
物体認識の高速化・省メモリ化の ための整数基底分解法

～線型モデルから深層学習まで～

デンソーアイティラボラトリ

安倍 満

2018/07/03

会社紹介

□ 社名：株式会社デンソーアイティラボラトリ

- ✓ 研究分野：画像認識・自然言語処理・信号処理・HMI
- ✓ 社員：32名
- ✓ 場所：東京都渋谷区

デンソーらしくない!

デンソーアイティラボラトリは、研究開発に特化したデンソーのグループ企業です。

デンソーが持つマーケットや技術を最大限に活用し、

最終的には自分達の研究開発成果を製品につなげていくことを最大の目的としています。

しかし、私達の研究開発テーマは、デンソーの製品ラインナップありきで定めたものではありません。

私たちが見据えているのは5年くらい先の世界。

私たちは、ネットワークであらゆるデバイスとモノがつながっていく中で、

自動車という枠に縛られずに



「こんなものがあったらユーザーにもっと喜んでもらえるんじゃないか」といった、

あえて「デンソーらしくない」自由な観点から、研究開発の種を探しています。



1. 論文をサーベイ
 2. PCでアルゴリズムを検討する
 3. 性能評価する
- ここまでは
うまくいくことが多い
4. マイコンに実装してみる → 遅過ぎてのらない!
 5. パラメータの妥協 → 性能を落として速度を稼ぐ
 6. できる限りのチューニングを施す
固定小数点化, SIMD命令の活用, ハード化の検討(FPGA), etc.
 7. なんとか完成

速度・メモリ消費量の効率化のためには？

- 実装の努力だけでは限界がある 
パラメータ調整, 性能の妥協, 固定小数点化、CPU依存命令の活用, etc.
- **整数基底分解法**の提案 (本日のトピック) 



整数基底分解法って、どういうもの？

一言でいうと、
「積和演算の高速化」
のためのテクニックです。





整数基底分解法って、何に使えるの？

2つの適用事例について紹介します。

- 事例 1 : 線型SVMによる歩行者検出
- 事例 2 : Deep Neural Network

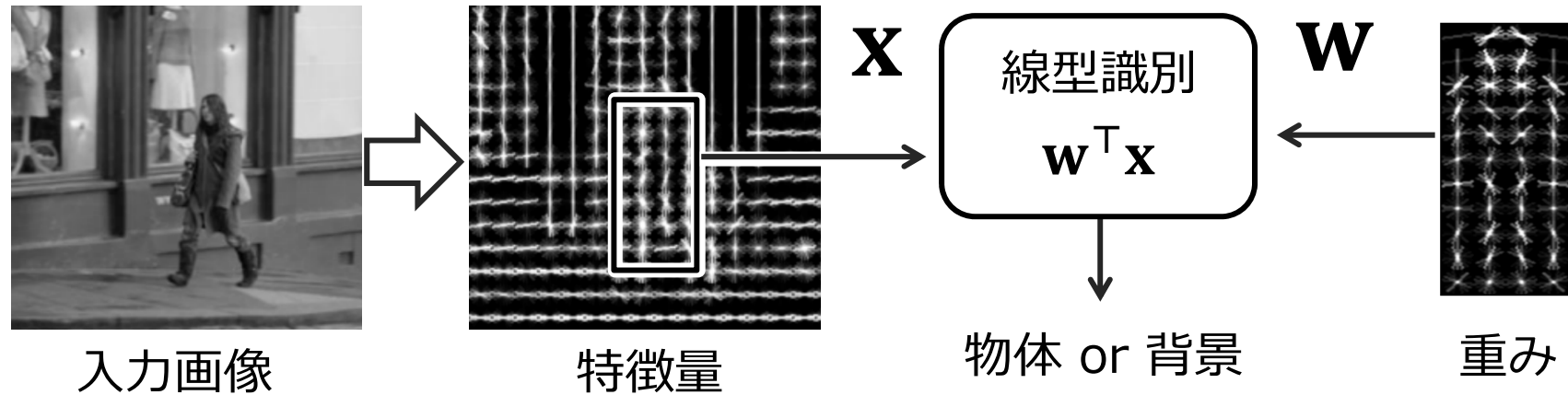


事例 1 : 線型SVMによる歩行者検出



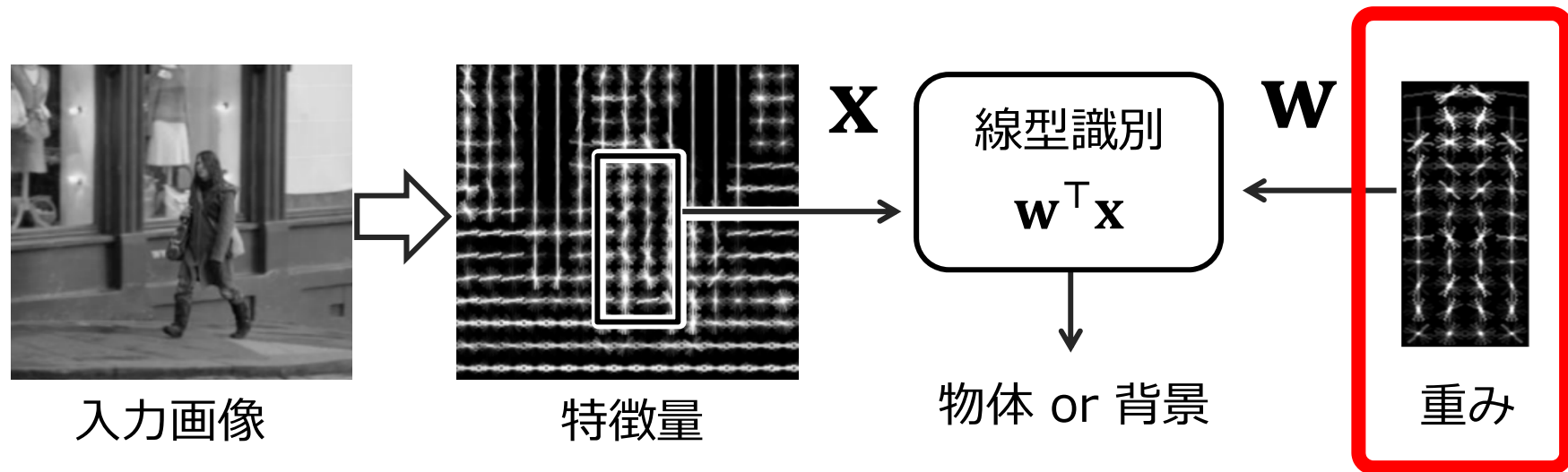
M.Ambai and S.Ikuro, SPADE: Scalar Product Accelerator by Integer Decomposition for Object Detection (ECCV2014)

