

# 画像認識, 多層パーセプトロン

M1 齋藤邦章

# アウトライン

- 画像認識のコンセプト、Deep Learningに至るまでの歴史
- Deep Learning (CNN)
- Perceptron (Multilayer Perceptron)

# 画像認識のプロセス

訓練時



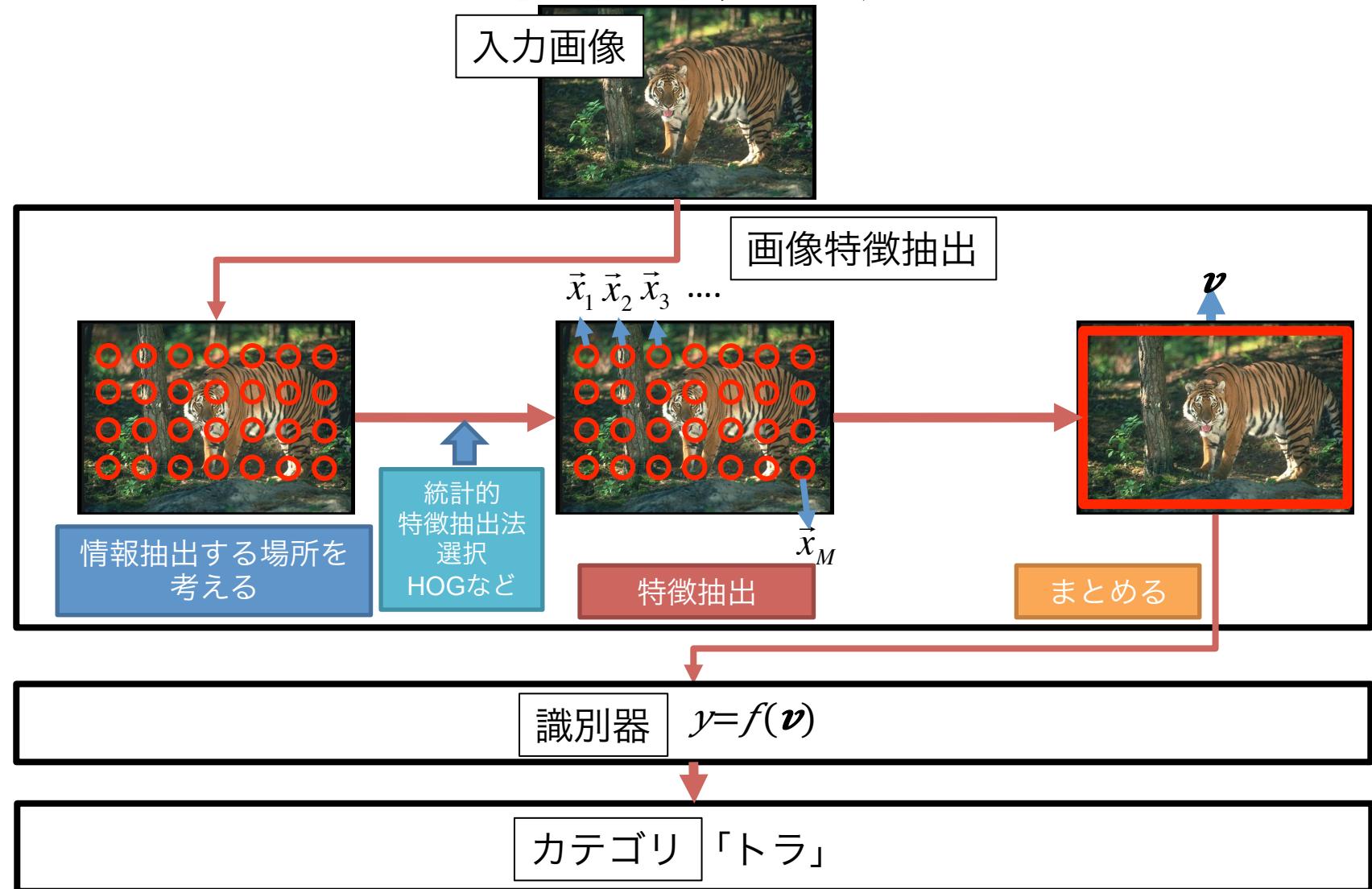
識別時



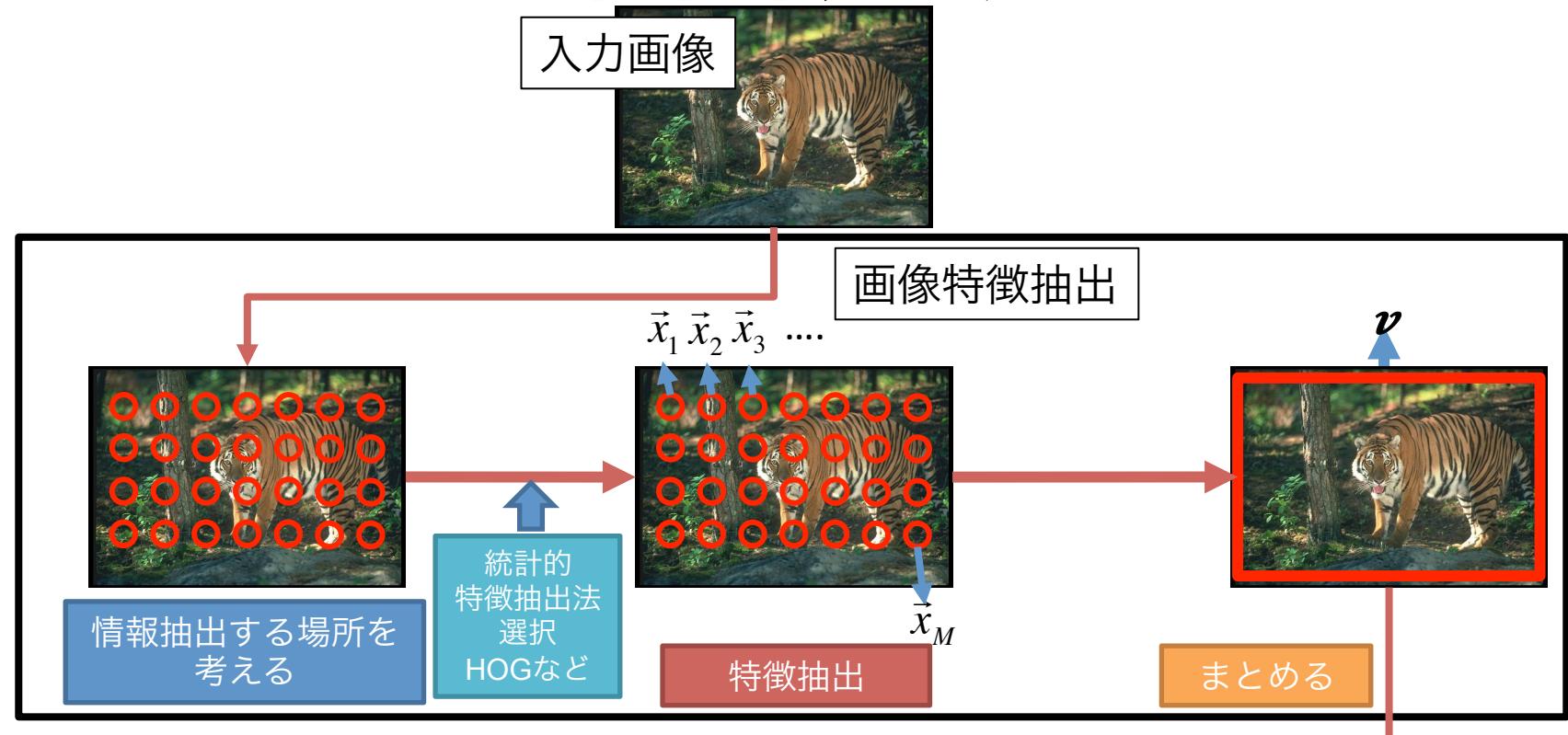
ここが大事！

- 高い質のデータ、特徴抽出が適切に行われていればシンプルなモデルで十分な精度が出る。特徴抽出の性能を向上させたのがDeep Learningだと言える。

# 画像認識の流れ



# 画像認識の流れ



特徴抽出の流れはDeep Learningも、従来の方法も  
基本的には同じ！

# Bag of Visual Wordsによる認識

## 1. 訓練データからCode wordの作成



## 2. 訓練データのBoWを計算

## Classifier

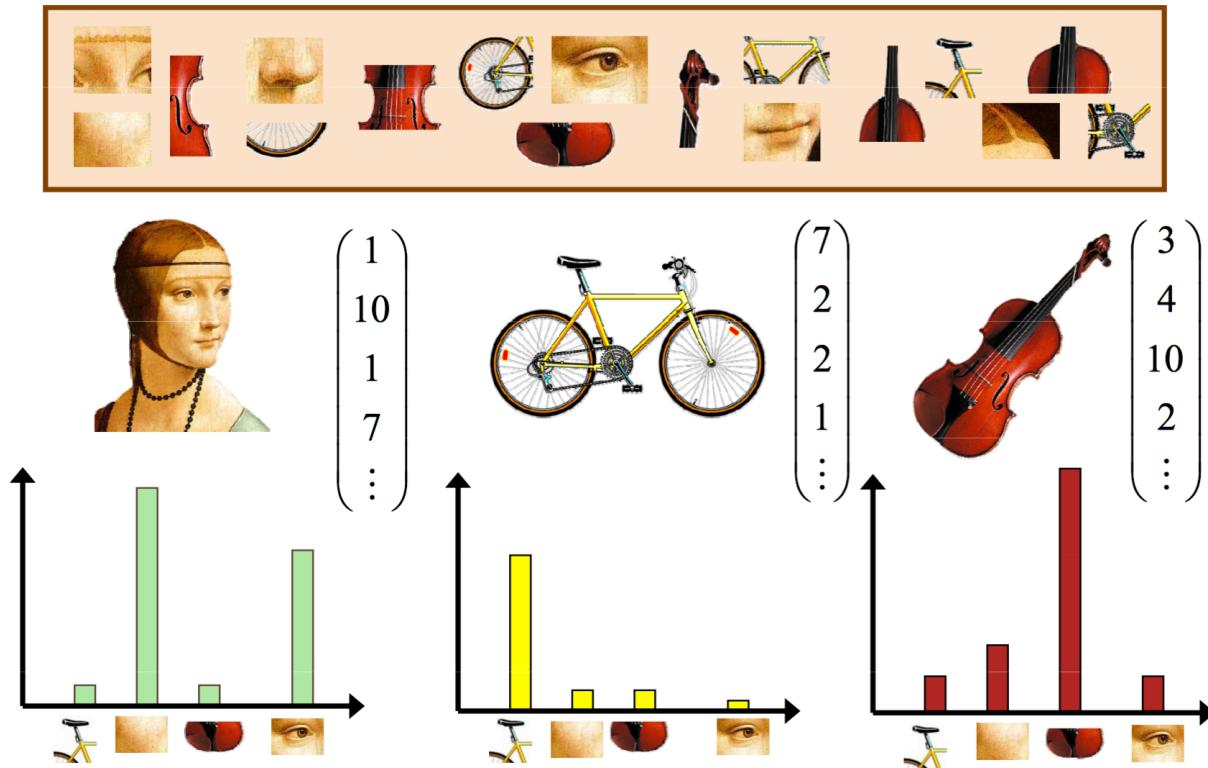
Bonsai, camera, cellphone

### 3. クラス識別器を構築

## 5. クラス識別器で識別

# 従来の特徴抽出

- Bag of Visual Words



# Deep 以前の特徴量

- 1999: SIFT (Local descriptor)
- 2003: Bag of Visual Words (Image representation)
- 2004~ BoVW + (kernel) SVM
  - 画像認識へ自然言語処理のアイデアの適用
- 2005~ BoVW + probabilistic topic model
  - 画像認識へ自然言語処理のアイデアの適用
- 2007~ BoVW + Multiple Kernel Learning
  - 複数の異なる特徴の融合
- 2009~ Sparse coding + Linear SVM
  - 大規模データへの適用
  - 特徴空間での多様体を考慮したコーディング
- 2007~ Fisher vector + Linear SVM
  - 大規模データへの適用
  - カーネル近似

# Deep 以前の特徴量

- 1999: SIFT (Local descriptor)
- 2003: Bag of Visual Words (Image representation)
- 2004~ BoVW + (kernel) SVM

これらに共通して言えるのは  
Hand-craftedな特徴量(人間が設計した指針による特徴量)を  
特徴量として用いているということ！

- 大規模データへの適用
- 特徴空間での多様体を考慮したコーディング
- 2007~ Fisher vector + Linear SVM
  - 大規模データへの適用
  - カーネル近似

# Deep Learning の台頭

- ILSVRC2012 Object Recognition Task



Team name	Error	Description
SuperVision	15.315%	Using extra training data from ImageNet Fall 2011 release
SuperVision	16.422%	Using only supplied training data
ISI	26.602%	「ケタ」が違う scores from classifiers using each FC
ISI	26.646%	Naïve sum of scores from classifiers using each FV
ISI	26.952%	Naïve sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV and CSIFT+FV, respectively
OXFORD_VGG	26.979%	Mixed selection from High-Level SVM scores and Baseline Scores, decision is performed by looking at the validation performance.
...	...	...

