第2章

統計的学習に基づく物体検出とその関連研究

物体検出は，画像から得られる特徴量を統計的学習法により学習することで構築した識別器を用いることで実現されている．本章では，統計的学習を用いた物体検出技術について述べ，特によく用いられる種々の統計的学習法について詳述する．また，これら統計的学習法を用いた物体検出を困難とする具体的な要因についても述べる．

2.1 統計的学習を用いた物体検出システム

本研究で対象とする物体検出システムについて述べる．物体検出問題は，画像中から物体の存在する領域の位置と大きさを求める問題である．物体検出は，図 2.1 に示すようにカメラで撮像した一般環境下での画像に対してウインドウスキャンを行い，各ウインドウが対象物体であるか否かを判定することで実現する．判定には，各ウインドウの画像特徴量と統計的学習により学習した識別器を用いる．統計的学習法とは，その特徴パターンが対象物であるか否かの判別基準を統計的に学習し，識別器を構築する手法である．顔検出 [1, 2] で用いられた Boosting，人検出 [3] で用いられた SVM などが多く利用されている．本章ではこれらの統計的学習法について述べる．

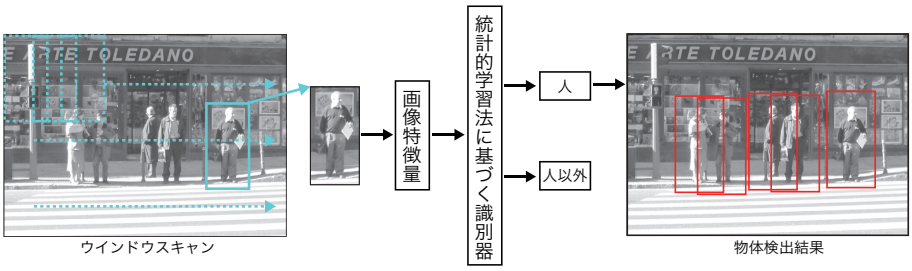

図 2.1: 統計的学習法を用いた物体検出システム（例：人検出）．

2.2 統計的学習に基づく識別器

本節では，物体検出に用いられる統計的学習法として，Boosting と SVM について述べる．

2.2.1 Boosting

複数の識別器を組み合わせることで，汎化性能の高い識別器を構築する学習法として，アンサンブル学習がある．アンサンブル学習法には Bagging や Boosting 等が存在する．Bagging はランダムにサンプリングした学習サンプルに対して識別器を多数構築して確率的に汎化性能を向上するのに対し，Boosting では前段の識別器の性能不足を補うように識別器を逐次構築することで識別性能を向上する．この特性から，Bagging は回帰問題に，Boosting は識別問題に用いられているアンサンブル学習法である．

Boosting のアプローチは，1988 年に Kearns らの提起した「弱い識別器（真の分布をそれほど高くない精度で分類する識別器）の集合は強い識別器（真の分布を高い精度で識別する識別器）となり得るか？」という疑問に根ざしている [4]．この疑問に対する解答として提案された Boosting は，識別性能がそれほど高くない識別器を逐次的に複数組み合わせ，識別性能の高い識別器を実現する．組み合わされる低性能な識別器を弱識別器 $h(\boldsymbol{x})$，複数の弱識別器 $h(\boldsymbol{x})$ から構成される高精度な識別器を強識別器 $H(\boldsymbol{x})$ と呼ぶ．この弱識別器をどのように選択，追加してより高精度な強識別器を構

築するかが Boosting の本質である．Boosting の代表的な学習アルゴリズムとして AdaBoost[5] がある．AdaBoost が提案されて以降，Real Adaboost[6]，Logit Boost[7]，Float Boost[8]，LP Boost[9] など AdaBoost の学習アルゴリズムが改良されてきた．ここでは，AdaBoost と AdaBoost を改良した Real AdaBoost の学習アルゴリズムについて述べる．

■ AdaBoost

Boosting の学習アルゴリズムでは，何らかの指標により弱識別器 $h(\boldsymbol{x})$ を逐次的に選択することで強識別器 $H(\boldsymbol{x})$ を学習する．この際の指標には重み付き誤差を用いるのが一般的である．AdaBoost の学習アルゴリズムを**アルゴリズム 2.1** に示す．AdaBoost では，学習サンプルに対する重み w という概念を取り入れ，弱識別器 $h(\boldsymbol{x})$ を選択後に学習サンプルに対して重み付けする．この学習サンプルに対する重みは，現在までに選択された弱識別器群，すなわち強識別器において識別が困難であるサンプルほど高く，次の弱識別器 $h(\boldsymbol{x})$ を選択する際に，その学習サンプルをどれほど考慮するかを表す．弱識別器を追加する度に，正しく識別したサンプルに対しては小さな重み，誤識別したサンプルに対しては大きな重みを与える．これを繰り返すことにより強識別器 $H(\boldsymbol{x})$ を学習する．

未知入力サンプルを識別する際には，図 2.2 に示すように各弱識別器 $h(\boldsymbol{x})$ の識別結果と重み α の積を計算し，これらの総和を強識別器 $H(\boldsymbol{x})$ の出力とする．この重み付き多数決といえる強識別器の出力がしきい値以上の値であれば対象物体，しきい値以下の値であれば非対象物体と識別される．