事例 1 : 線型SVMによる歩行者検出



P. Felzenszwalb et.al., Object Detection with Discriminatively Trained Part Based Models, PAMI2010
上記論文から一部図を引用(入力画像と重み係数)

事例 1 : 線型SVMによる歩行者検出

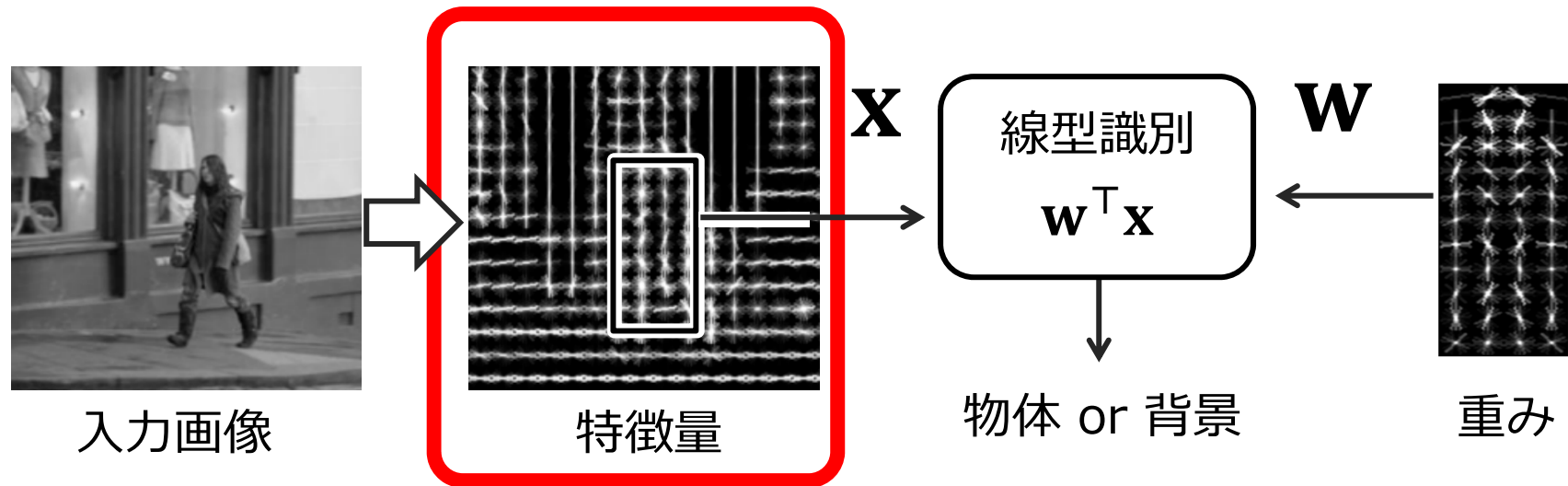


● 大量のデータから「重み w 」を学習

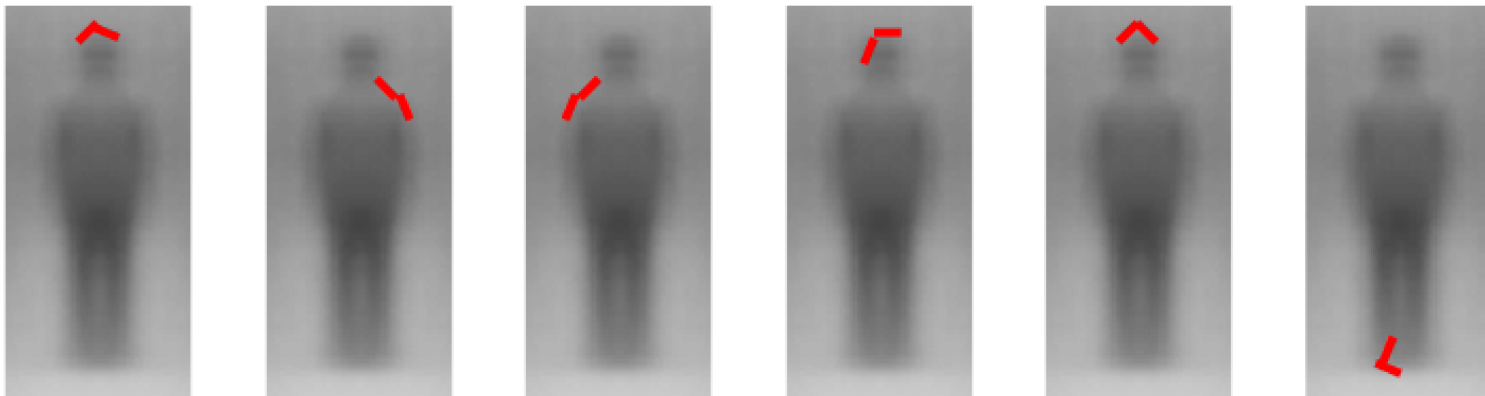


INRIA Person Dataset <http://pascal.inrialpes.fr/data/human/>

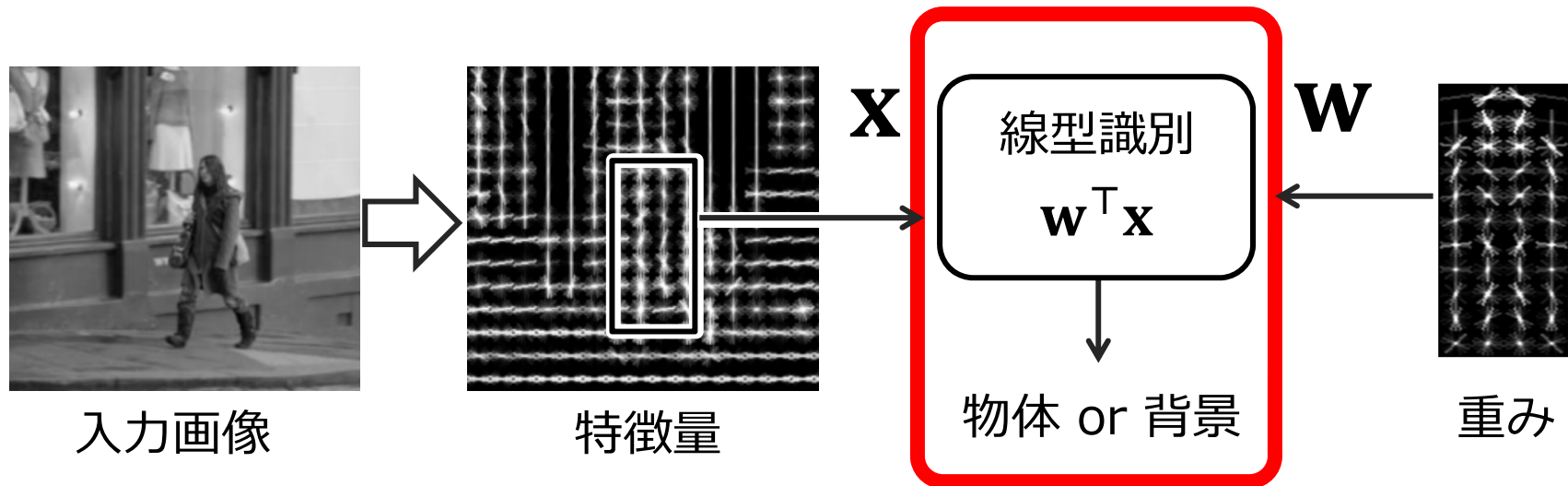
事例 1 : 線型SVMによる歩行者検出



- 画像から「特徴量 x 」を抽出 (HOG特徴量など)



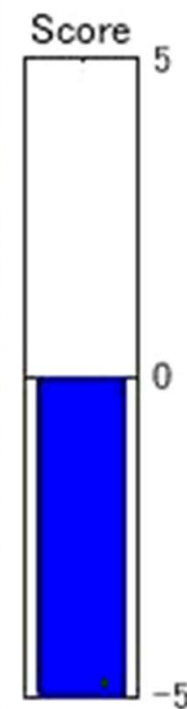
事例 1 : 線型SVMによる歩行者検出

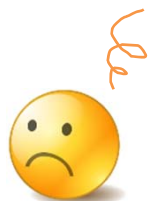


● 識別処理

