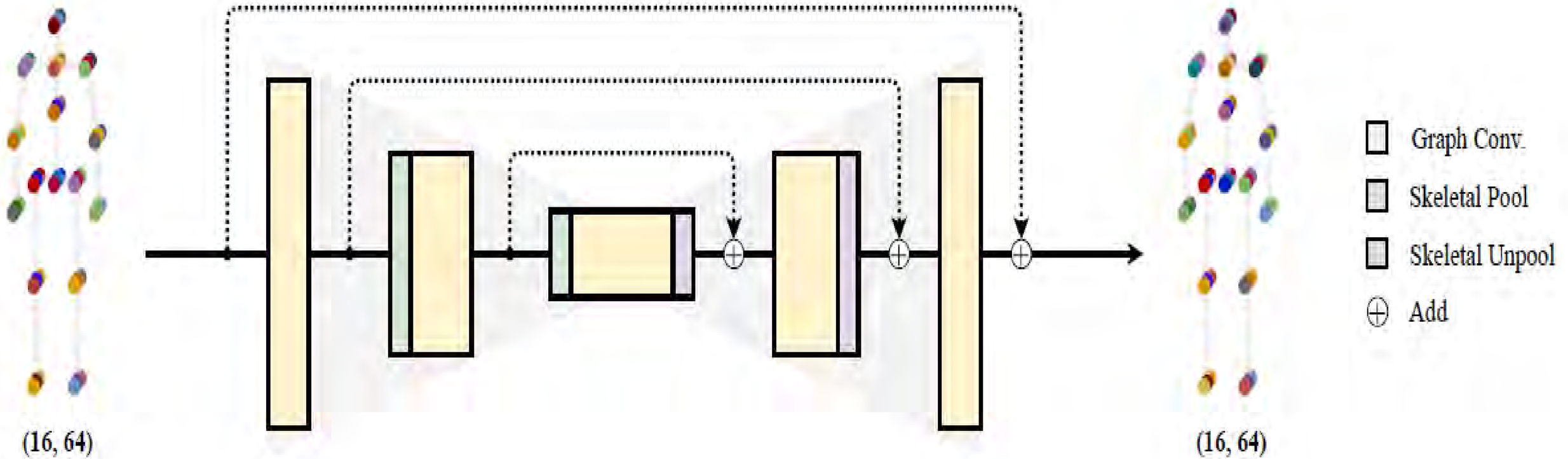

2Dから3Dの復元　【不良設定問題】

カメラ焦点と画像平面上のキーポイントを結ぶ線分上に関節がある。

しかし、線分上には無数の候補点がある。

3D運動データを事前知識として活用

# 2次元姿勢から3次元身体運動の深層学習

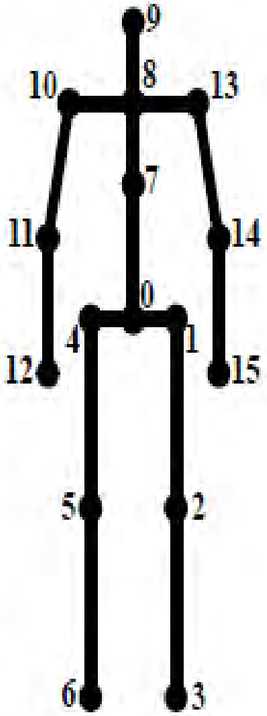


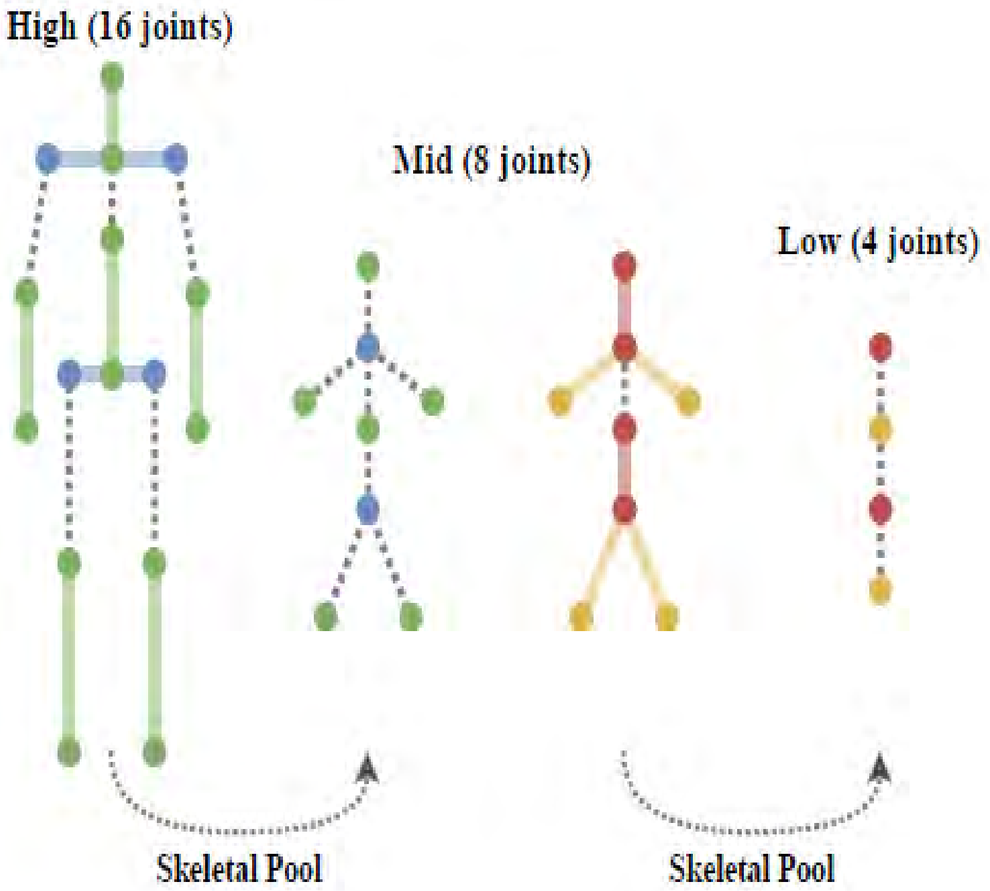

2Dキーポイント