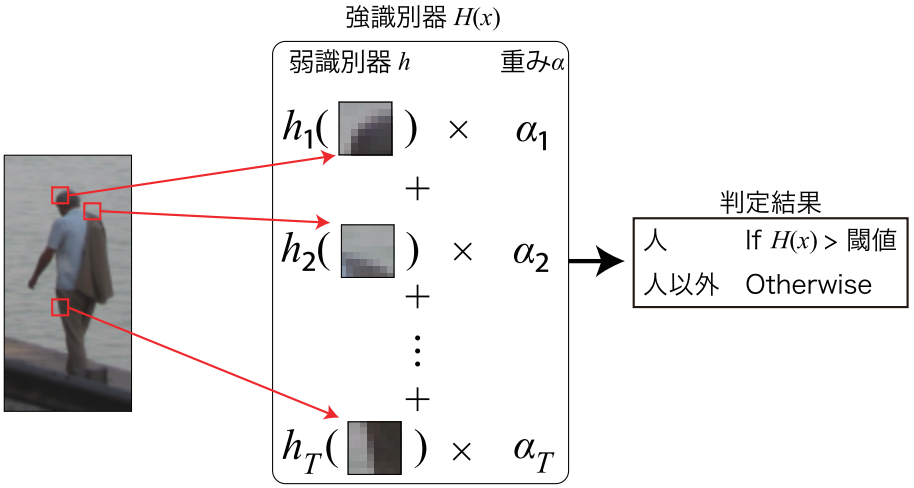

図 2.2: AdaBoost による人の識別の流れ．

AdaBoost の弱識別器 $h(\boldsymbol{x})$ は出力が 2 値であることから Discrete AdaBoost とも呼ばれる．また，AdaBoost の弱識別器 $h(\boldsymbol{x})$ はアルゴリズム的には明確に定義されておらず，識別率が 50% を超えるものであれば，どのように設計しても良いとされている．式 (2.3) で表わされる弱識別器 $h(\boldsymbol{x})$ は，文献 [2] の顔検出で用いられているしきい値処理を利用した識別器である．この弱識別器は，特徴量の値 $v(\boldsymbol{x})$ としきい値 θ を比較することにより特徴量を 2 値化する．$p\{\ \}\ \ 1,+1\langle$ はパリティ符号を表し，しきい値と特徴量の大小関係を変えるための符号である．

アルゴリズム 2.1: しきい値関数を弱識別器に用いた AdaBoost の学習アルゴリズム.

1. 入力: I 個の学習サンプル $\}\boldsymbol{x}_1, y_1\langle \ldots \}\boldsymbol{x}_I, y_I\langle$ を用意する.
   $\boldsymbol{x}$ は学習サンプル，$y_i \{\ \}\ \ 1, 1\langle$ は学習サンプルが属するクラスラベルを表す.

2. 初期化: 学習サンプルの重み D を初期化

$$D_1(i) = 1/I \tag{2.1}$$

3. 学習:
   For $t = 1, \bullet\bullet\bullet, T$ //学習ラウンド

   - 全ての弱識別器候補からエラー率 ϵ_t が最小の弱識別器候補 $h_t(\boldsymbol{x})$ を選択
   - エラー率 ϵ_t の算出

$$\epsilon_t = \sum_{i:h_t(\boldsymbol{x}_i)\neq y_i}^{I} D_t(i) \tag{2.2}$$

   - 弱識別器

$$h_t(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \ p \bullet v(\boldsymbol{x}) > p \bullet \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

   - 弱識別器 h の重み α を算出

$$\alpha_t = \frac{1}{2}\log\left(\frac{1 \quad \epsilon_t}{\epsilon_t}\right) \tag{2.4}$$

   - 学習サンプルの重み D を更新 •正規化

$$D_{t+1}(i) = \frac{D_t(i)\exp[\ \ \alpha_t y_i h_t(\boldsymbol{x}_i)]}{\sum_{i=1}^{I} D_t(i)\exp[\ \ \alpha_t y_i h_t(\boldsymbol{x}_i)]} \tag{2.5}$$

   End for

4. 出力: 強識別器

$$H(\boldsymbol{x}) = sign\left[\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right] \tag{2.6}$$

■ Real Adaboost

AdaBoost における弱識別器の出力 $h(\boldsymbol{x})$ は $\}+1,\ \ 1\langle$ であり，$\boldsymbol{x}$ の微小な変化によりしきい値処理の結果が反転すると，その重み α が反転して強識別器の出力に影響する．また，多峰性を持つような複雑な学習サンプルの分布を表現するには多くの弱識別器を必要とする．Real AdaBoost では弱識別器の出力を実数として扱う拡張を取り入れている．そのため，弱識別器の出力値は，明確に対象クラスに属する場合大きくなり，非対象クラスとの境界付近に位置する曖昧な場合では小さくなる.