# Deep learning の頭

- ILSVRC2012

悔しいです！



ディープ  
ラーニング

長年の  
特徴量設計  
の工夫

Team name	Error	Description
SuperVision	15.315%	Using extra training data from ImageNet release
SuperVision	16.422%	Using only supplied training data
ISI	26.602%	「ケタ」が違う scores from classifiers
ISI	26.646%	Naïve sum of scores from classifiers
ISI	26.952%	Naïve sum of scores from each classifier: LBP+FV, GIST+FV and CSIFT+FV, respectively.
OXFORD_VGG	26.979%	Mixed selection from High-Level Features and Baseline Scores, decision is performed based on the validation performance.
...	...	...



# 従来の特徴量との比較

伝統的方法論  
("Shallow" learning)



人手で設計

人手で設計/教師なし学習

Low-level image feature

SIFT, HOG,  
SURF, etc.

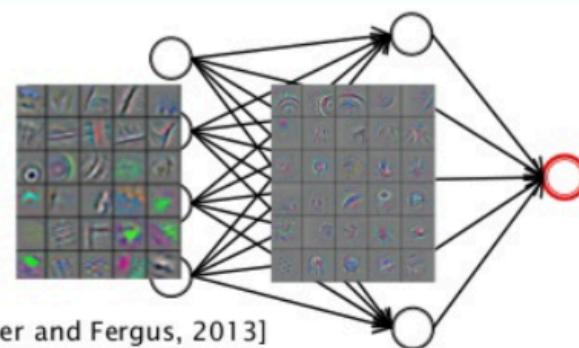
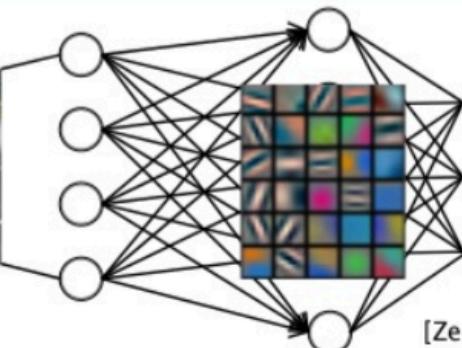
Mid-level image feature

BoVW, VLAD,  
Fisher Vector, etc.

Supervised Classifier:  
SVM, Logistic Regression, etc.

"Car"

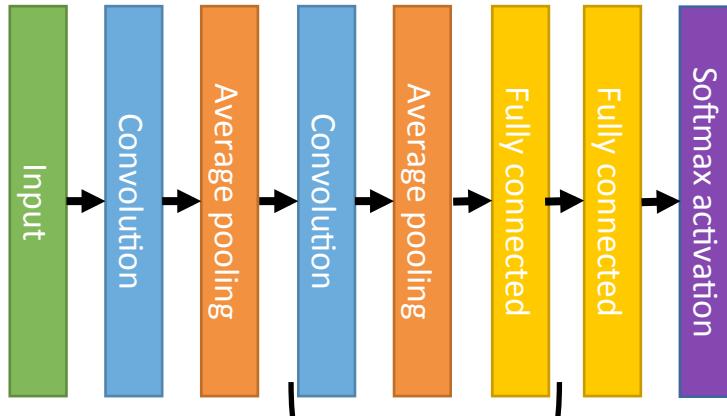
Deep learning



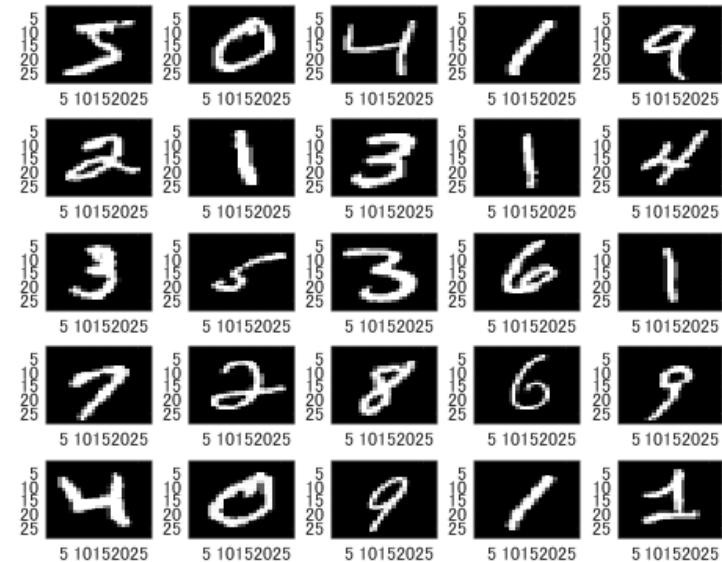
"Car"

生の画素値から、識別に至る階層構造を直接的に学習  
従来の特徴量に相当する構造が中間層に自然に出現

# Convolutional Neural Networks (CNN)

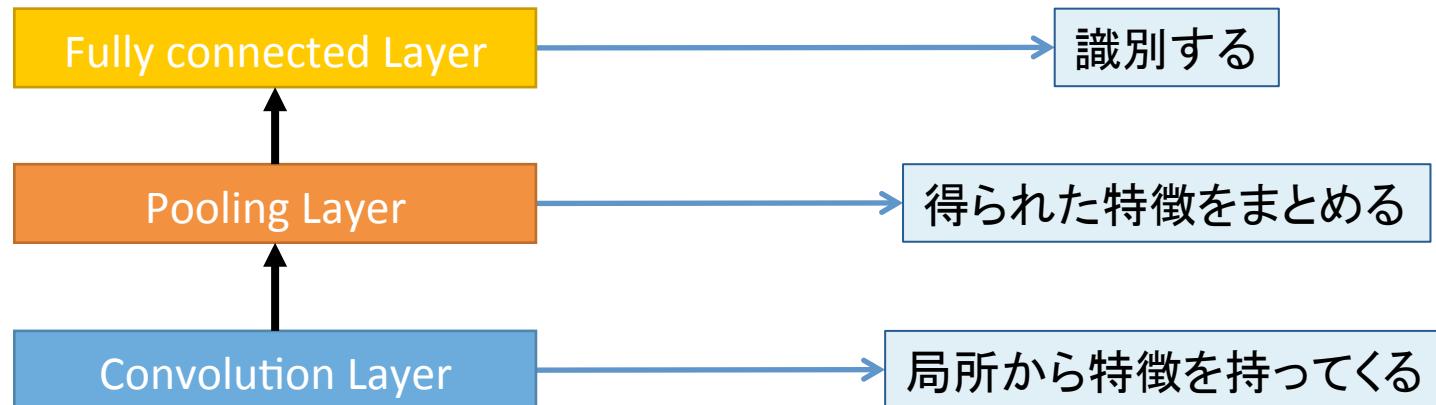


(a) LeNet-5, 1989

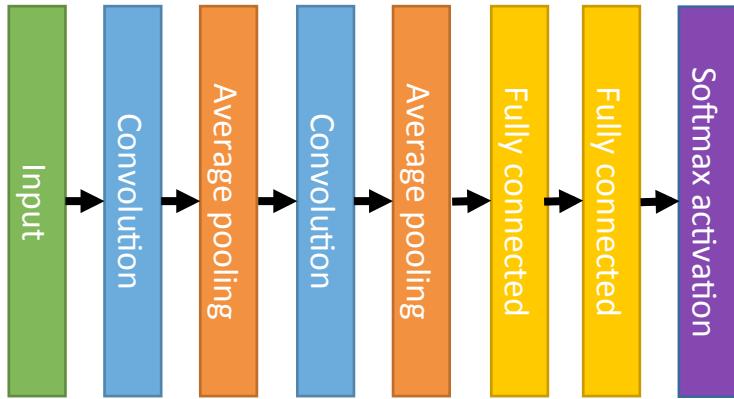


Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. NIPS 1989.

## CNNの要素



# Convolutional Neural Networks (CNN)



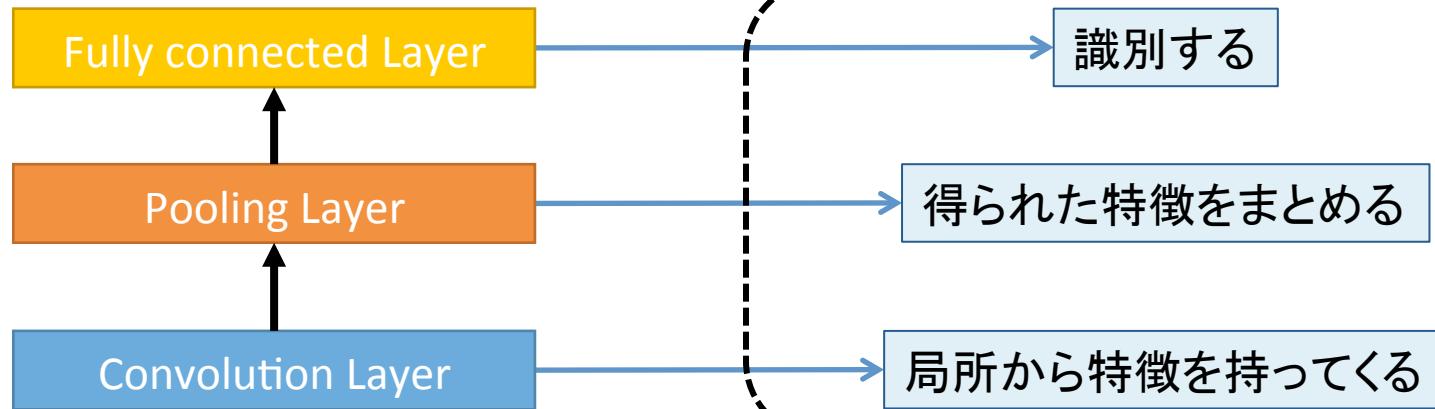
(a) LeNet-5, 1989

やりたいことは同じ。  
全部の機構を、上手く結合させて  
エンドツーエンド(一気通貫)の形で  
学習させることで、Hand-crafted でない識  
別的な特徴を獲得した



Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. NIPS 1989.

## CNNの要素



# なぜ流行るようになったか

- データセット
  - Facebook, FlickrなどWebから大量に画像を得られるようになった
- 計算機の進歩
  - GPUを利用した高速な計算
- 学習手法の進歩
  - Dropoutなど