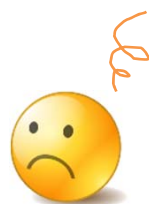
内積 $\mathbf{w}^T \mathbf{x}$ の値に応じて物体か否かを判定

特徴量 $\mathbf{x} =$	0.1	1.1	2.0	0.7	-2.2	0.1	...	-1.4	1.1	
	×	×	×	×	×	×		×	×	
重み $\mathbf{w} =$	1.2	3.1	-1.0	-3.1	-1.4	1.2	...	-2.1	0.6	
	↓	↓	↓	↓	↓	↓		↓	↓	
内積 $\mathbf{w}^T \mathbf{x} = \text{sum}(\$	0.1	3.4	-1.0	-2.2	3.1	0.1	...	2.9	0.7)





(内積の演算量 × 検出窓数) 分の積和演算が必要
数億～数十億回の浮動小数点演算が発生




動画 = 1秒間に30枚の静止画像
演算量はさらに増大


高速化へのアプローチ

組み込みでは、実数演算よりもビット演算の方が有利

浮動小数点演算

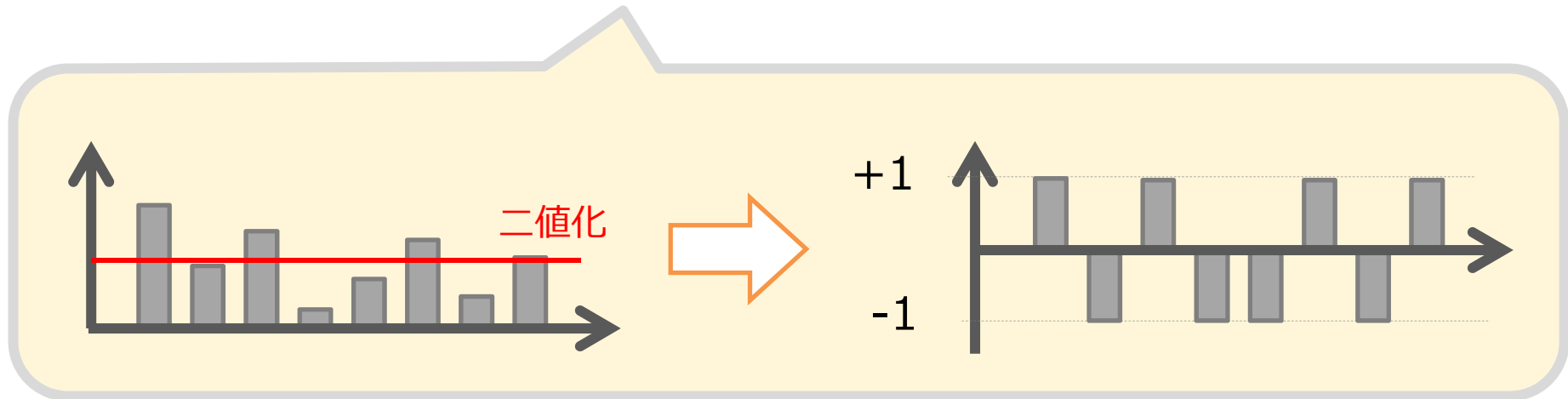
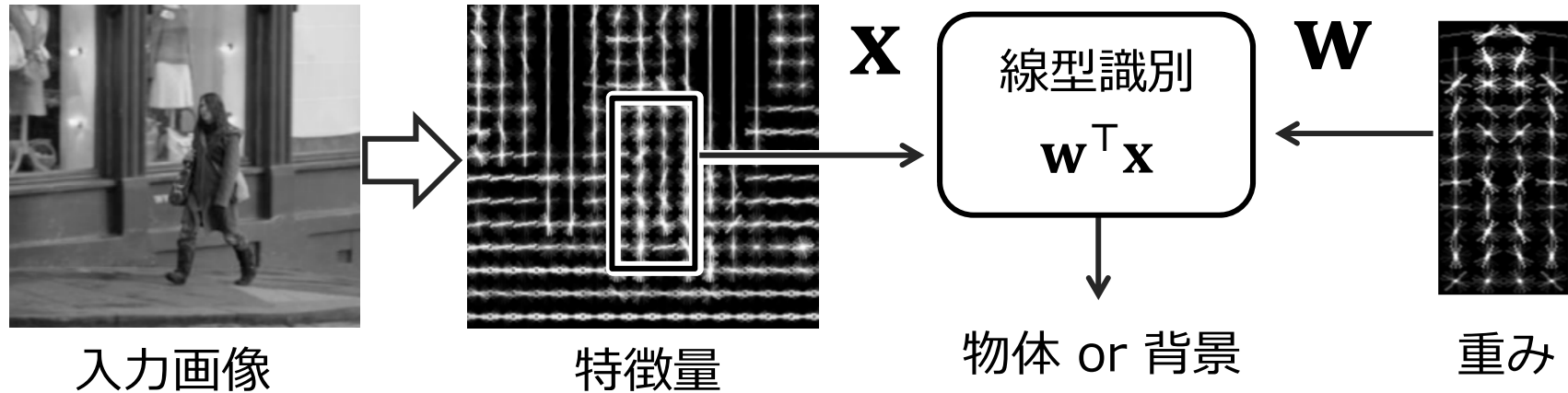
- 足算 +
 - 引算 -
 - 掛算 ×
 - 割算 ÷
- 遅い... 