これは，AdaBoost（Discrete）に比べ弱識別器 1 つあたりの分布の表現能力を大きく拡張し，少ない弱識別器数で高精度な識別を可能にする．Real AdaBoost の弱識別器も，AdaBoost と同様にどのように設計しても良いとされている．本論文では，文献 [10] で用いられた確率密度関数を利用した方法を採用する．確率密度関数は，学習サンプル全てを特徴量毎にヒストグラム化することで得られ，入力されたサンプルの特徴量の値から対象クラスらしさを確率分布で表現する．Real AdaBoost の学習アルゴリズムを**アルゴリズム 2.2** に示す．AdaBoost との大きな違いは，弱識別器の出力をしきい値による 2 値出力関数から確率密度関数 W による確率（式 (2.11)）に基づく実数値へと変更したことである．式 (2.11) は入力されたパターンの観測値 k が $0 \in K$ のどの区間に収まるかによって異なる出力を返す．学習ラウンド毎に式 (2.9)，(2.10) で確率密度関数を求めることで，サンプルの重み $D_t(i)$ を考慮し%た重み付き誤差最小の弱識別器は式 (2.8) を最大とする弱識別器として求められる．て弱識別器を選択する．サンプルの重み $D_t(i)$ は式 (2.12) により，選択された弱識別器で正解できなかったサンプルを重視するよう逐次更新される．

アルゴリズム 2.2: 確率密度関数を弱識別器に用いた Real AdaBoost の学習アルゴリズム.

1. 入力: I 個の学習サンプル $\}\boldsymbol{x}_1, y_1\langle \ldots \}\boldsymbol{x}_I, y_I\langle$ を用意する.
   $\boldsymbol{x}$ は学習サンプル，y_i { } $1, 1\langle$ は学習サンプルが属するクラスラベルを表す.

2. 初期化: 学習サンプルの重み D を初期化
$$D_1(i) = 1/I \tag{2.7}$$

3. 学習:
   For $t = 1, \bullet\bullet\bullet, T$ //学習ラウンド

   • 評価値 Z_t が最大の弱識別器 $h(\boldsymbol{x})$ を選択

   • 評価値 Z_t の算出
$$Z_t = 1 \quad 2 \Big[_k \quad \overline{W_+^k W_-^k} \tag{2.8}$$

   • 確率密度関数 W_+, W_- の算出
$$W_+^k = \Big[_{i:k\in K \wedge y_i=+1} D_t(i) \tag{2.9}$$
$$W_-^k = \Big[_{i:k\in K \wedge y_i=-1} D_t(i) \tag{2.10}$$

   • 弱識別器
$$h(\boldsymbol{x}) = \frac{1}{2} \ln \frac{W_+^k + \varepsilon}{W_-^k + \varepsilon}, \ (\varepsilon = 1/I) \tag{2.11}$$

   End for

   • 学習サンプルの重み D を更新 •正規化
$$D_{t+1}(i) = \frac{D_t(i) \exp[\ y_i h_t(\boldsymbol{x}_i)]}{\Big[_{i=1}^{I} D_t(i) \exp[\ y_i h_t(\boldsymbol{x}_i)]} \tag{2.12}$$

   End for

4. 出力: 強識別器
$$H(\boldsymbol{x}) = sign\Big] \Big[_{t=1}^{T} h_t(\boldsymbol{x}) \Big(\tag{2.13}$$

2.2.2 Support Vector Machine

Support Vector Machine(SVM) は 1960 年代に Vapnik 等が考案した Optimal Separating Hyperplane を起源とし，1990 年代になってカーネル学習法と組み合わせた非線形の識別手法へと拡張された．カーネルトリックにより拡張した SVM は，現在知られている手法の中で最もパターン認識性能の優秀な学習モデルの一つである．その大きな特徴として，マージン最大化に基づく汎化性能の獲得とカーネルトリックによる非線形問題への対応が挙げられる．

マージン最大化　まず，図 2.3(a) に示す線形分離可能な問題について考える．このような分布から線形判別関数を用いて識別境界を求める際，図 2.3(a) のように複数の識別境界を考えることができる．これらの境界は，学習サンプルに対する識別率において等価である．しかし，実際に未知入力を識別させると，図 2.3(a) 中において境界線 (1) や境界線 (3) よりも境界線 (2) の方がよい結果を示すことが多い．これは，境界線 (2) はサンプルから識別境界への距離（マージン）が大きく，サンプルと実データの差を飲み込む見込みが最も高いためである．このような，実データに対する頑健性を汎化性能という．SVM は，識別境界と境界付近のサンプル群との距離を最大とすることで汎化性能の高い識別境界を構築する．

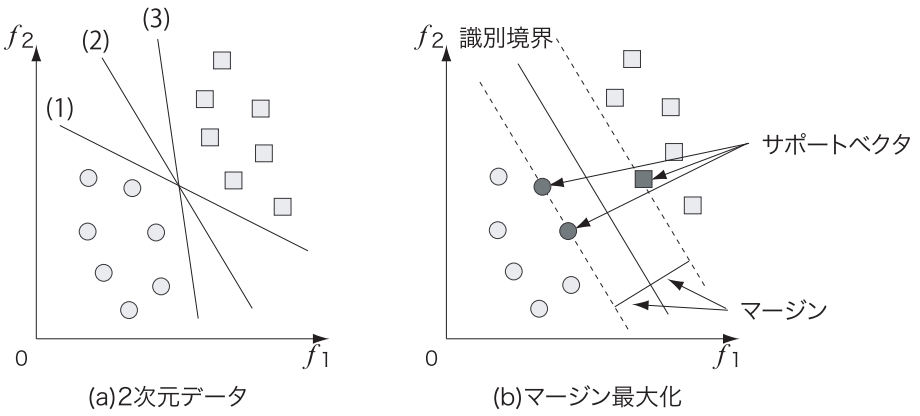

図 2.3: SVM によるマージン最大化.