# 私が選ぶ Deep Learning四天王



# プロフィール紹介

Hinton



- ・Deep Learningの祖
- ・Dropout の開発者
- ・四天王の中でも最強
- ・Tronto→Google

Bengio



- ・多分半端ない。
- ・べんじょー
- ・最近会社作ったらしい
- ・弟もすごい研究者

Lecun



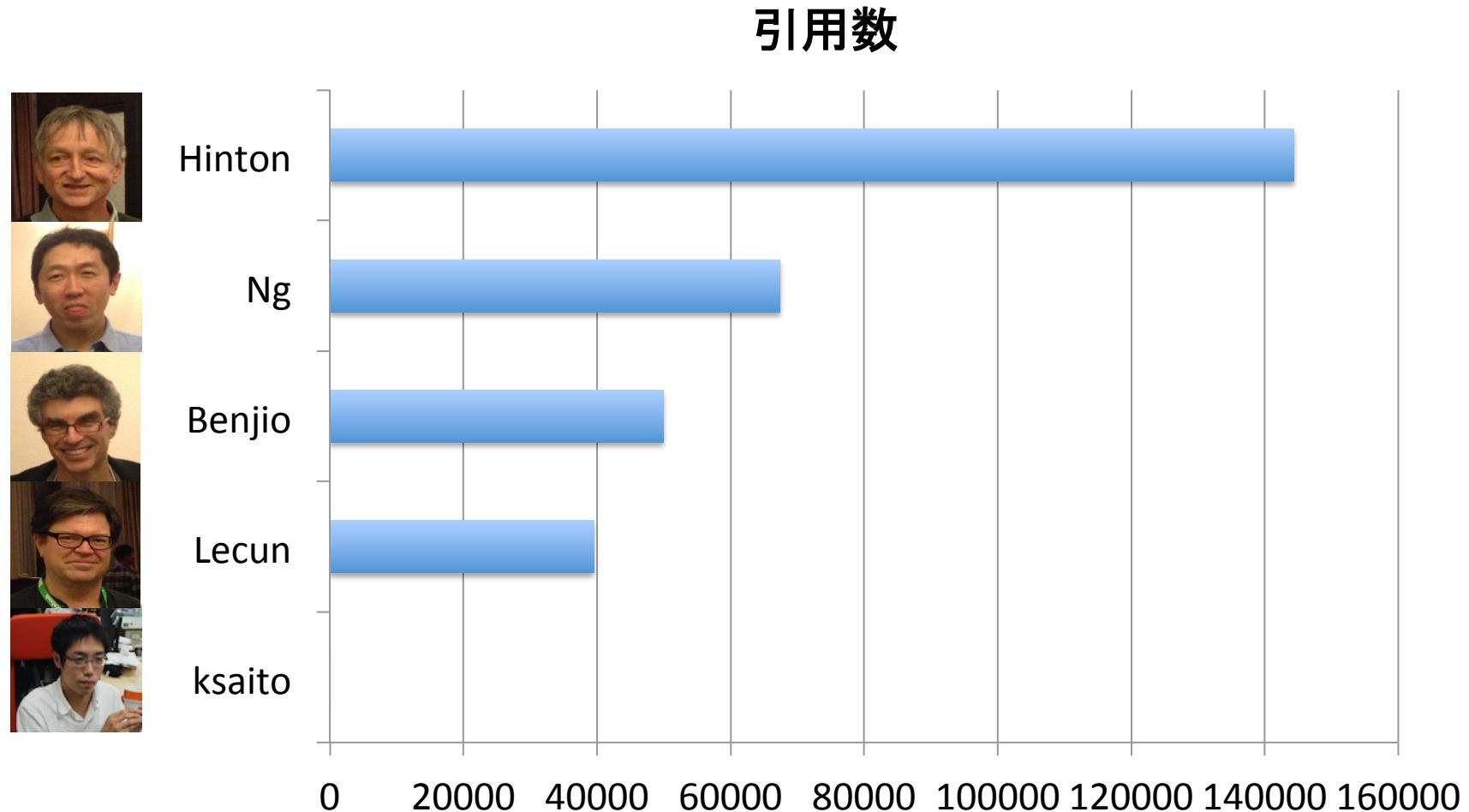
- ・CNNを作った
- ・Hintonの弟子
- ・メガネがオシャレ
- ・Facebook AI Research
- ・弟子入りしたい

Andrew Ng

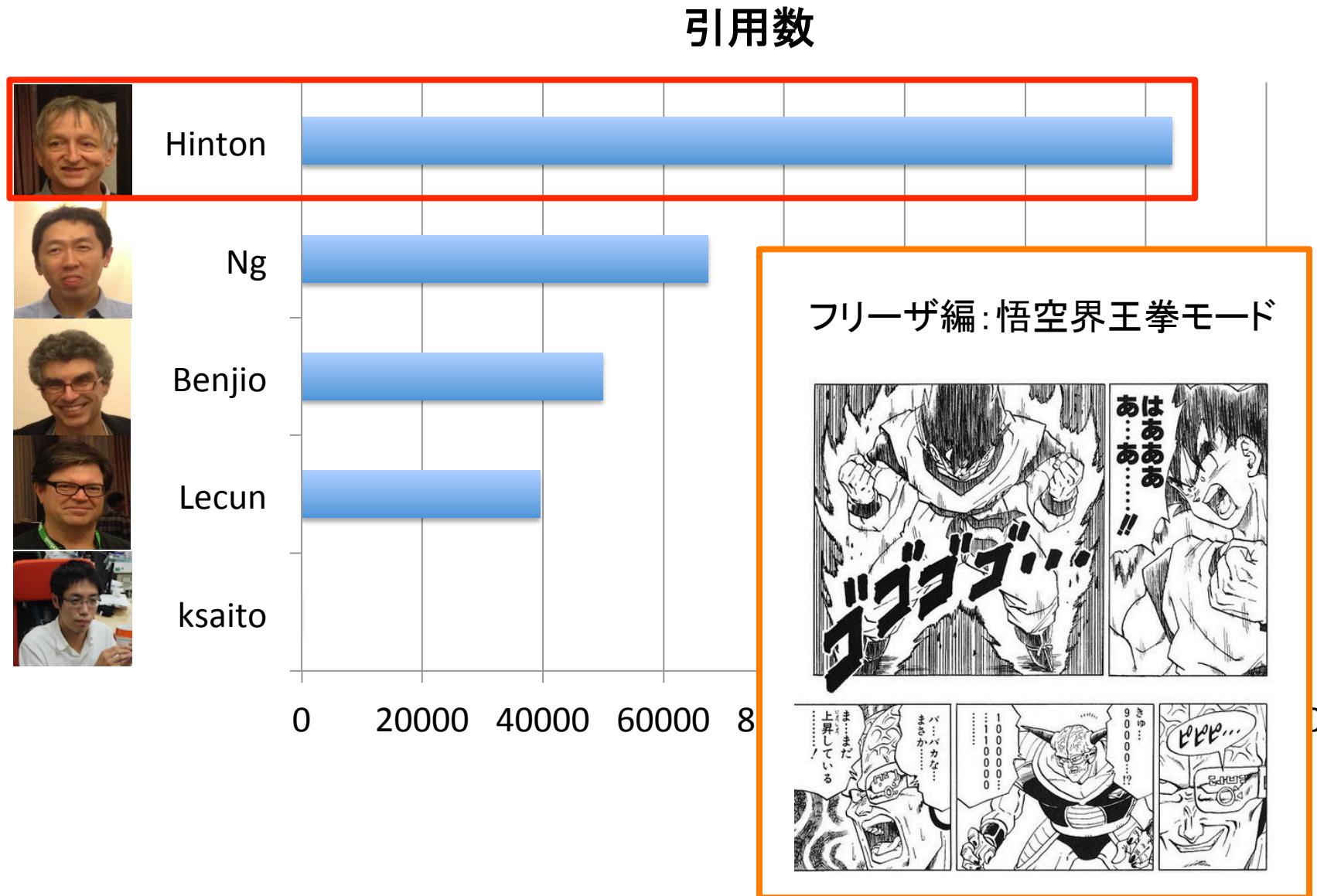


- ・Courseraの機械学習コースで有名
- ・BaiduのChief Scientist
- ・40歳(ちなみに原田先生は43歳くらい)
- ・ROS開発者の一人?