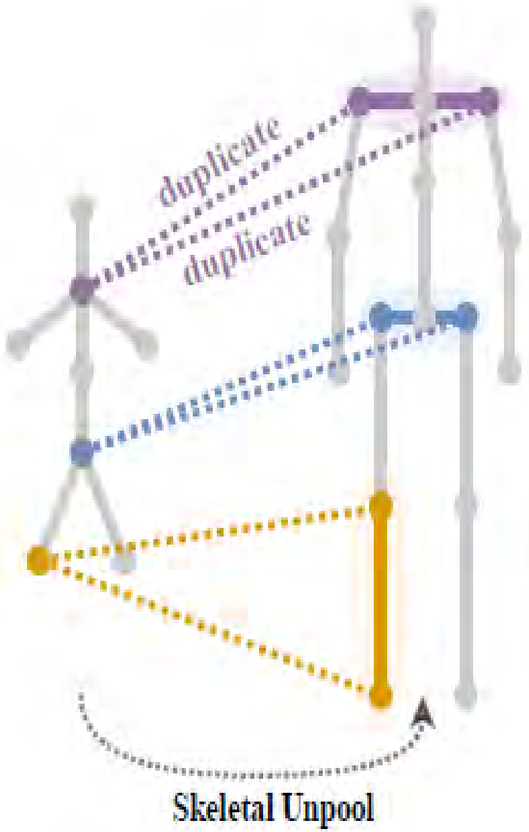
3Dキーポイント

# 身体運動ビッグデータ

0. Pelvis
1. R. Hip
2. R. Knee
3. R. Ankle
4. L. Hip
5. L. Knee
6. L. Ankle
7. Thorax
8. Neck
9. Head
10. L. Shoulder
11. L. Elbow
12. L. Wrist
13. R. Shoulder
14. R. Elbow
15. R. Wrist