ビット演算

- AND
 - OR
 - XOR
 - NOT
 - BitCount
 - BitShift
- 速い! 

高速化へのアプローチ

- 特徴量 x を二値化して高速化できないか？



P. Felzenszwalb et.al., Object Detection with Discriminatively Trained Part Based Models, PAMI2010
 上記論文から一部図を引用(入力画像と重み係数)

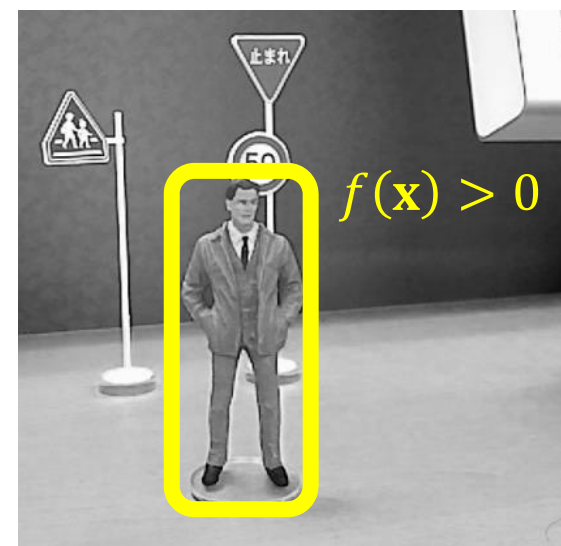
高速化へのアプローチ

- 特徴量が**二値**のとき，線型識別関数の速度を飛躍的に高める方法を提案

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

重み
(実数)

二値特徴量
+1 or -1



Contribution:



線型識別関数の速度を **36.9倍** 高速化

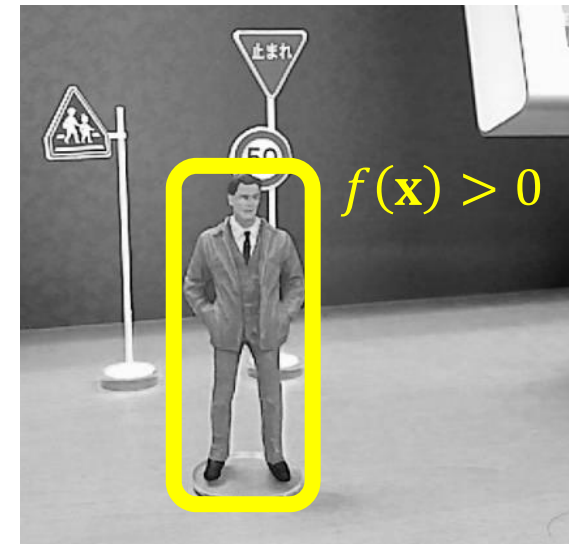
高速化へのアプローチ

- 特徴量が**二値**のとき，線型識別関数の速度を飛躍的に高める方法を提案

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

重み
(実数)

二値特徴量
+1 or -1

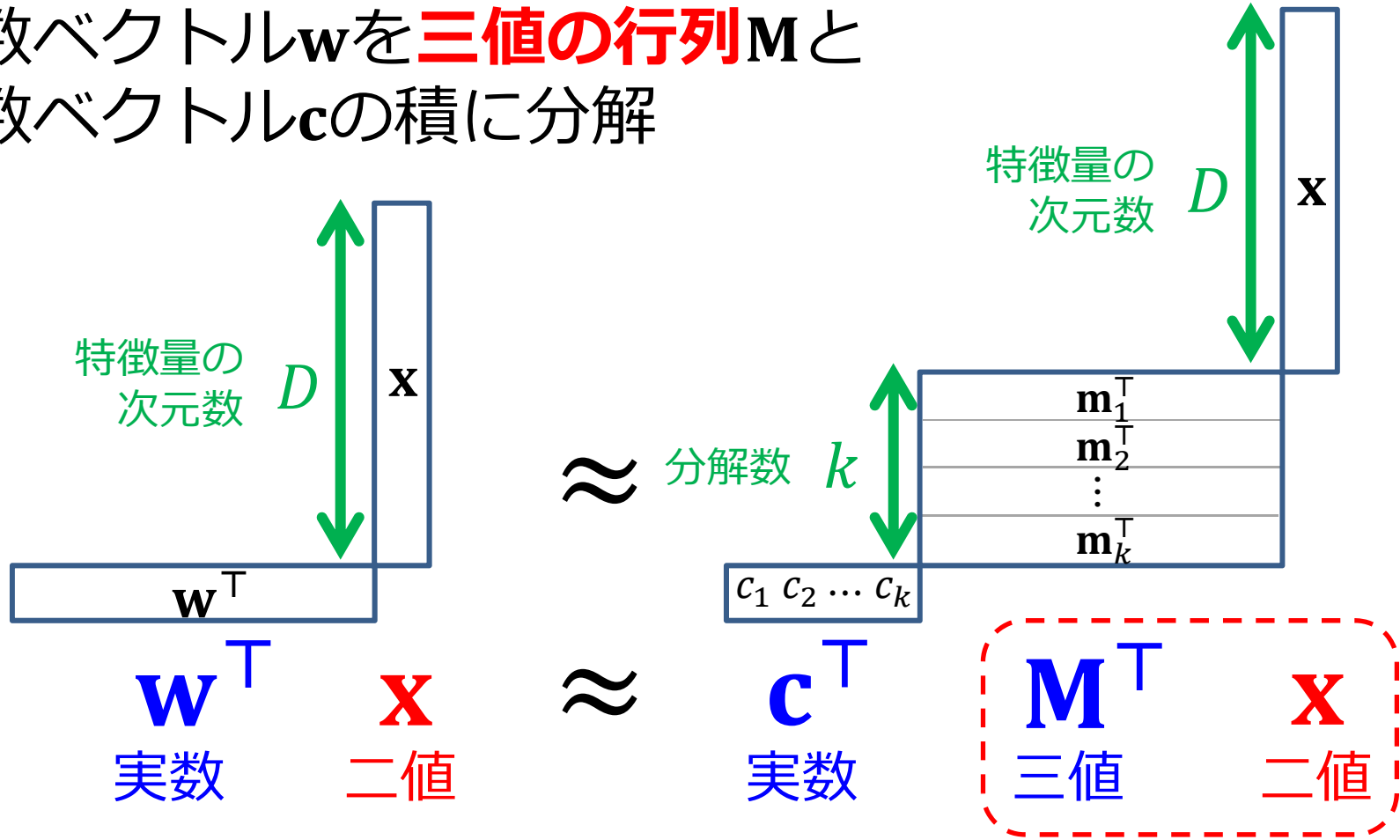


問題設定:

「**実数ベクトル**」と「**二値ベクトル**」の内積をどうにかして
ビット演算で処理したい！

整数基底分解法：重みベクトルの三値分解

実数ベクトル w を **三値の行列** M と
実数ベクトル c の積に分解



内積 $m_i^T x$ は、ビット演算で極めて高速に算出可能
(XOR, AND, BitCountなど)



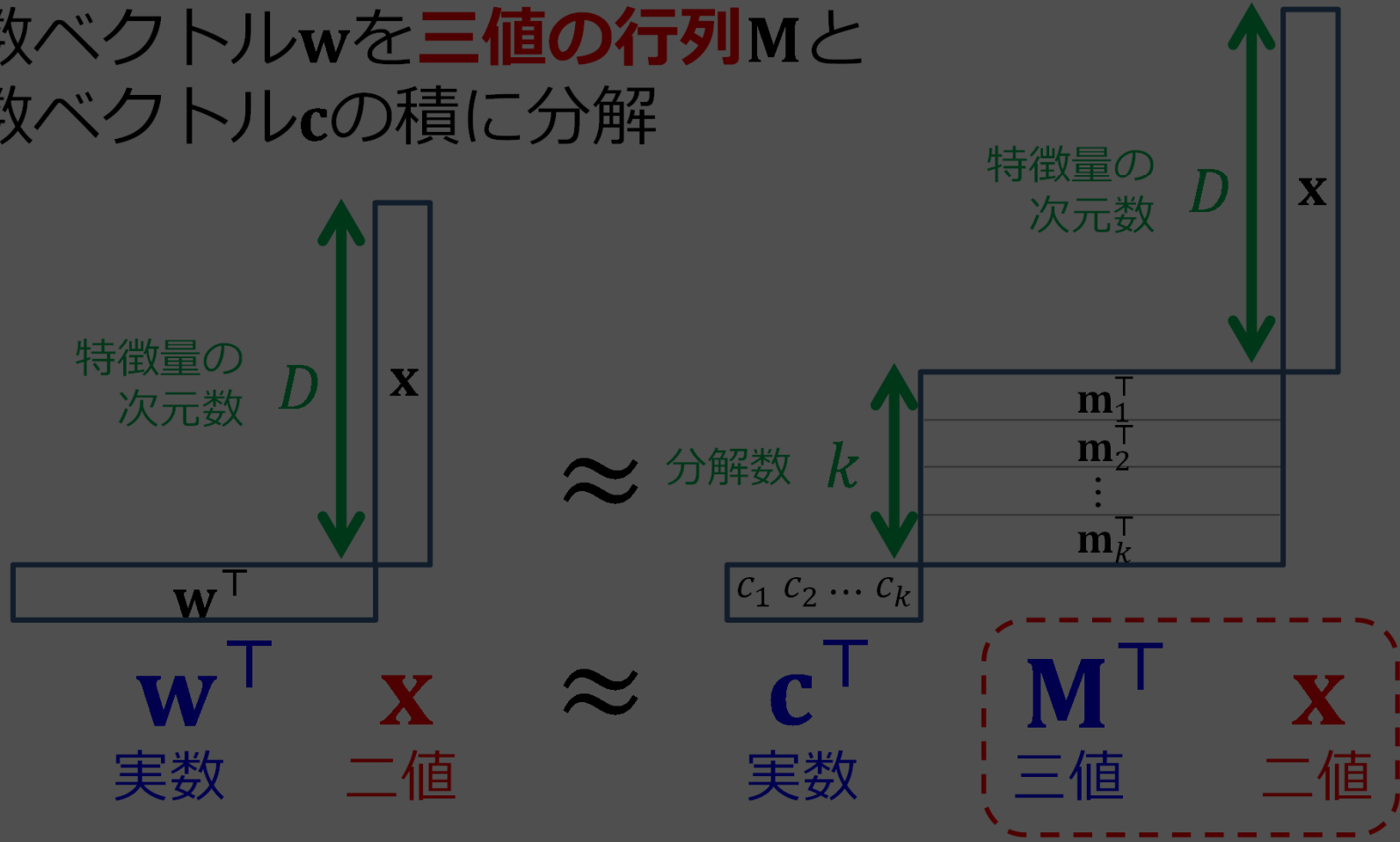
高速化の効果

● HOGとの比較

Method (32dim/blockの場合)	Acceleration	Miss rate @FPPI=0.1
HOG (従来法)	1×	0.370
二値化HOG + 整数基底分解法	36.9×	0.385

整数基底分解法：重みベクトルの三値分解

実数ベクトル w を **三値の行列** M と
 実数ベクトル c の積に分解



最適化問題 $\min_{M,c} \|w^T x - c^T M^T x\|_2^2$ を解く必要がある

MATLAB®による整数基底分解ソルバーの開発^{22/34}

● 最適化問題 $\min_{\mathbf{M}, \mathbf{c}} \|\mathbf{w}^T \mathbf{x} - \mathbf{c}^T \mathbf{M}^T \mathbf{x}\|_2^2$