各サンプルのマージンはサンプルから識別境界までの距離であるため，入力 $\boldsymbol{x}$ に対する線形識別器を

$$f(\boldsymbol{x}) = \langle \boldsymbol{w} \bullet \boldsymbol{x} | + b \tag{2.14}$$

$$= \sum_{i=1}^{n} w_i x_i + b \tag{2.15}$$

とすれば，その距離は

$$\gamma = \frac{1}{2}\left(\frac{f(\boldsymbol{x}^+)}{\|\boldsymbol{w}\|_2} - \frac{f(\boldsymbol{x}^-)}{\|\boldsymbol{w}\|_2}\right) = \frac{1}{2\|\boldsymbol{w}\|_2}\left(\langle \boldsymbol{w}\cdot\boldsymbol{x}^+ + b\right) - \left(\langle \boldsymbol{w}\cdot\boldsymbol{x}^- + b\right) \tag{2.16}$$

として表現される．ここで $\boldsymbol{x}^+$ は正であるサンプル，即ちラベル $y=1$ であるサンプルであり、$\boldsymbol{x}^-$ は負，即ち $y=0$ であるサンプルである．black また，I はサンプル数である．そこで，学習サンプル全てに正解する上で

$$\gamma = \frac{1}{2\|\boldsymbol{w}\|_2} \tag{2.17}$$

を最大とする境界を求めることで汎化性能を大きくすることができる．このマージン γ を最大化するには，ノルム $\|\boldsymbol{w}\|_2$ を最小化すればよい．マージンの最大化に関わるのは識別境界面に最も近いサンプルのみであるため，境界の構築に関わるか否かを重み付けして最適化する．最適化には，式 (2.19) に示す Lagrange の未定乗数法により得られる更新式を用いる．Lagrange の未定乗数法は束縛条件のもとで最適化を行う手法であり，本問題では識別性能とマージン双方の最大化を解く．また，この問題を二次凸最適化問題として解くために Karush-Kuhn-Tucker の相補条件（KKT 条件）の制約が存在する．最急降下法を用いて解くと，SVM の学習は**アルゴリズム 2.3** となる．

式 (2.15) で用いる識別時の重みベクトル $\boldsymbol{w}$ は

$$\boldsymbol{w} = \sum_{i=1}^{I} \alpha_i y_i \boldsymbol{x}_i \tag{2.21}$$

で求められる．ここで，識別境界付近以外のサンプルの持つ α は全て 0 であるため，重み $\alpha > 0$ となるサンプル $\boldsymbol{x}$ のみの集合を $\boldsymbol{S}$ とすると $\boldsymbol{w}$ は以下となる．

$$\boldsymbol{w} = \sum_{i\in\boldsymbol{S}} \alpha_i y_i \boldsymbol{x}_i \tag{2.22}$$

$\boldsymbol{S}$ に含まれるサンプルは，識別境界を支持することからサポートベクタと呼ばれる．ただし本アルゴリズムは線形分離可能な問題でない場合，線形分離が困難な少数の特異な点が大きな α を持ち，それらの点の境界が重視されるため，データ全体を正しく分類できない．そこで，実問題には誤差をどの程度許すかというパラメータを設置したソフトマージン SVM が用いられる．

カーネルトリック　SVM を用いることで，線形分離可能な問題に対して高精度な識別境界を得ることができる．しかし，非線形で複雑な識別課題に対しては，必ずしも良い性能の識別器を構築できるとは限らない．非線形な問題に対応するための方法として，特徴ベクトルを非線形変換して，その空間で線形の識別を行う「カーネルトリック」と呼ばれる方法が知られている．この方法を用いることで複雑な識別境界を設定できるため，SVM の性能は劇的に向上する．

アルゴリズム 2.3: SVM の学習アルゴリズム（最急降下法）.

1. 入力: I 個の学習サンプル $\}\boldsymbol{x}_1, y_1\langle \ldots \}\boldsymbol{x}_I, y_I\langle$ を用意する
   $\boldsymbol{x}$ は学習サンプル，y_i { } $1, 1\langle$ は学習サンプルが属するクラスラベルを表す

2. 初期化: Lagrange 乗数
$$\alpha_i = 0 \tag{2.18}$$

3. 学習:
   For $t = 1, \bullet\bullet\bullet, T$ //ここでは最適化回数を停止基準とする
   For $m = 1, \bullet\bullet\bullet, I$ //全サンプルについて最適化

   •α を更新
$$\alpha_i = \alpha_i + \eta(1 \quad y_i \Big[\Big]_{j=1}^{I} \alpha_j y_j \boldsymbol{x}_j^t \boldsymbol{x}_i) \tag{2.19}$$
   η は任意の学習係数
   •KKT 条件 //識別境界付近のサンプル以外の重みを 0 にする
$$\alpha_i = 0 \ \text{ if } \alpha_i < 0 \tag{2.20}$$