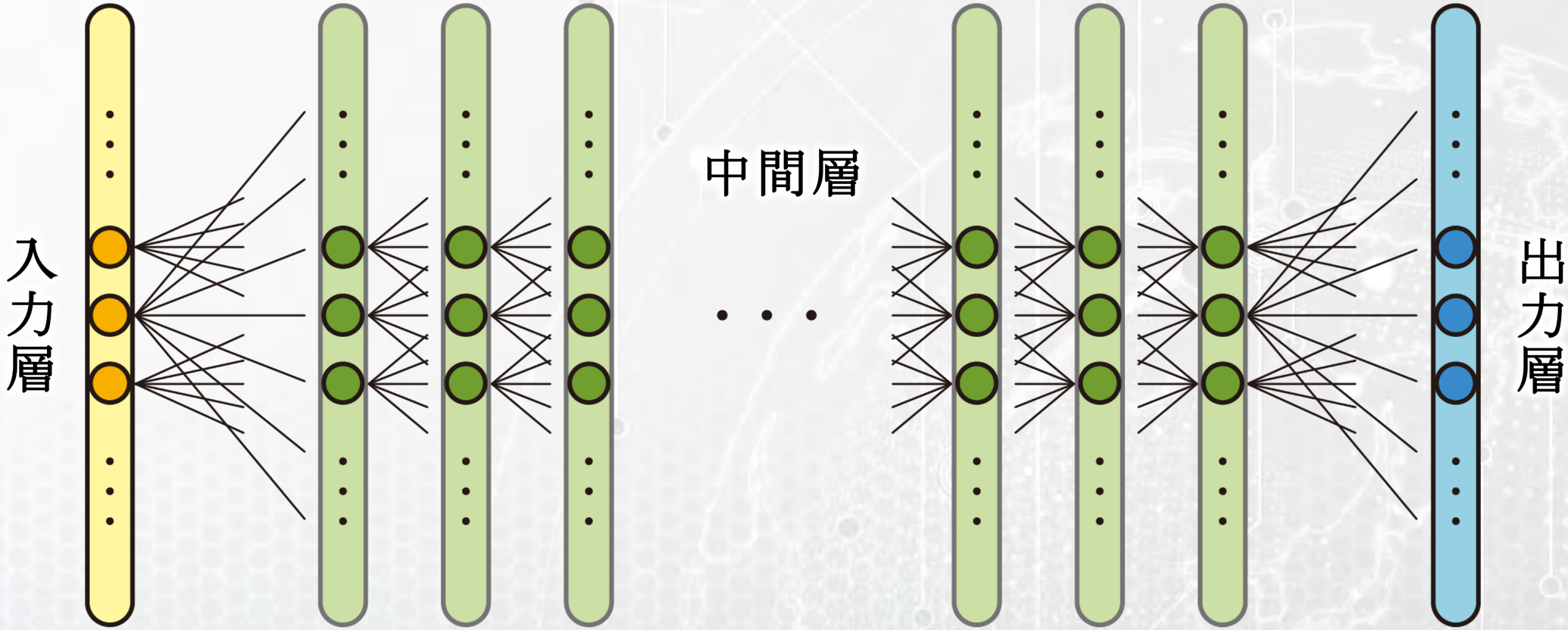



# 身体運動ビッグデータ

# 深層ニューラルネットワーク

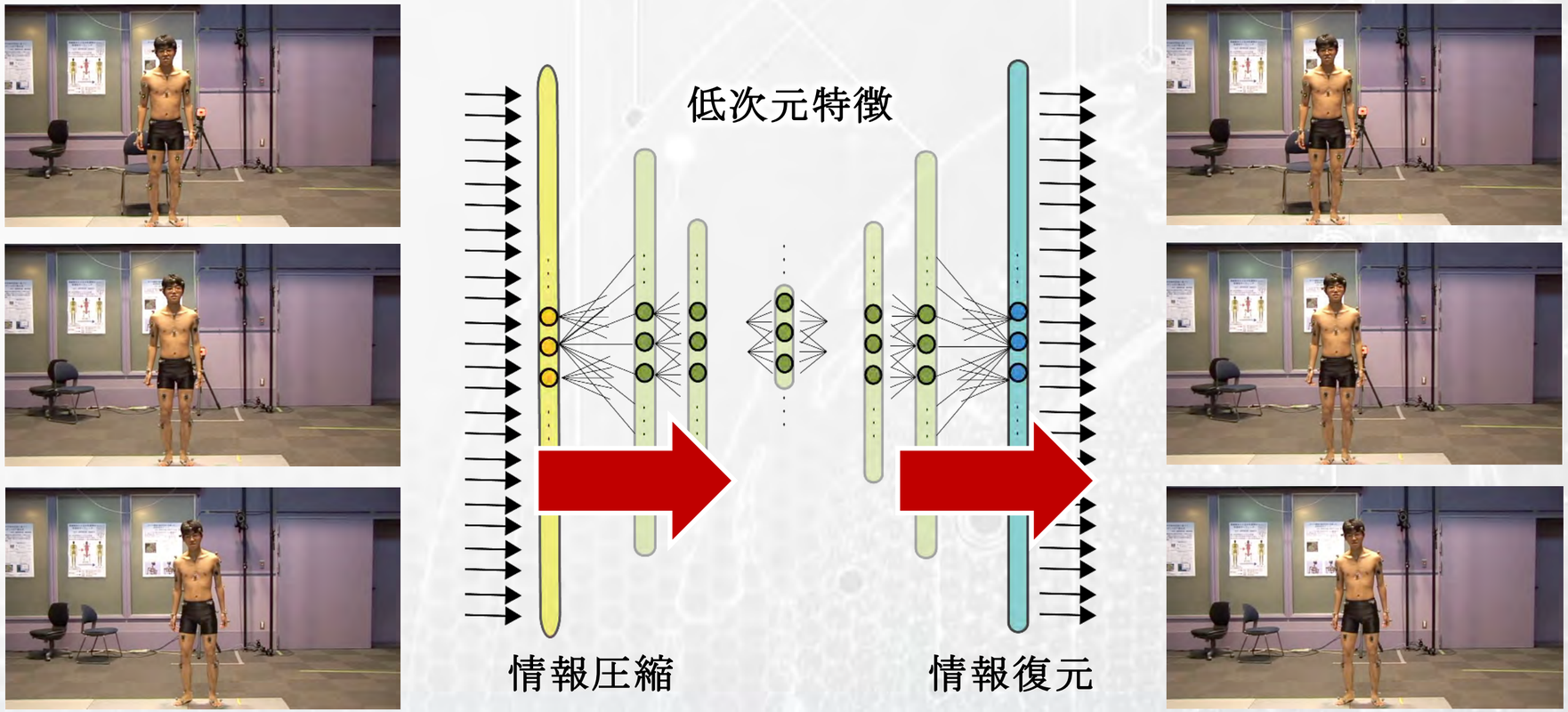
## 深層ニューラルネット



# 深層ニューラルネットワークの情報圧縮

## 深層ニューラルネットの適用例

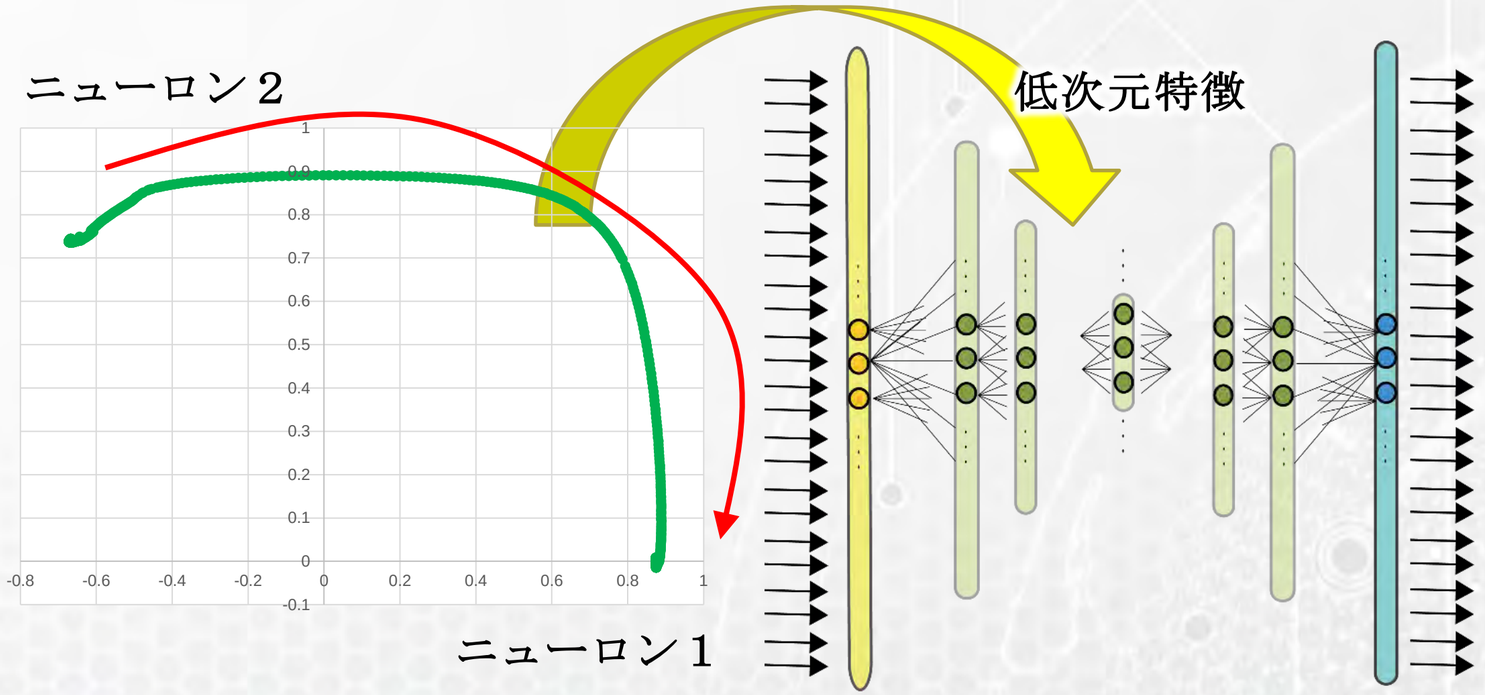