汎用最適化ライブラリでは最小化が難しい

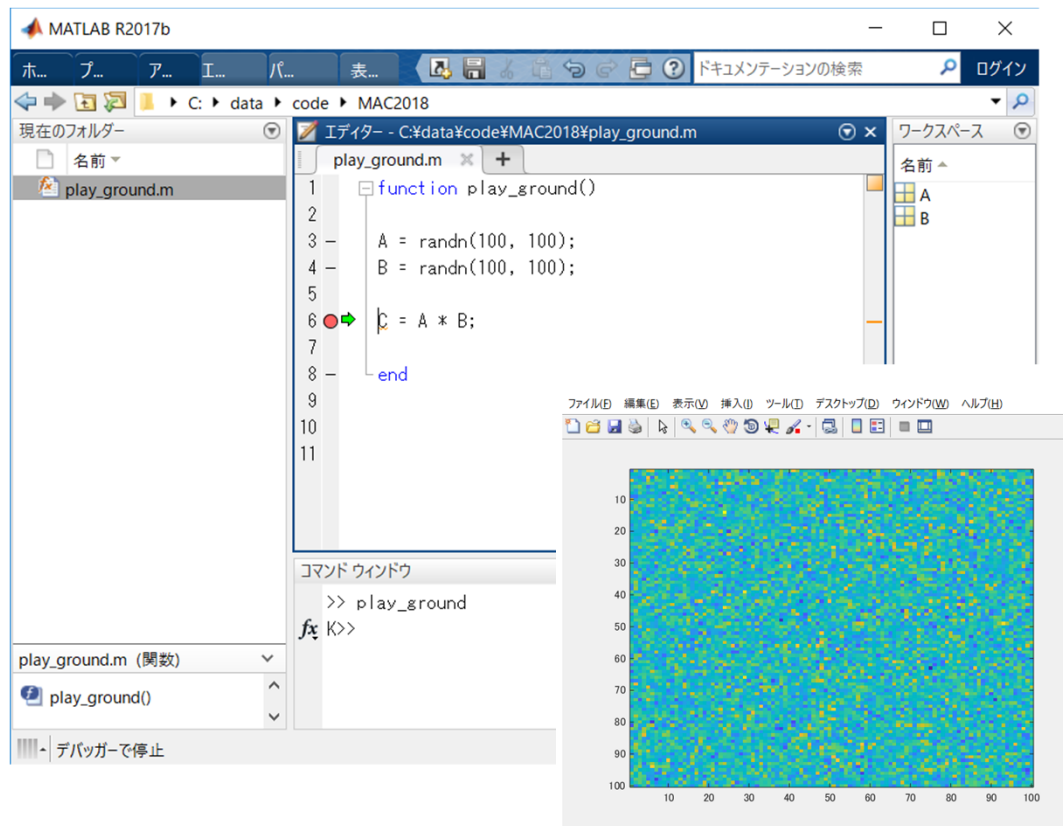
- 整数と実数を扱う混合整数計画問題
- NP困難
- Optimization Toolbox™でも取り扱いづらい

→ 専用ソルバーを独自に開発

数分～数時間で最小化完了

なぜMATLAB?

- **内部状態のインタラクティブな可視化に優れる**
- 短く書けるため、アルゴリズムの検討に集中できる



- Break point
- Save, Load
- Imagesc, plot, etc.

なぜMATLAB?

- 内部状態のインタラクティブな可視化に優れる
- **短く書けるため、アルゴリズムの検討に集中できる**

```
for cnt_C1 = 1:size(C, 1)
    for cnt_C2 = 1:size(C, 2)
        f = A(:, idx1 + cnt_C1, idx2 + cnt_C2);
        C(cnt_C1, cnt_C2) = f(:)' * B(:) + bias;
    end
end
```

例：スライディングウィンドウによる検出

機械学習の多くは行列と馴染みがよく、簡潔に書ける

なぜMATLAB?