   End for

   End for

4. 出力: 識別器
   •学習サンプルのうち $\alpha_i > 0$ でないものを抽出し，サポートベクタ $\boldsymbol{S}$ とする
   •サポートベクタ$\boldsymbol{S}$から重み係数ベクトル $\boldsymbol{w}$ を算出
$$\boldsymbol{w} = \Big[\Big]_{i \in \boldsymbol{S}} \alpha_i y_i \boldsymbol{x}_i$$
   •識別関数
$$f(\boldsymbol{x}) \quad = \quad \rangle \boldsymbol{w} \bullet \boldsymbol{x}| + b$$

一般に，線形分離可能性はサンプル数が大きくなればなるほど難しくなり，逆に，特徴空間ベクトルの次元が大きくなるほど易しくなる．例えば，特徴ベクトルの次元がサンプルの数よりも大きいなら，どんなラベル付けに対しても線形分離可能である．図 2.4 に示すように，2 次元では分類不可能な問題をカーネル関数 $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ を用いて高次元に写像することで線形識別器である SVM を適用する．

カーネルとしては，以下に示す多項式カーネル，RBF カーネル，シグモイドカーネル等が用いられる．

多項式カーネル
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \rangle \boldsymbol{x}_i \bullet \boldsymbol{x}_j|)^p \tag{2.23}$$

RBF カーネル
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\Big) \frac{\boldsymbol{x}_i \quad \boldsymbol{x}_j \ _2}{2\sigma^2} \Big\{ \tag{2.24}$$

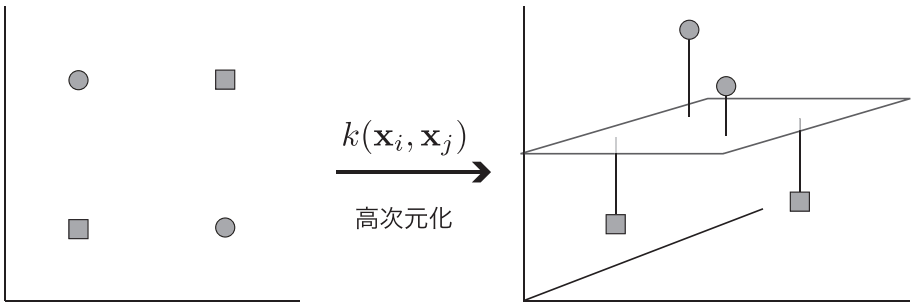

図 2.4: カーネルトリックを用いた高次元への写像.

シグモイドカーネル

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = tanh\left(c\boldsymbol{x}_i \bullet \boldsymbol{x}_j + \theta\right) \tag{2.25}$$

カーネルの選定は，サンプルの分布に対応したものを実験的に求めて設定することが多い．カーネル関数として利用できる関数の条件として，関数の対称性，Cauchy-Schwartz 不等式，Mercer の定理の 3 つを満たすことが必要である．関数の対称性とは

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i) \tag{2.26}$$

となることであり，Cauchy-Schwartz 不等式とは簡単に書けば

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = k(\boldsymbol{x}_i, \boldsymbol{x}_i)k(\boldsymbol{x}_j, \boldsymbol{x}_j) \tag{2.27}$$

を満足することである．そして，Mercer の定理は，対称性に加え

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \int_{l=1}^{\inf} \lambda_l \phi_l(\boldsymbol{x}_i)\phi_l(\boldsymbol{x}_j), \quad \lambda_l \sim 0 \tag{2.28}$$

が半正定を満たせば良い．これらのカーネルを用いることにより，非線形な識別境界を SVM によって設定することができる．

2.2.3 Boosting と SVM

Boosting は識別困難なサンプルを重視し，SVM は識別境界付近のサンプルを重視することで高精度な識別を可能とする．識別困難なサンプルとは，弱識別器で識別した際に別のクラスに間違えられるサンプルであり，つまり識別境界付近のサンプルである．そのため，両者は類似した性能を

持つことが多い．そこで，この 2 つの統計的学習法の違いは，どの弱識別器をどれだけ重視するか（AdaBoost における h，α）と，どのサポートベクタとの類似をどれだけ重視するか（SVM における w）という識別器の形の違いであるといえる．Boosting は，弱識別器を任意の特徴量 1 次元を用いたしきい値関数または確率密度関数とすることが可能である．この場合，Boosting では弱識別器がどの特徴量を捉えているかを解析することが容易である．本研究では，識別器の解析が容易であるため，統計的学習法の高精度化と効率化の基盤として，特徴量 1 次元に基づく判別関数を弱識別器とした Boosting を用いる．

2.3 マルチクラス識別器

AdaBoost，SVM は対象クラス，背景クラスの 2 クラス分類を扱う手法である．複数種の検出対象を 1 つのシステムで同時に識別するマルチクラス分類も重要なタスクであり，盛んに研究が行われている．マルチクラス分類では，検出対象毎の 2 クラス分類問題として解く One vs Other 戦略が取られることが多いが，この戦略は独立した識別器を多数用意する必要があるため，識別結果の最適な統合が困難である．そこで，1999 年に，検出対象毎の識別器を統合的に構築する手法として AdaBoost.MH[6] が提案されている．さらに 2007 年には，特徴の共有により効率的にマルチクラス識別器のを構築する Joint Boosting[11] が提案されている．