# 引用数でみる四天王の強さ



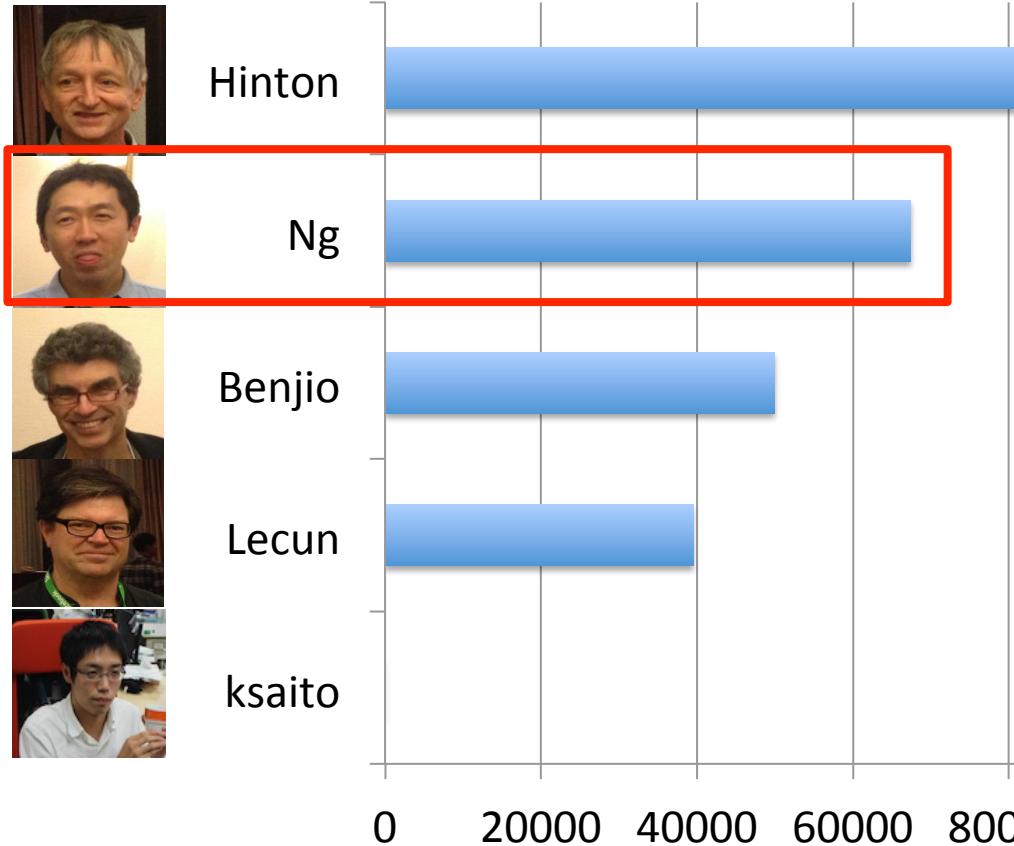
# ドラゴンボールからみる四天王の強さ



ドラゴンボール第11章より

# ドラゴンボールからみる四天王の強さ

引用数

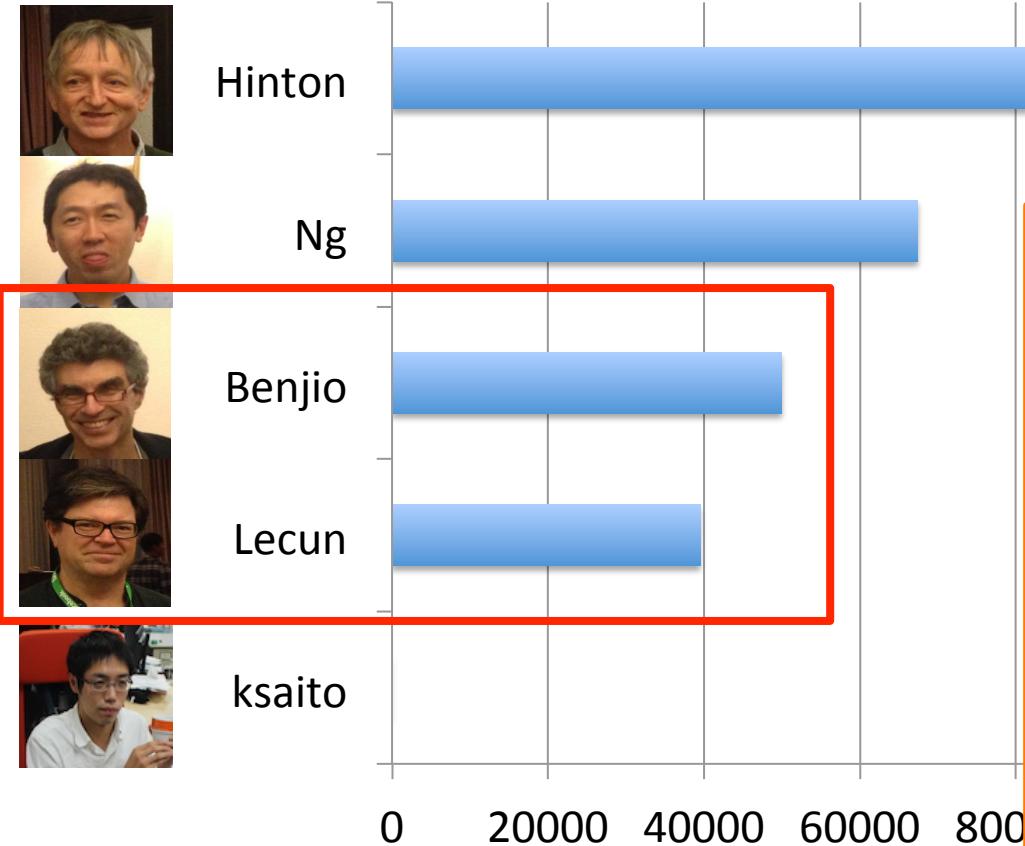


フリーザ編：悟空通常モード



# ドラゴンボールからみる四天王の強さ

引用数

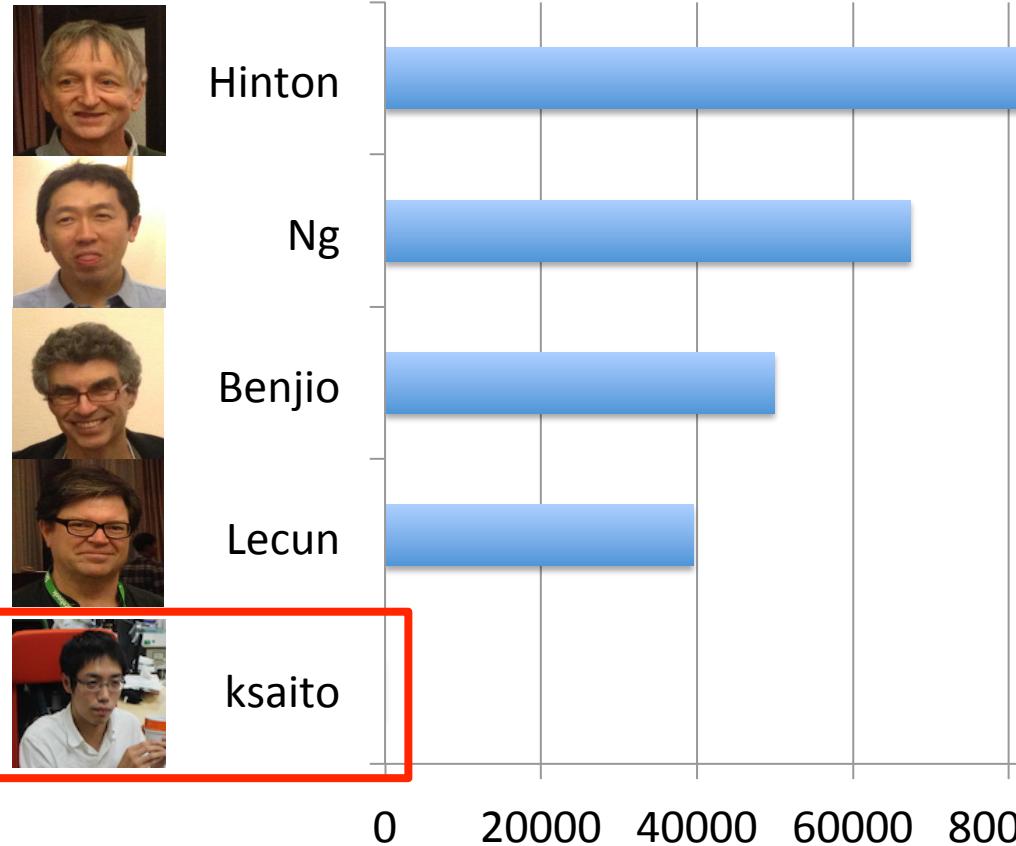


フリーザ編：ネイル（ナメック星人）

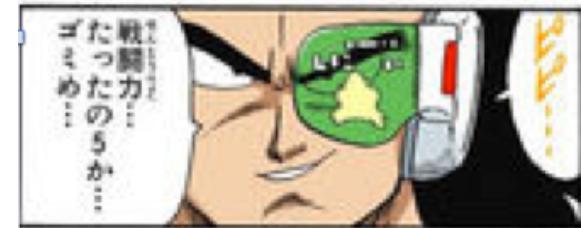


# ドラゴンボールからみる四天王の強さ

引用数



引用数5: 農夫



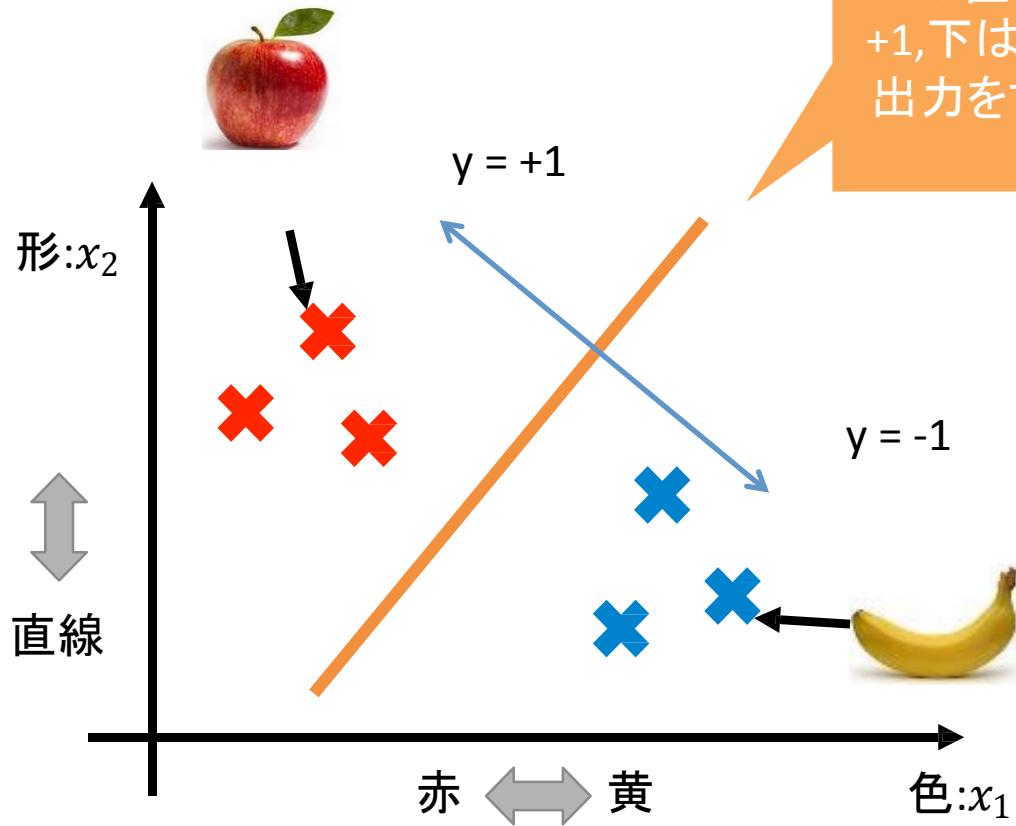
# まとめ

- 画像認識における特徴抽出指針
- Deep Learningが流行るようになった経緯
- ドラゴンボールとDeep Learning四天王

# パーセプトロン

- 線形識別器の基本的な考え方
- パーセプトロン
- 多層パーセプトロン

# パターン分類



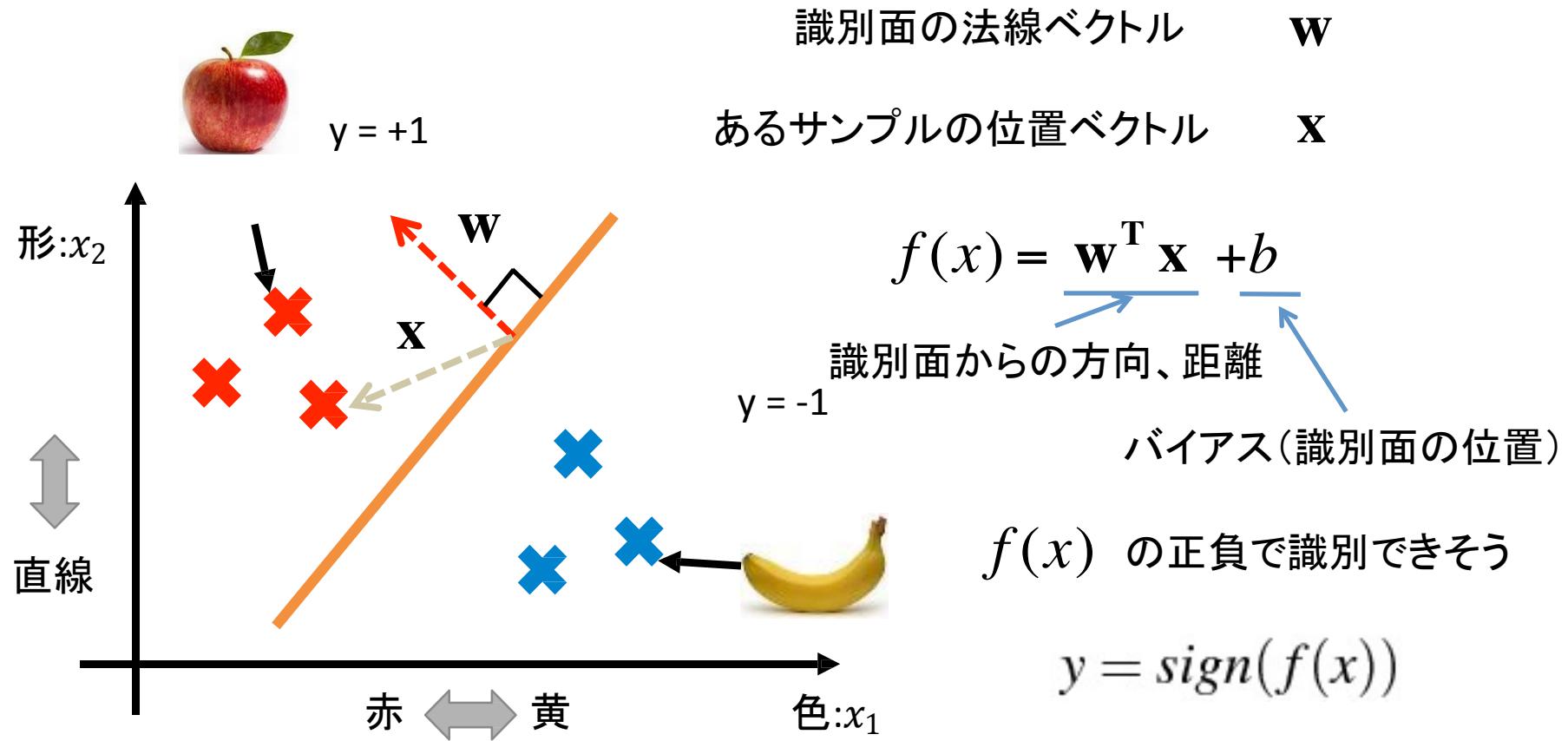
直線の関数を $f(x)$ とすると、