- 入力と出力が同じとなるニューラルネット（オートエンコーダー）

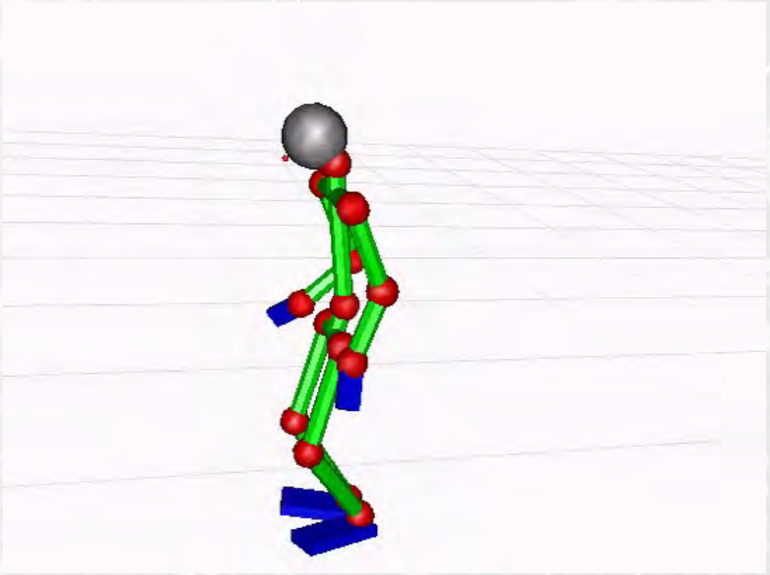


# 深層ニューラルネットワークの情報圧縮

## 深層ニューラルネットの適用例

- 入力と出力が同じとなるニューラルネット（オートエンコーダー）