2.3.1 One vs Other 戦略

簡易にマルチクラス識別を実現する手法として，One vs Other 戦略がある．これは，対象とするクラス毎に自クラス（One）と他クラス全て（Other）の 2 クラス分類識別器を構築し，未知入力サンプルに対して，全てのクラスの出力を求める．そして，各クラスの出力を統合することでマルチクラスの判定結果を得る．しかし，複数の識別器を独立に学習するため識別器の出力値の範囲はクラスごとに大きく異なり，どのようにスケーリングして統合するかは経験的に決定する必要がある．One vs Other 戦略によるマルチクラス Boosting の強識別器は，図 2.5 のように構成される．識別は，未知入力サンプル $\boldsymbol{x}$ に対して，各クラス c の 2 クラス識別器の出力 $H^c(\boldsymbol{x})$ を次式で求め，最大の出力を持つ識別器のクラスを最終的な出力とする．

$$H^c(\boldsymbol{x}) = \prod_{t=1}^{T} h_t^c(\boldsymbol{x}) \tag{2.29}$$

$$\hat{c} = \arg\max_c H^c(\boldsymbol{x}) \tag{2.30}$$

学習アルゴリズムは各クラスに対して**アルゴリズム 2.1** をそれぞれ適用する．

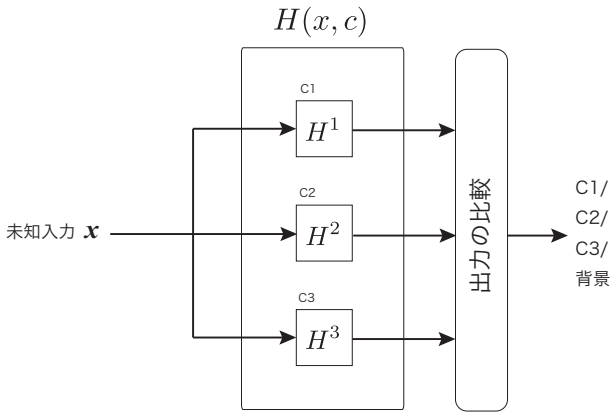

図 2.5: マルチクラス識別器.

2.3.2 AdaBoost.MH

One vs Other 戦略では，どのクラスに対してどの程度学習を行うかという点などを実験的に試行錯誤する必要がある．そこで，より効率的にマルチクラス問題を学習する方法として AdaBoost.MH[6] が提案されている．AdaBoost.MH の識別器は，One vs Other 戦略による識別器と同様に独立した各クラスに対する複数の 2 クラス識別器の集合であるが，学習過程において性能が低いクラスを重視して学習する点が異なる．まず，クラスごとにサンプルの重みを用意する．次に，各クラスのうちどれか 1 クラスを対象とした弱識別器を選択する．この選択の際，クラス c の評価値算出は c に対応する重みを用いて重み付き誤差評価を行うため，現在エラーの大きいクラスについて正解可能な弱識別器の誤差評価値は小さくなり選択されやすくなる．これを繰り返すことで，クラスに大きく偏ることなくマルチクラス学習を行うことができる．AdaBoost.MH の強識別器の構成は，One vs Other 同様に各クラスに毎に独立した識別器の集合である．強識別器の学習は逐次的に，最も性能の低いクラスの精度を改善するよう，そのクラスの弱識別器を選択する．そのため識別器を，対象とするクラスを示す変数 c を入力に加え次式のように表現する．

$$H(\boldsymbol{x}, c) = \sum_{m=1}^{M} h_m(\boldsymbol{x}, c) \tag{2.31}$$

ここで，$\boldsymbol{x}$ は入力サンプル，M は弱識別器 h_m の総数，c はクラスラベルである．全クラスに対して $H(\boldsymbol{x}, c)$ を求め，最大の値を持つ $H(\boldsymbol{x}, c)$ が対象とする c が最終的な識別結果のクラスとなる．

アルゴリズム 2.4: AdaBoost.MH の学習アルゴリズム.

1. 入力: I 個の学習サンプル $\}\boldsymbol{x}_1, z_1\langle \ldots \}\boldsymbol{x}_I, z_I\langle$ を用意する.
   $\boldsymbol{x}$ は学習サンプル，$z_i \{ \}\ \ 1, 1, ..., C\langle$ は学習サンプルが属するクラスラベルを表す.