$$y = \begin{cases} +1 & (f(x) > 0) \\ -1 & (\text{otherwise}) \end{cases}$$

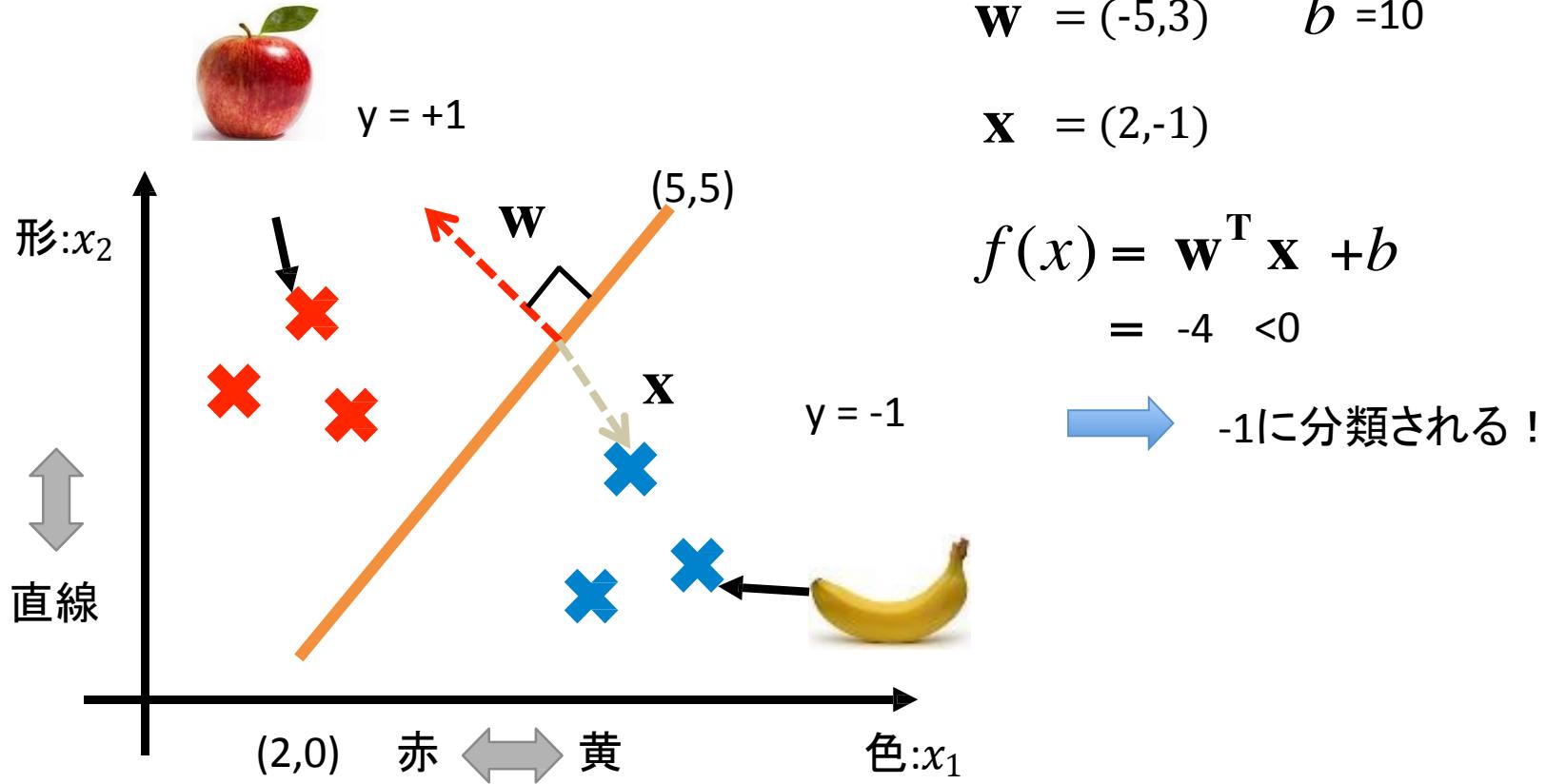
$$y = \text{sign}(f(x))$$

# 内積による識別

図の便宜上、原点の位置がずれています

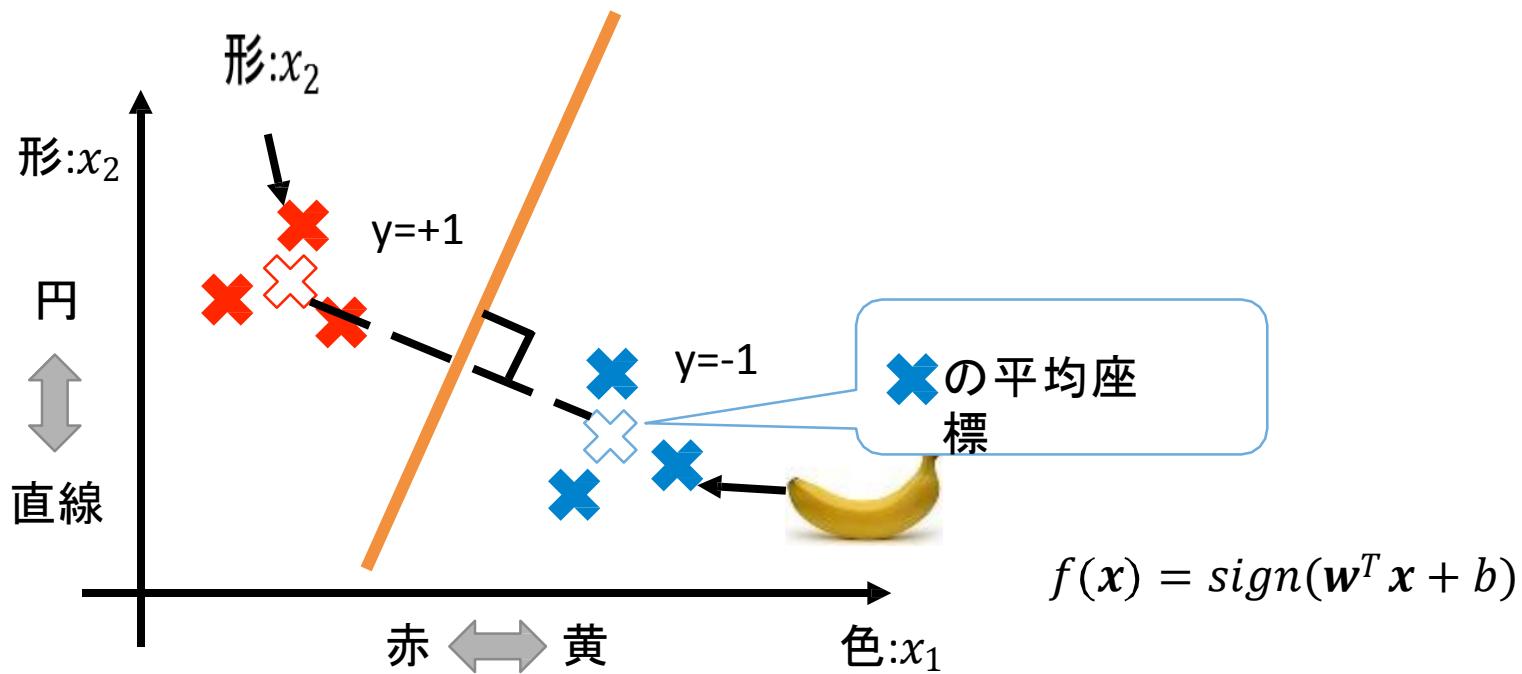


# 例



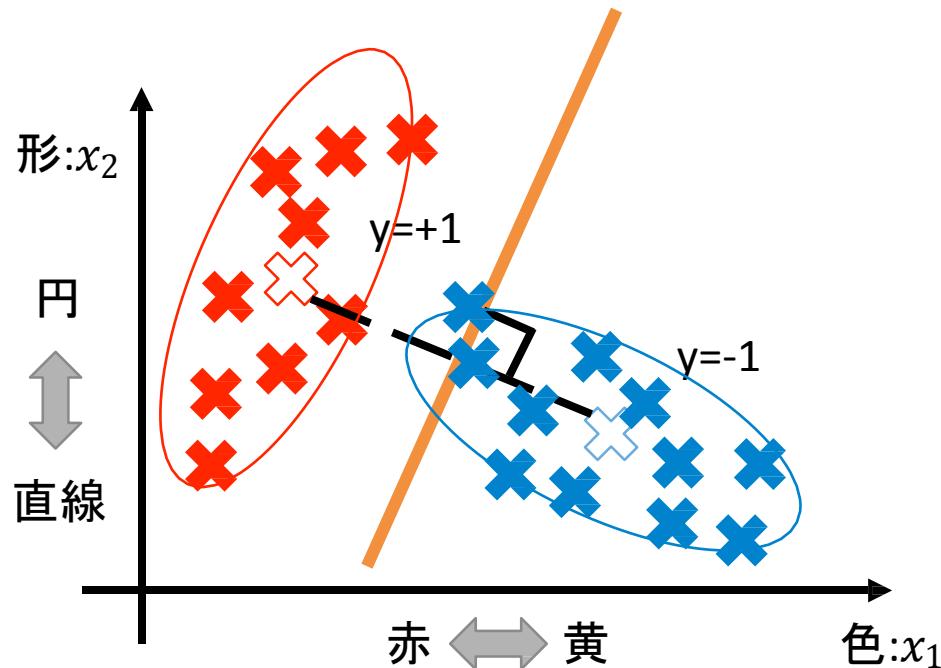
# 線形識別器の学習

- ・どうやって線形識別器のパラメータ( $w, b$ )を求める？
- ・クラスごとのパターンの平均値を結んだ線分の垂直二等分線？



# 線形識別器の学習

- ・パターンの分布の仕方がクラスごとに違うかも...
- ・なんとなくで設計するのはダメ

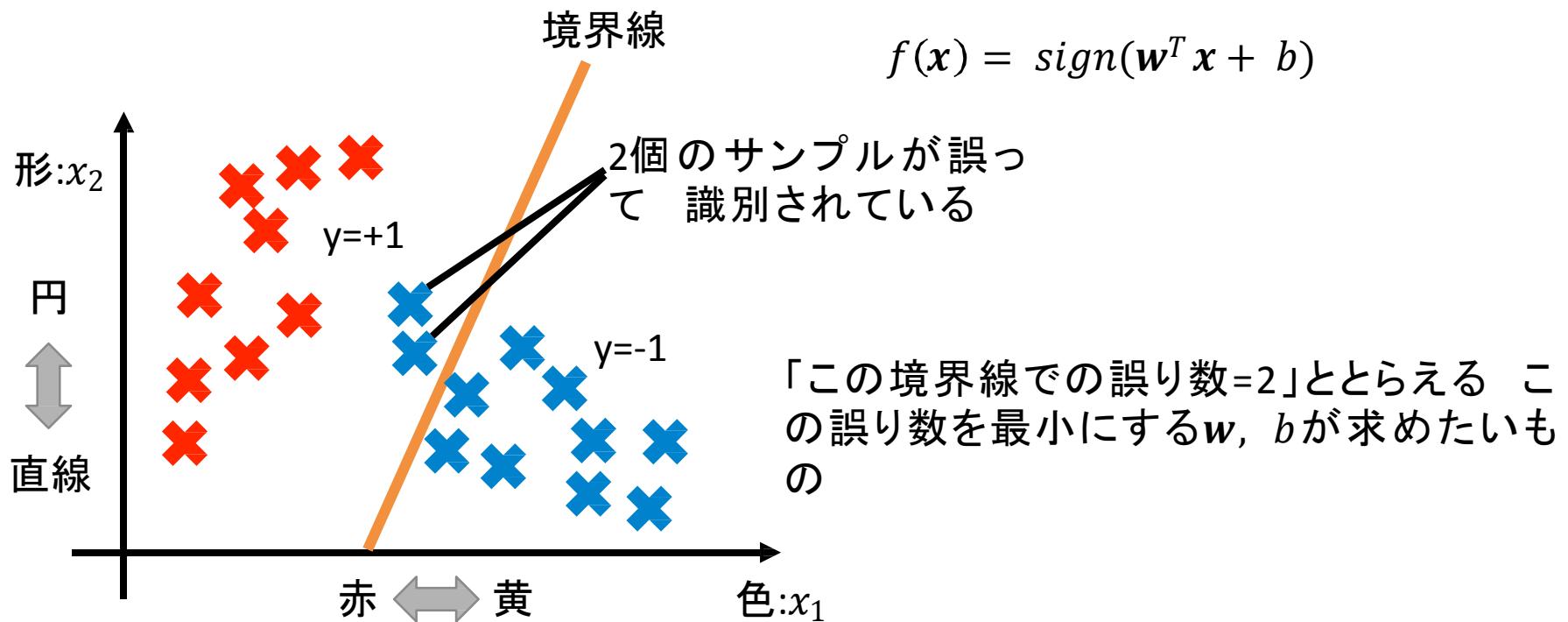


識別器の「良さ」の基準を定めて、最適化しなければならない

$$f(x) = \text{sign}(w^T x + b)$$

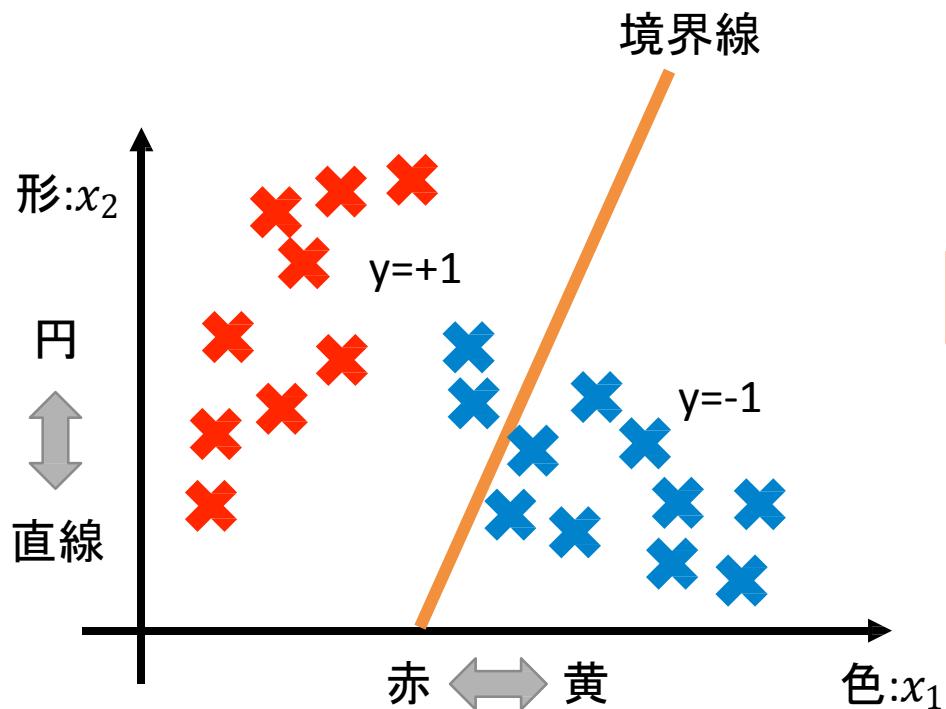
# 識別器の評価

- ・識別器の「良さ」を数値で表し、これを最大にするようなパラメータを求めるという方針をとる



# 損失関数

- 識別器の「良さ」を数値で表し、これを最大にするようなパラメータを求めるという方針をとる