- 内部状態のインタラクティブな可視化に優れる
- **短く書けるため、アルゴリズムの検討に集中できる**

$$y = W3 * \max(0, W2' * \max(0, W1' * x));$$

例：3層の全結合型ニューラルネットワーク
機械学習の多くは行列と馴染みがよく、簡潔に書ける



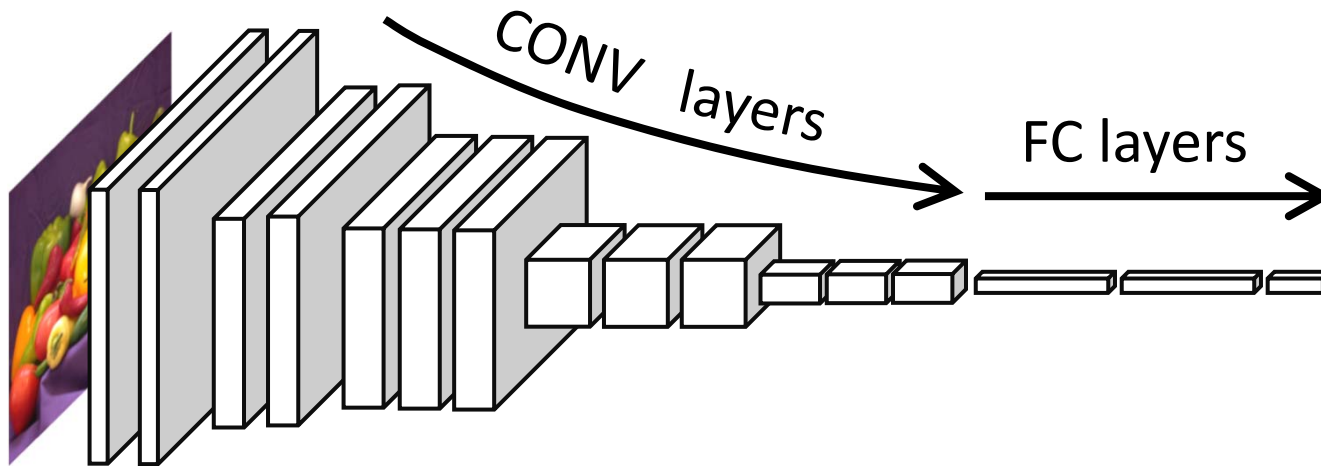
整数基底分解法って、何に使えるの？

2つの適用事例について紹介します。

- 事例 1 : 線型SVMによる歩行者検出
- 事例 2 : Deep Neural Network

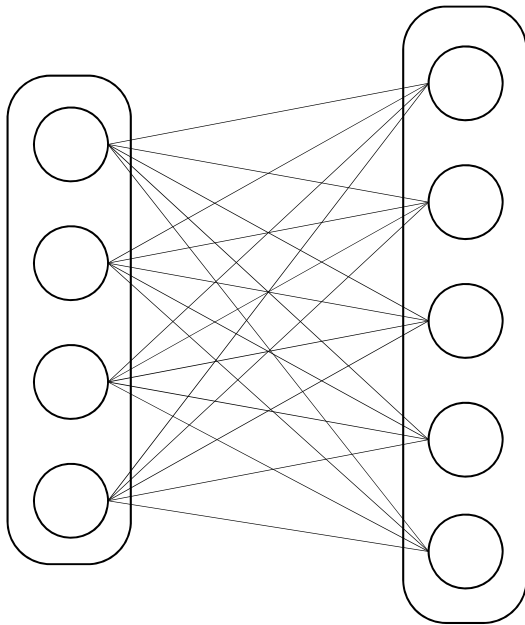


事例 2 : Deep Neural Network



- 課題
 - 全結合層のメモリ消費量が大きい
 - 畳み込み層の計算時間が遅い

DNNにおけるベクトル行列積

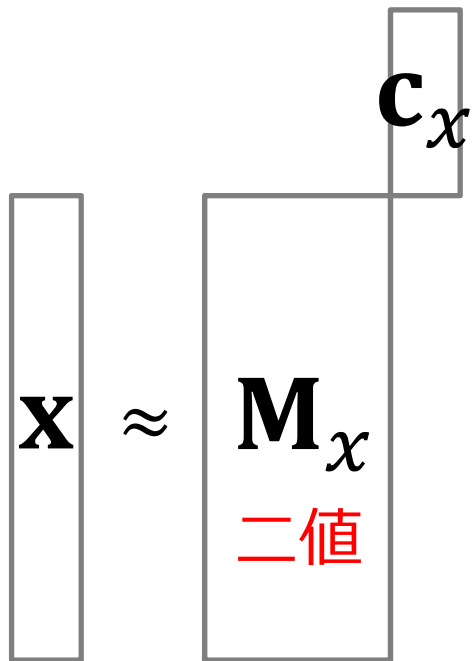


$$\mathbf{y} = f(\mathbf{W}^T \mathbf{x} + \mathbf{b})$$

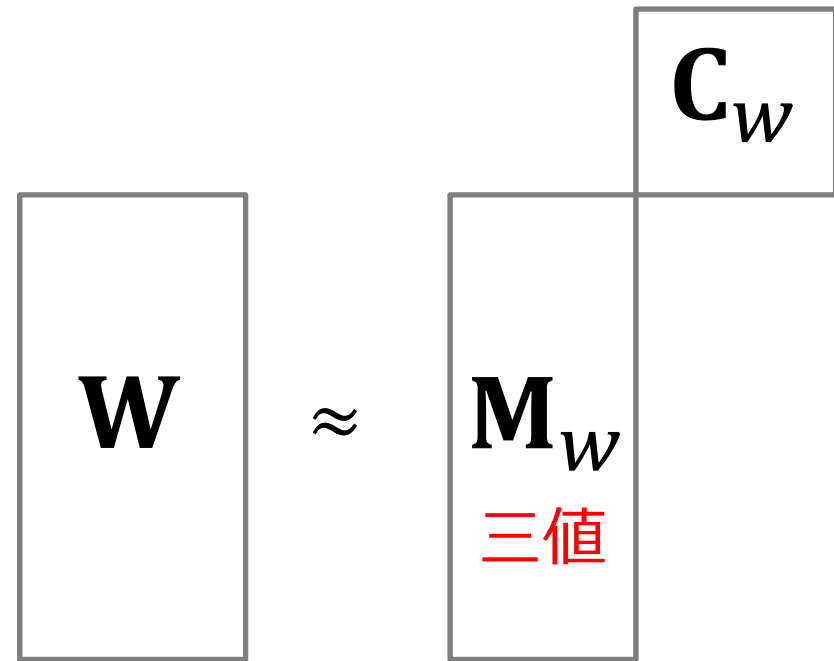
DNNの場合、 \mathbf{x} は容易に二値化できない
→ 重み行列 \mathbf{W} だけでなく、入力 \mathbf{x} も分解

Neural Networkの整数基底分解

～入力の分解～



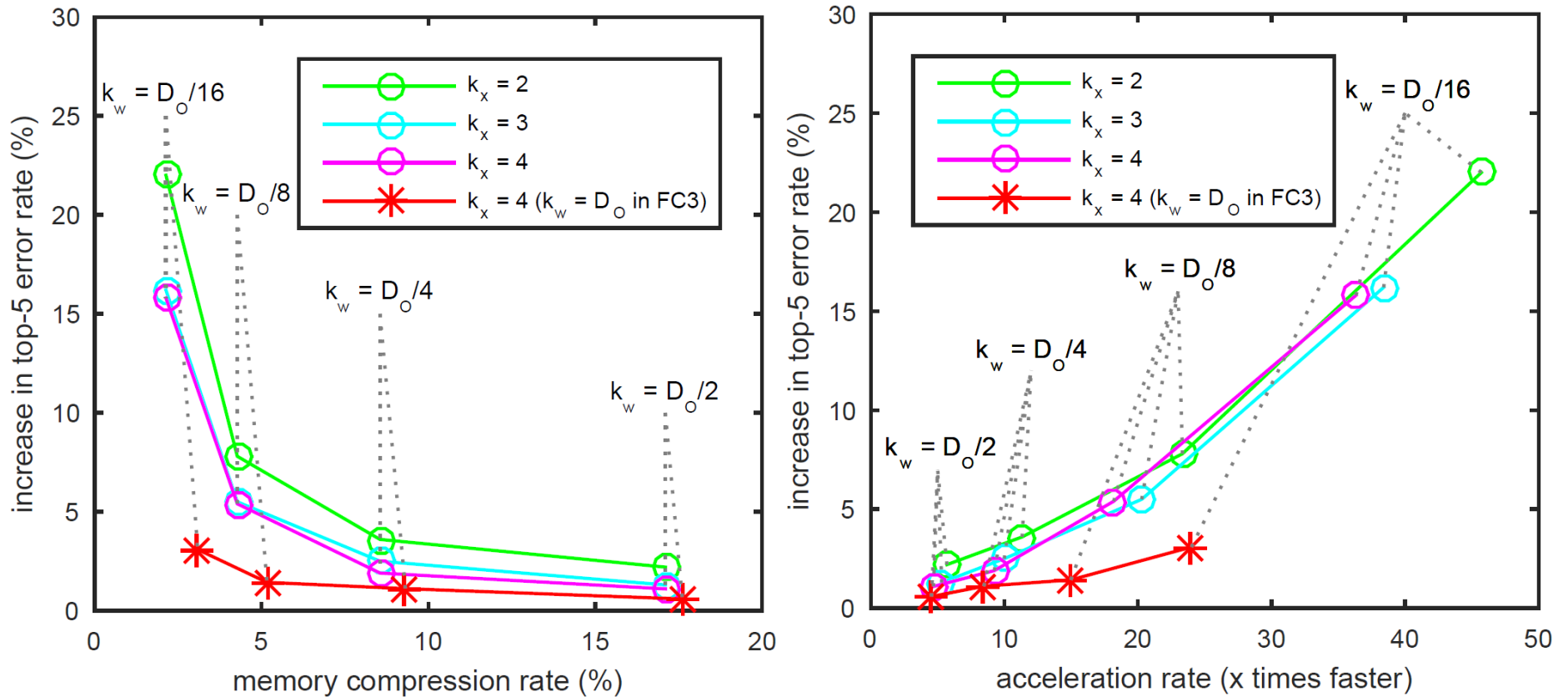
～重みの分解～



実験結果 (ImageNet Classification Task)