2. 初期化: 学習サンプルの重み w を初期化
$$w_1^c(i) = 1/I \tag{2.32}$$

3. 学習:
   For $t = 1, \bullet\bullet\bullet, T$　　//学習ラウンド

   For $c = 1, \bullet\bullet\bullet, C$　　//すべてのクラス

   - 全ての c，全ての特徴量について評価値 J_{wse} を算出し，
     J_{wse} を最小とする $h_m(\boldsymbol{x}_i, c)$ を弱識別器として選択

$$J_{wse} = \prod_{c=1}^{C} \prod_{i=1}^{N} w_i^c (y_i^c \quad h_m(\boldsymbol{x}_i, c))^2 \tag{2.33}$$

   End for

   - 学習サンプルの重み w を更新

$$w_{t+1}^c(i) \quad = \quad w_t^c(i) \exp^{(y_i^c - h_m(\boldsymbol{x}_i, c))^2} \tag{2.34}$$

   End for

4. 出力: 強識別器

$$H(\boldsymbol{x}, c) = \prod_{m=1}^{M} h_m(\boldsymbol{x}, c) \tag{2.35}$$

2.3.3 Joint Boosting

AdaBoost を基に識別性能を向上した学習法として gentleboost[12] がある．Torralba らが提案した Joint Boosting[11] は，gentleboost をマルチクラス問題へ拡張している．マルチクラスへの拡張は，AdaBoost.MH のフレームワーク同様に，評価値を最小とする弱識別器を選択する形で実現する．AdaBoost.MH との違いは，弱識別器を共有し，弱識別器数の低減を可能とした点である．

■ **gentleboost**

gentleboost は 2 クラス問題に対する AdaBoost の発展形であり，AdaBoost に比べて，学習サンプルの分布を表現する能力が高い弱識別器を採用している．AdaBoost における弱識別器の出力は，しきい値判定を行った際の符号 $h(\boldsymbol{x})\ \{\ \}1,\ \ 1\langle$ と重み α の積で表現される．これは，しきい値判定結果によって出力がそのまま反転してしまい，繊細な出力が困難である．gentleboost の弱識別器は，$h(\boldsymbol{x})$ の出力を実数 $h(\boldsymbol{x})\ \{\ [\ \ 1,1]$ とし，しきい値判定の結果に応じて正負それぞれ異なる大きさを出力できる．この点から，gentleboost はしきい値処理を用いながらも，2 区間のみで確率密度関数を構成した Real AdaBoost に近い性質を持つ．従って，Discreat 型と Real 型の中間的な Boosting であるといえる．gentleboost の強識別器は以下となる．

$$H(\boldsymbol{x}) = \int_{m=1}^{M} h_m(\boldsymbol{x}) \tag{2.36}$$

ここで，$\boldsymbol{x}$ は入力サンプル，M は弱識別器 h_m の総数である．弱識別器 h_m は次式により計算される．

$$h_m(\boldsymbol{x}) = a\delta(x^f > \theta) + b \tag{2.37}$$

x^f は入力における特徴 f の値，θ はしきい値，δ は Kronecker のデルタ関数を表す．a, b はしきい値判定された際の出力を決定するパラメータである．ここで，a, b のパラメータは，それぞれしきい値以上の区間におけるサンプルの重み付き分布，しきい値以下の区間におけるサンプルの重み付き分布を式 (2.38)，式 (2.39) で表現する．

$$b = \frac{\Sigma_i w_i y_i \delta(\boldsymbol{x}_i^f \sim \theta)}{\Sigma_i w_i \delta(\boldsymbol{x}_i^f \sim \theta)} \tag{2.38}$$

$$a + b = \frac{\Sigma_i w_i y_i \delta(\boldsymbol{x}_i^f > \theta)}{\Sigma_i w_i \delta(\boldsymbol{x}_i^f > \theta)} \tag{2.39}$$

また，y は $\boldsymbol{x}$ が正例か負例かを示すラベルであり，$\}1,\ \ 1\langle$ をとる．式について指数評価関数 J を最も低くする弱識別器 h を逐次選択する．

$$J = E\left]e^{-yH(\boldsymbol{x})}\right\surd \tag{2.40}$$

その際，各サンプル i の重みが $w_i = e^{-z_i H(\boldsymbol{x}_i)}$ として更新される．gentleboost を AdaBoost.MH 同様マルチクラスに拡張した際の強識別器は次式である．

$$H(\boldsymbol{x}, c) = \sum_{m=1}^{M} h_m(\boldsymbol{x}, c) \tag{2.41}$$

このとき，弱識別器 h_m は以下となる．

$$h_m(\boldsymbol{x}, c) = a\delta(\boldsymbol{x}^f > \theta) + b \tag{2.42}$$

ここで，$\boldsymbol{x}$ は入力サンプル，M は弱識別器 h_m の総数，c はクラスラベルである．また，$\boldsymbol{x}^f$ はサンプル $\boldsymbol{x}$ の f 番目の要素値を表す．

■ 特徴を共有するマルチクラス学習

Joint Boosting は，効率的なマルチクラス分類のための Boosting 手法であり各クラスに共通する特徴量（弱識別器）を共用しながら学習を行う．

AdaBoost.MH では，各クラスに対して弱識別器を設定して強識別器を逐次構築する．そのため，クラス間で共有可能な特徴量を共有できない．Joint Boosting では，ラベル c の対象を各クラスだけでなく，その和集合に対しても拡張し，各クラスとクラスの和集合をノード $S(n)$ で表現する．

例えば 3 クラス（$1, 2, 3$）の場合，ノード $S(n)$ は $S(1) : (1)$，$S(2) : (2)$，$S(3) : (3)$，$S(4) : (1, 2)$，$S(5) : (1, 3)$，$S(6) : (2, 3)$，$S(7) : (1, 2, 3)$ をそれぞれ表現する．これらノードを並列に置き，各ノードについて誤差最小の弱識別器 $h_m(v, S(n))$ を逐次探索する．