# 損失関数とパラメータ更新

$$l(x, y, w) = \max(-yw^T x, 0)$$



損失関数をパラメータで微分

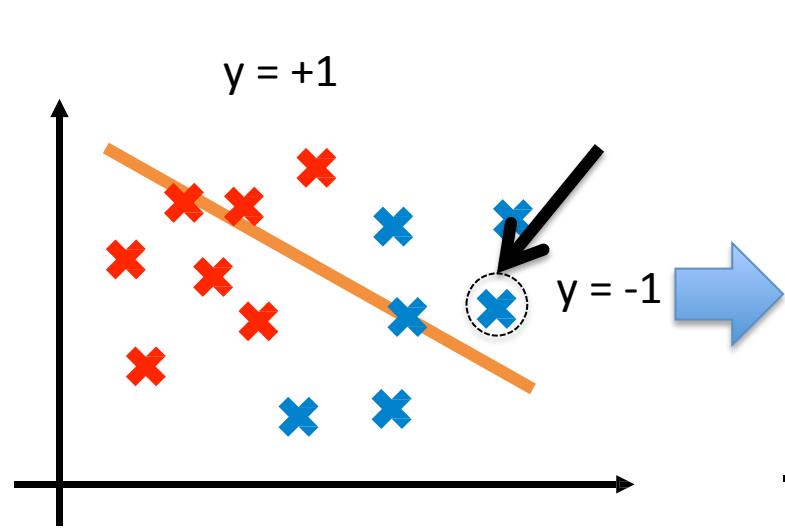
$$\nabla l(x, y, w) = \begin{cases} -yx & (if -yw^T x \geq 0) \\ 0 & (otherwise) \end{cases}$$



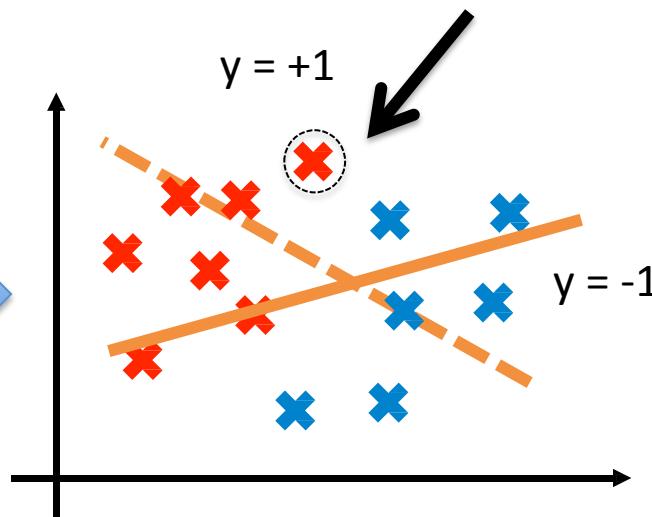
更新！

$$\nabla w^{(t+1)} = \begin{cases} w^{(t)} + y^{(t)}x^{(t)} & (if -y^{(t)}w^{(t)T} x^{(t)} \geq 0) \\ w^{(t)} & (otherwise) \end{cases}$$

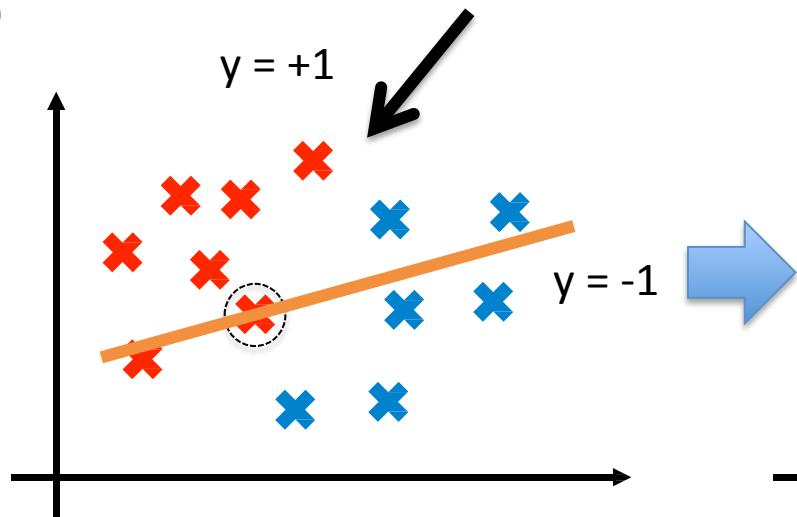
① 間違っているのでパラメータ更新



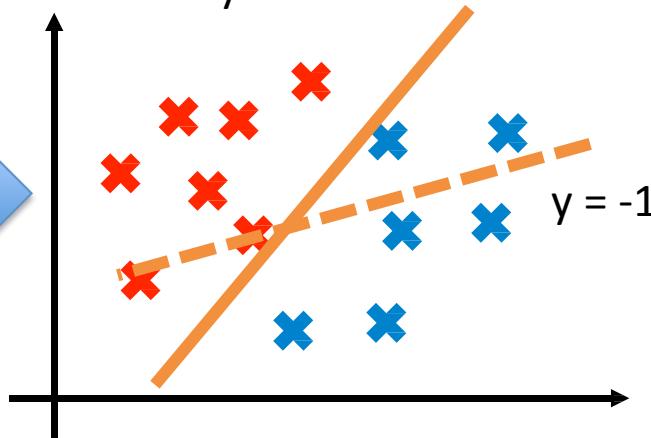
② あってるので更新しない



③ 間違っているので更新

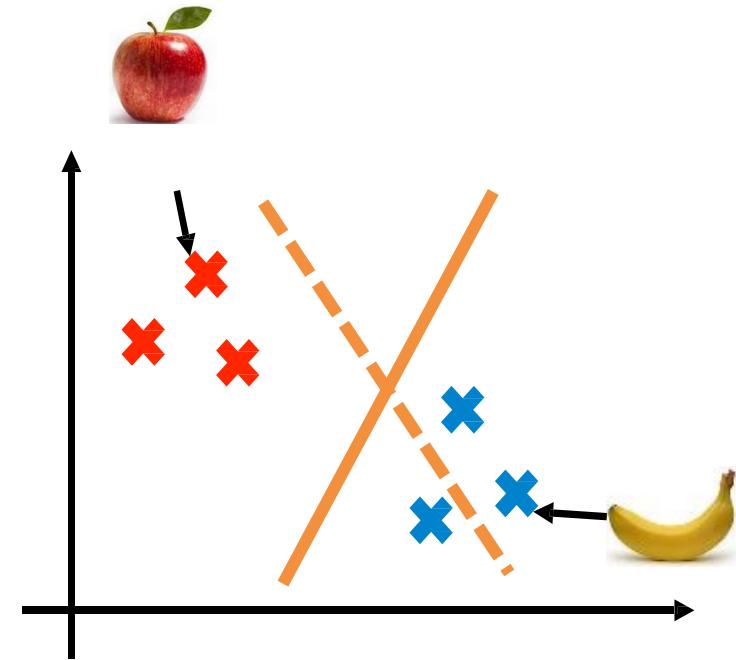
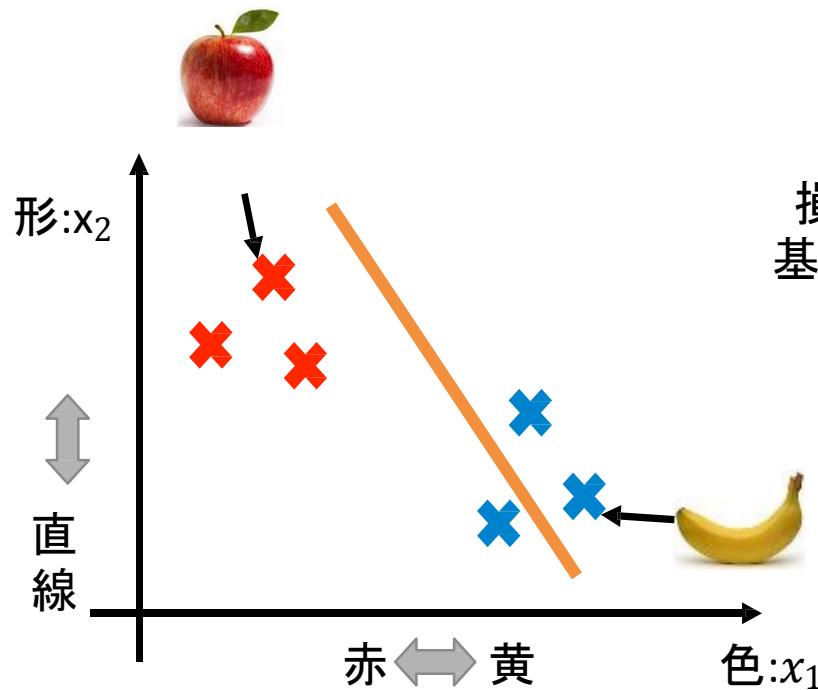