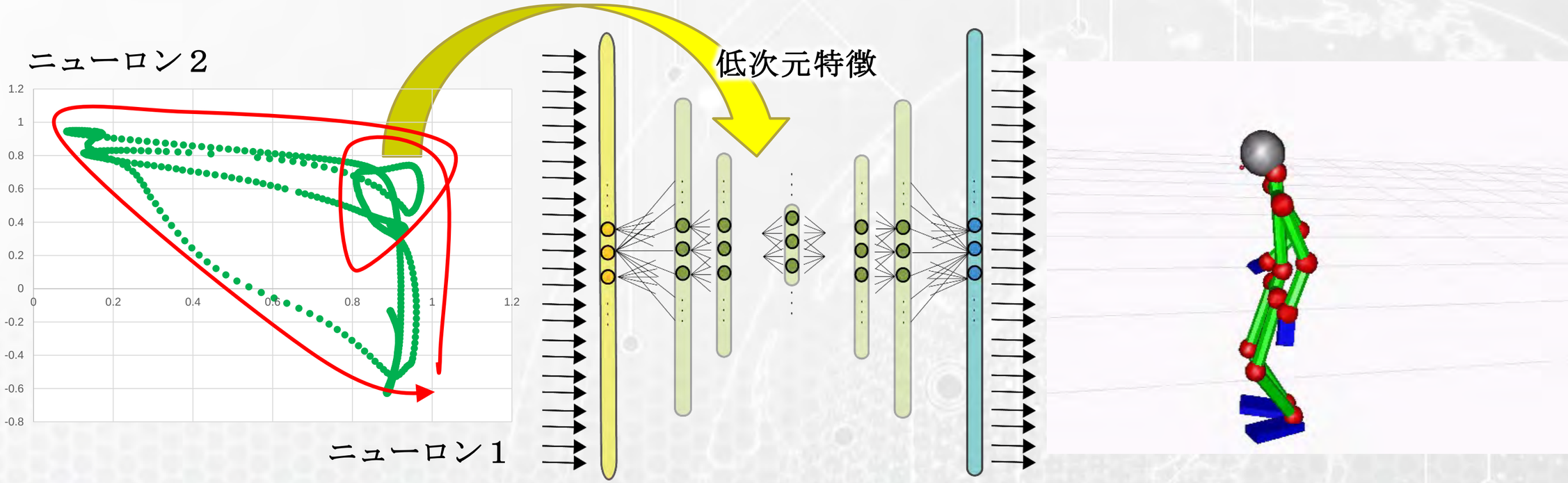


# 深層ニューラルネットワークの情報圧縮

## 深層ニューラルネットの適用例

- 入力と出力が同じとなるニューラルネット（オートエンコーダー）



# まとめ

- ２次元姿勢の推定
  - 画像の深層学習によって、画像集の各画素に関節が存在する確率を計算することによって、画像から2次元姿勢を推定
- ２次元姿勢から３次元運動の復元
  - グラフ埋め込みニューラルネットによって2次元姿勢の情報圧縮（エンコード）圧縮情報から3次元生成の生成（デコード）
- 運動の低次元化と運動生成
  - 運動のオートエンコーダによる大自由度運動の制御
- 運動の言語化
  - クラウドソーシングを通じた運動と言語のビッグデータ
  - 運動データのアノテーションデータから運動の言語化