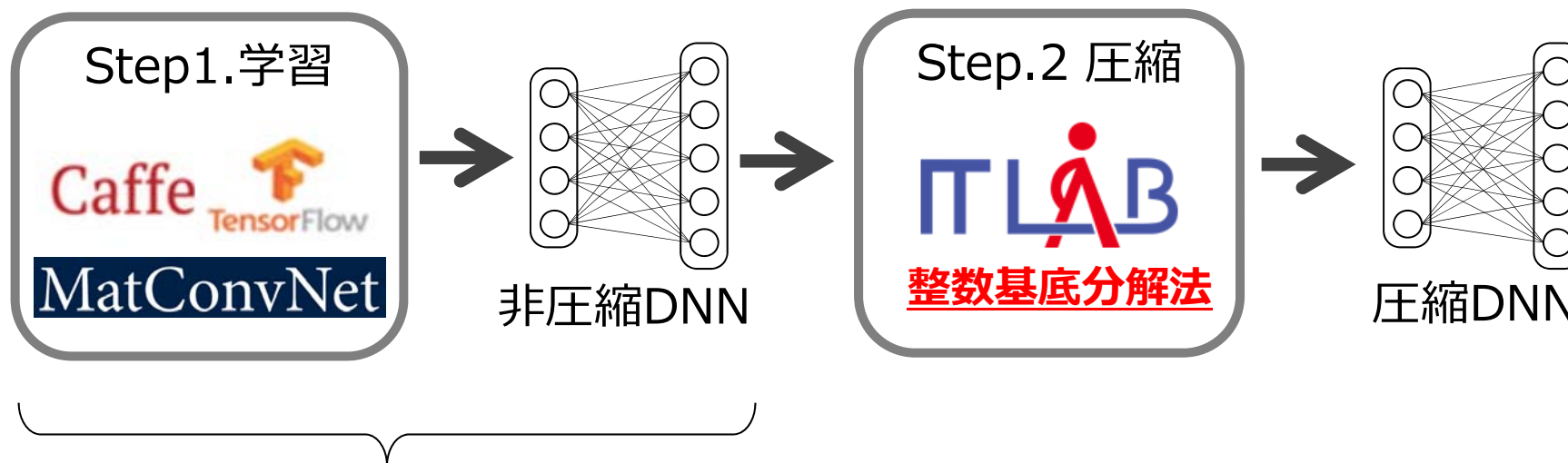
- ネットワーク構造 VGG-16 (Top-5 error rate = 13.4%)
- 全結合層(14,15,16層)を圧縮

1/20のメモリ圧縮率, 15倍の高速化 (1.43%のエラー率増加)



組み込み向けDNN生成のプロセス

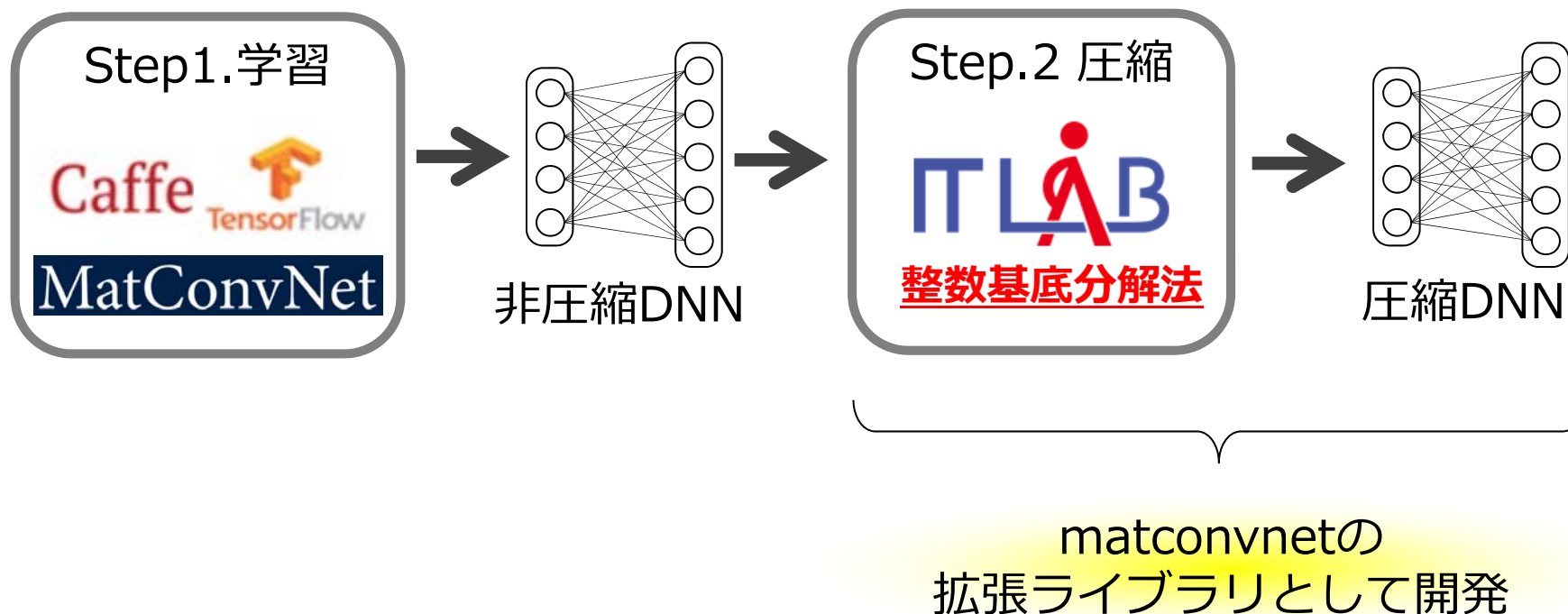
Point: 学習と圧縮の分離設計



既存の学習アルゴリズムや
学習ライブラリを自由に使ってよい

組み込み向けDNN生成のプロセス

Point: 学習と圧縮の分離設計



まとめ

- 高速化・省メモリ化のための整数基底分解法を提案した。
 - ✓ 事例1：線型SVMによる歩行者検出
 - ✓ 事例2：Deep Neural Network
- 時として、実装の努力では越えられない壁に直面する。
どうしよう！？
 - ✓ 最先端の技術に解決のヒントが隠されている
 - ✓ が、そのままでは使えない
 - ✓ **問題を再設定し、適切に解きなおす努力が必要**
 - ✓ うまく解ければ大幅な性能アップにつながる！

僕たちの
仕事はココ！





Denso IT Laboratory, Inc.

manbai@d-itlab.co.jp

@ambee_whisper

アイコン素材入手先 : <http://www.visualpharm.com>