どれを選んでもあってい



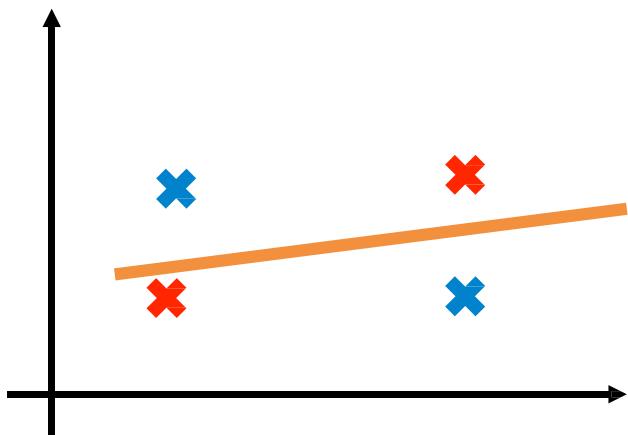
# パターン識別の基本的な考え方

- クラスを分離できる決定面を学習したい
- 決定面の良さの基準となる損失関数を選ぶ
- 微分、更新

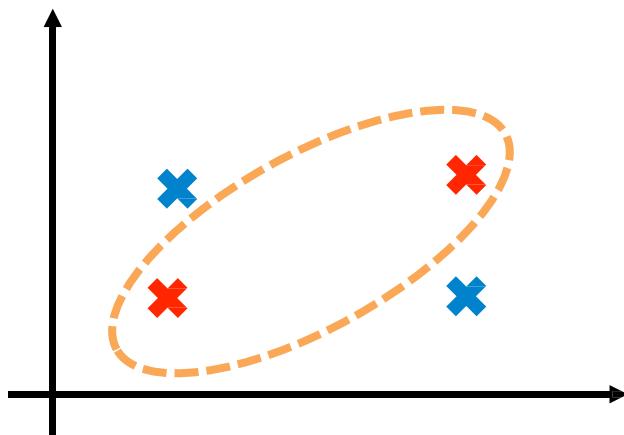


# 線形分離できない例

どう考えても無理！



本当はこんな感じになってほしい

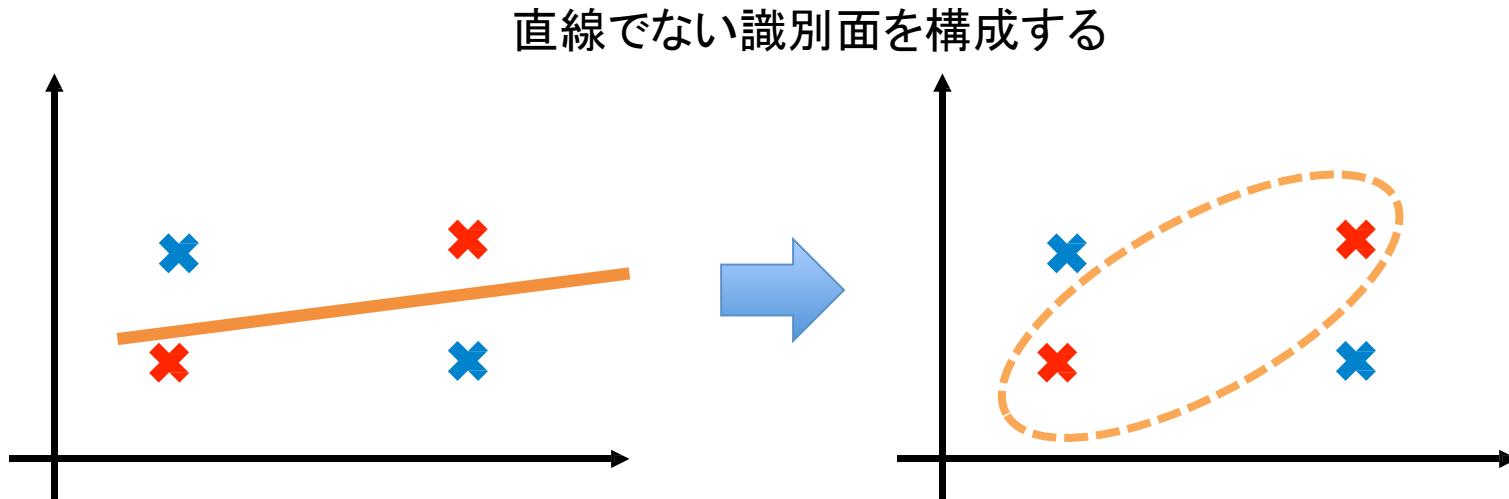


識別関数をもっと複雑にする必要あり。

多層パーセプトロンの出番だ！

# 多層パーセプトロン

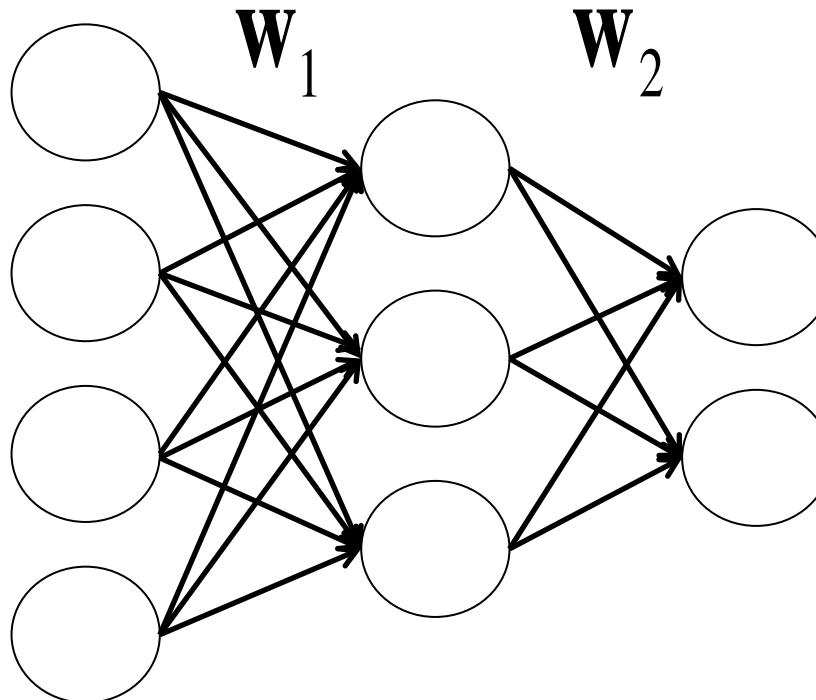
- ・入力パターンを多重に変換していく
- ・非線形性を加えることで、線形非分離な例も分離できるようになる



# 多層パーセプトロンの構造

入力パターン  
入力層  
中間層  
出力層  
得たい結果  
(確率っぽいもの)

$x$

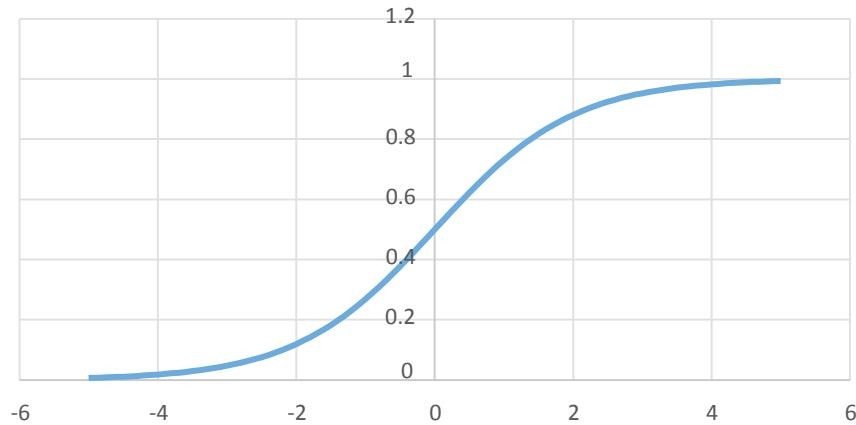


$$y = f_2(W_2 f_1(W_1 x))$$

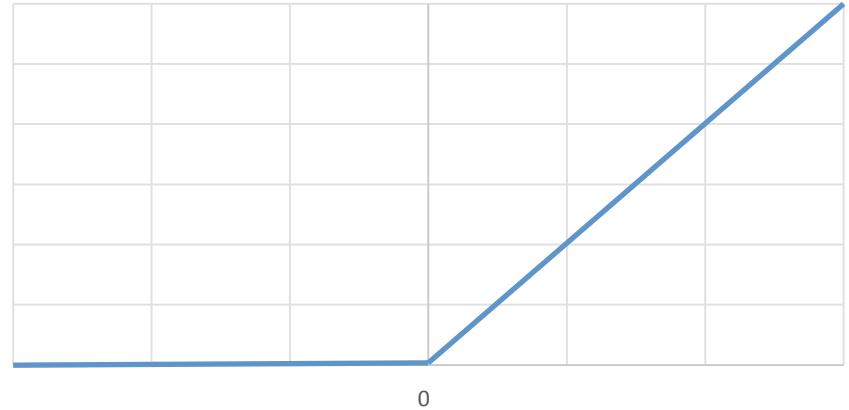
$f_2, f_1$  は、各層におけるアクティベーション関数

# アクティベーション関数

Sigmoid



ReLU

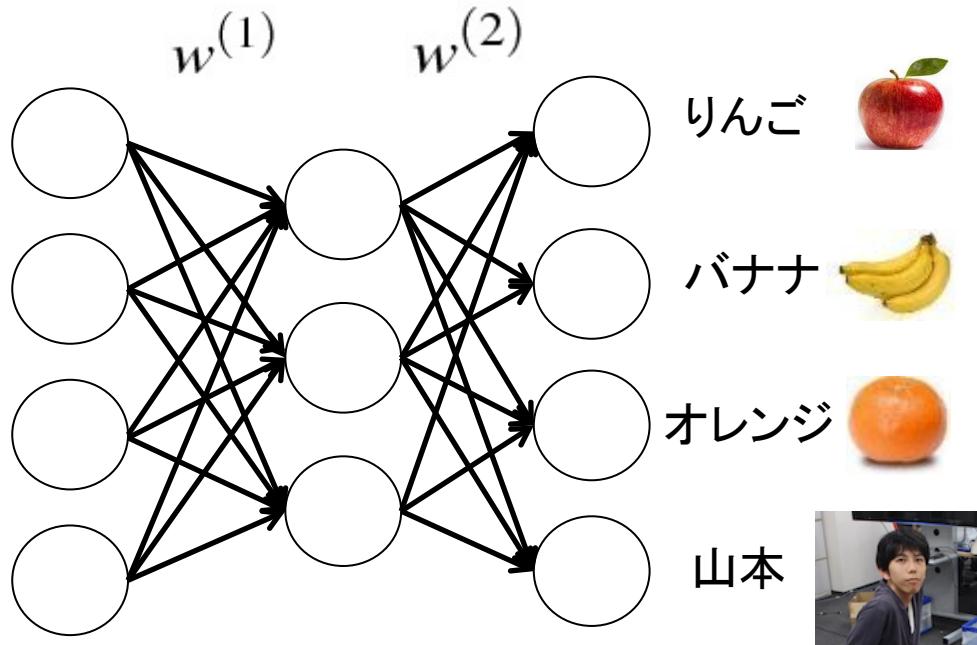


Softmax(出力時)

$$\text{softmax}(x_k) = \frac{\exp(x_k)}{\sum_{j=0}^D \exp(x_j)}$$

ネットワークに非線形性をもたせるためにアクティベーション関数を適応する。  
線形な関数をいくら重ねても、結局線形なのであまり嬉しくない

# ネットワーク例



$h \cdots \text{シグモイド}$

$$y_k(\mathbf{x}, \mathbf{w}) = \left( \sum_{j=0}^M \mathbf{w}_{kj}^{(2)} h \left( \sum_{i=0}^D \mathbf{w}_{ji}^{(1)} \mathbf{x}_i \right) \right)$$

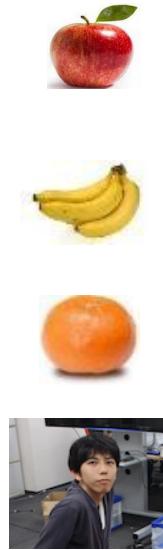
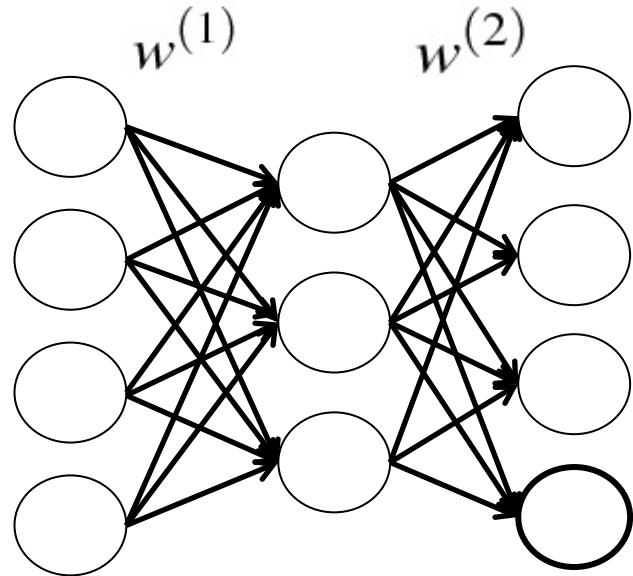
$$\text{softmax}(y_k(x, w)) = \frac{\exp(y_k(\mathbf{x}, \mathbf{w}))}{\sum_{j=0}^C \exp(y_j(\mathbf{x}, \mathbf{w}))}$$

各ユニットで出力される値が、  
入力パターンに対する各クラスの確率に  
相当するようにしたい

ソフトマックス関数

$$\text{softmax}(x_k) = \frac{\exp(x_k)}{\sum_{j=0}^D \exp(x_j)}$$

# 識別例



$$y_k(\mathbf{x}, \mathbf{w}) \quad softmax(y_k)$$

-4
-10
-7
-100

0.95
0.005
0.045
ほぼゼロ



山本はりんごからほど遠い！

$$y_k(\mathbf{x}, \mathbf{w}) = \left( \sum_{j=0}^M \mathbf{w}_{kj}^{(2)} h \left( \sum_{i=0}^D \mathbf{w}_{ji}^{(1)} \mathbf{x}_i \right) \right)$$

$$softmax(y_k(x, w)) = \frac{\exp(y_k(x, w))}{\sum_{j=0}^C \exp(y_j(x, w))}$$

$$\sum_{k=0}^C softmax(y_k(x, w)) = 1$$

# パラメータの決定

- 方針は線形モデルと同じ
  - 1, 目的関数(損失関数)を定義する
  - 2, パラメータで目的関数を微分して更新する

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \mathbf{E}_{\mathbf{n}}(\mathbf{w}^{(\tau)})$$

# 損失関数の勾配計算

- 多層になっているので、パーセプトロンみたいにわかりやすくは解けない→Chain Ruleを使って解く!

# 単純な例

出力  
 $y_k = \sum_i w_{ki} x_i$       入力パターンnに対する損失関数  
 $E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$

勾配計算

$$\begin{aligned}\frac{\partial E_n}{\partial w_{ji}} &= \frac{\partial E_n}{\partial y_{nj}} \frac{\partial y_{nj}}{\partial w_{ji}} \\ &= (y_{nj} - t_{nj}) x_{ni}\end{aligned}$$

  
出力側の誤差信号      入力

# 単純な例

出力  
 $y_k = \sum_i w_{ki} x_i$

入力パターンnに対する損失関数  
 $E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$

勾配計算

$$\begin{aligned}\frac{\partial E_n}{\partial w_{ji}} &= \frac{\partial E_n}{\partial y_{nj}} \frac{\partial y_{nj}}{\partial w_{ji}} \\ &= (y_{nj} - t_{nj}) x_{ni}\end{aligned}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \boxed{\nabla \mathbf{E}_n(\mathbf{w}^{(\tau)})}$$

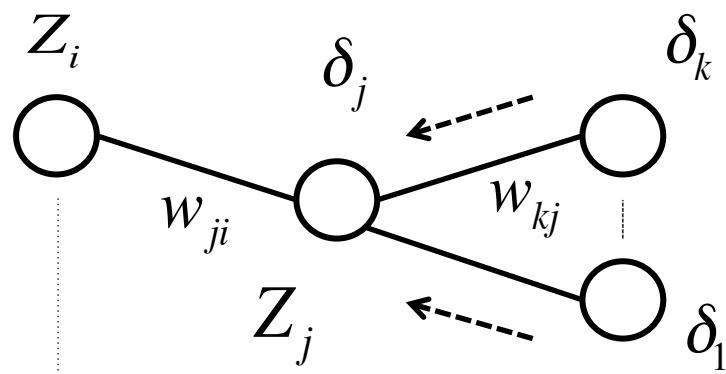
$$a_j = \sum_i w_{ji} z_i \quad (\text{a})$$

非線形関数  $h$ を考える

$$z_j = h(a_j) \quad (\text{b})$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \quad (\text{c})$$

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} \quad (\text{d})$$



$$\frac{\partial a_j}{\partial w_{ji}} = z_i \quad (\because (\text{a}))$$

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \quad (\because (\text{a}), (\text{c}), (\text{d}))$$

$$\delta_k = y_k - t_k$$

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \quad (\text{Chain rule})$$

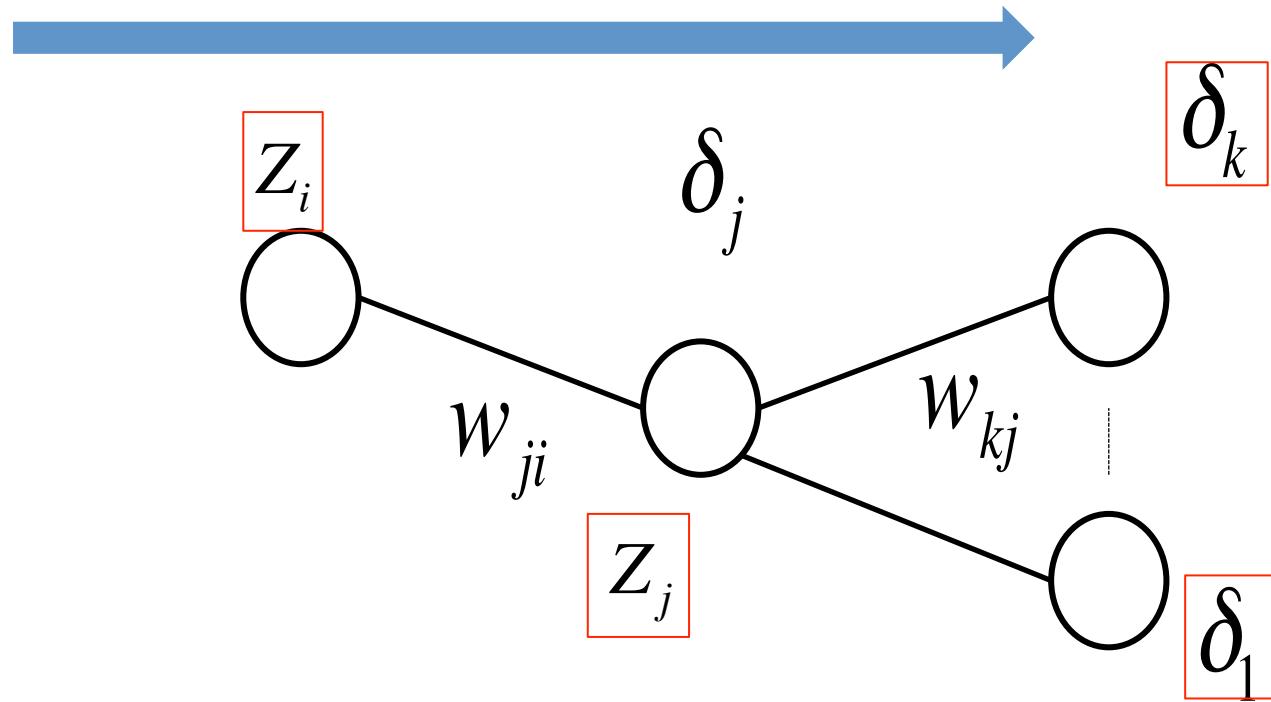
$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

自身の値

上位層に於ける誤差と  
リンクする重みの積

# 誤差逆伝播の全体

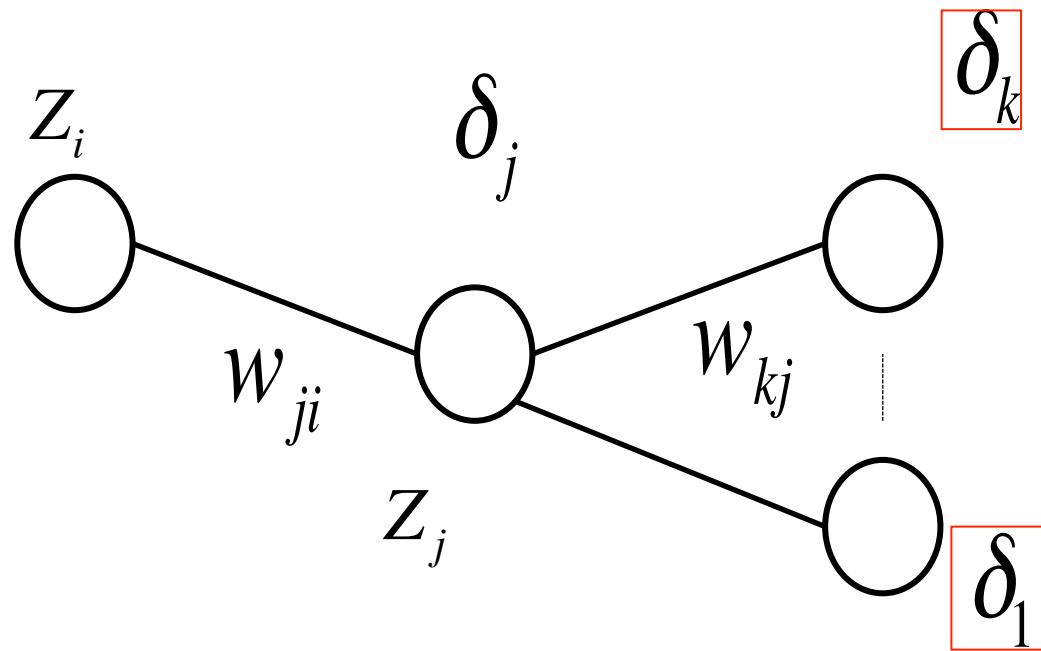
- 入力からフォワードして出力、中間層の値を得る



# 誤差逆伝播の全体

- 誤差を評価

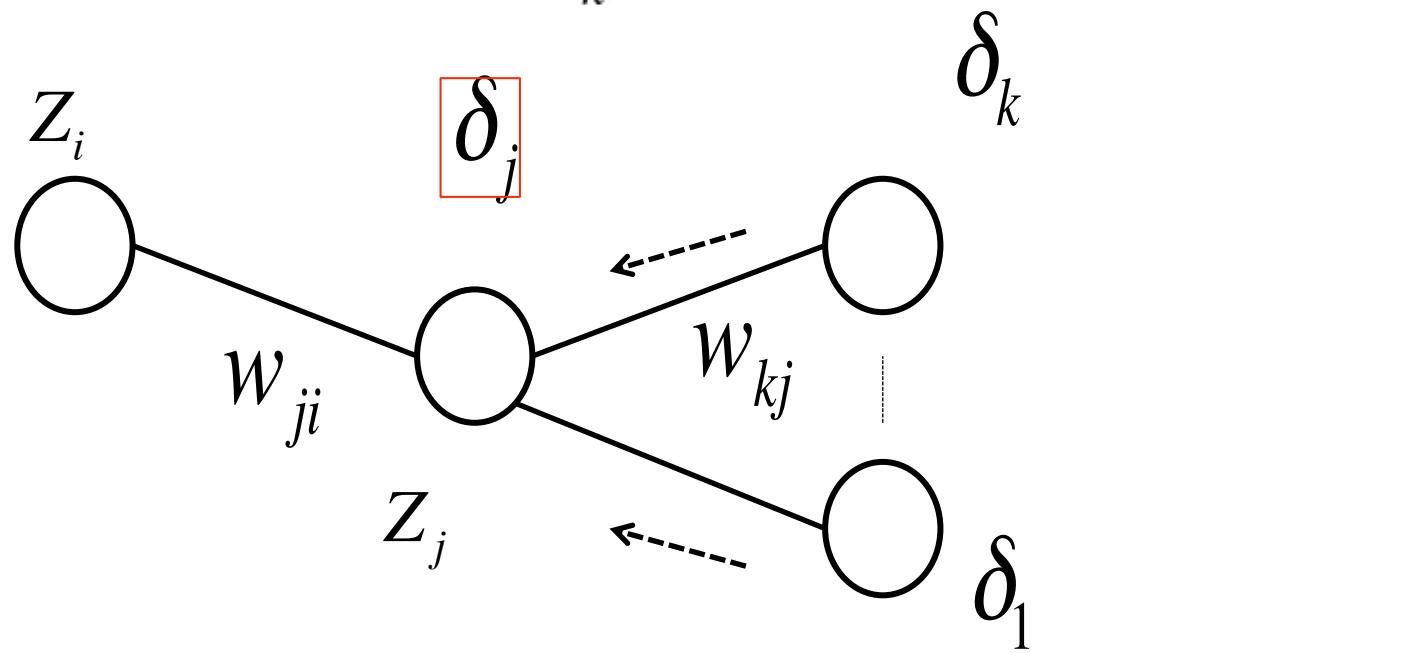
$$\delta_k = y_k - t_k$$



# 誤差逆伝播の全体

- 逆伝播で各ユニットの  $\delta_j$  を計算

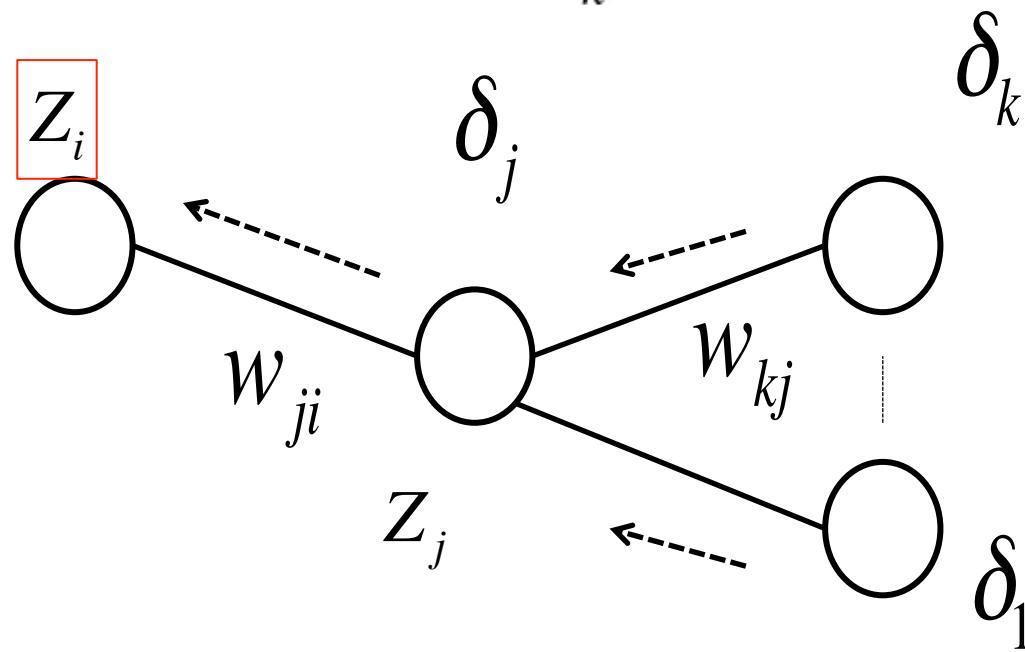
$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \quad \delta_k = y_k - t_k$$



# 誤差逆伝播の全体

- $\delta_j$ を使ってまた逆伝播していく

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \quad \delta_k = y_k - t_k$$



# まとめ

- パーセプトロン
  - 分離面の決定は目的関数定義→勾配計算
- 多層パーセプトロン
  - Chain ruleで、勾配計算しよう

# Thank you for listening!!