ここで，探索で得られた弱識別器を各ノードごとに格納する集合 $G^{S(n)}$ を図 2.6 のように定義する．弱識別器集合 $G^{S(n)}$ はそれぞれ $S(n)$ の表現するクラス集合に対し識別性能を持つため，図 2.6 中の上位であるほど複数のクラスで共有可能な識別器が格納されている．つまり，G^{123} は検出対象 1〜3 全てと背景を，G^{12} は検出対象 1 と 2 全てと背景を分離するのに有効な弱識別器群である．この G^{123} と G^{12} は検出対象 1 に対する強識別器 $H(\boldsymbol{x}, 1)$，検出対象 2 に対する強識別器 $H(\boldsymbol{x}, 2)$ で共有される．この共有により，同様の弱識別器が各クラス毎に学習される無駄を防ぐことができる．Joint Boosting による各クラスに対する最終的な強識別器はそれぞれ以下となる．

$$\begin{aligned} H(\boldsymbol{x}, 1) &= G^{123}(\boldsymbol{x}) + G^{12}(\boldsymbol{x}) + G^{13}(\boldsymbol{x}) + G^{1}(\boldsymbol{x}) \\ H(\boldsymbol{x}, 2) &= G^{123}(\boldsymbol{x}) + G^{12}(\boldsymbol{x}) + G^{23}(\boldsymbol{x}) + G^{2}(\boldsymbol{x}) \\ H(\boldsymbol{x}, 3) &= G^{123}(\boldsymbol{x}) + G^{13}(\boldsymbol{x}) + G^{23}(\boldsymbol{x}) + G^{3}(\boldsymbol{x}) \end{aligned} \tag{2.43}$$

学習は，あるクラス集合 $S(n)$ に対応する弱識別器 $h(\boldsymbol{x})$ を毎ラウンド選択することで進む．$h(\boldsymbol{x}, c)$ はノード $S(n)$ に含まれるクラス全てで共有される．その際，最良の弱識別器を選択するために以下の評価関数 J_{wse} という重み付き二次誤差を用いる．

$$J_{wse} = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c (y_i^c \quad h_m(\boldsymbol{x}_i, c))^2 \tag{2.44}$$

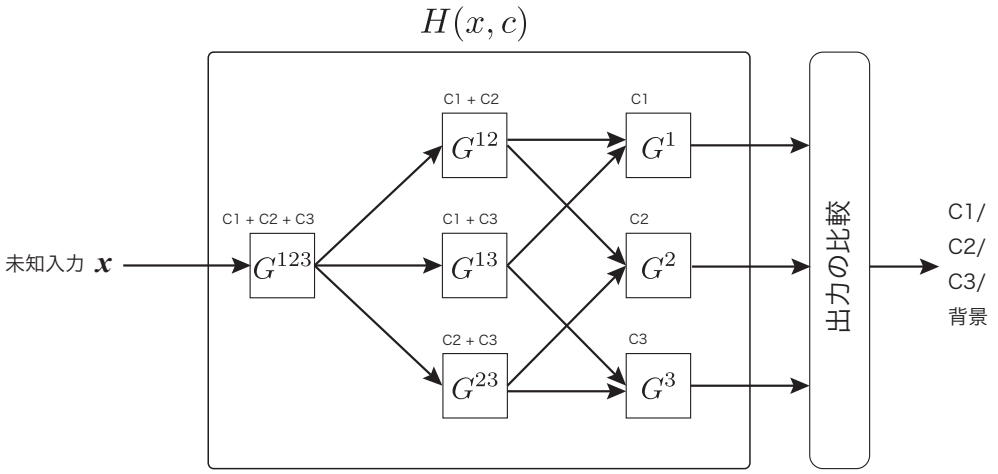

図 2.6: 弱識別器の共有.

C はクラス数，c は各クラスラベルを表し，集合に対する弱識別器 $h_m(\boldsymbol{x}_i, S(n))$ を再度 $h_m(\boldsymbol{x}_i, c)$ として c 毎に表現している．また，y^c は $\boldsymbol{x}$ がクラス c に含まれるかどうかを示すラベルであり，$\}1,\ \ 1\langle$ をとる．この評価値を最小にするよう弱識別器を選択することで，汎化性能の高い識別器の構築が期待できる．また，以下の拡張により弱識別器をノードによるマルチクラス識別に適応させている．

$$h_m(\boldsymbol{x}, c) = \begin{cases} a\delta(\boldsymbol{x}^f > \theta) + b & \text{if } c \{\ S(n) \\ k^c & \text{otherwise} \end{cases} \tag{2.45}$$

ここで，a, b, k^c のパラメータはそれぞれ以下となる．

$$b = \frac{\sum_{c \in S(n)} \sum_i w_i^c z_i^c \delta(\boldsymbol{x}_i^f \sim \theta)}{\sum_{c \in S(n)} \sum_i w_i^c \delta(\boldsymbol{x}_i^f \sim \theta)} \tag{2.46}$$

$$a + b = \frac{\sum_{c \in S(n)} \sum_i w_i^c z_i^c \delta(\boldsymbol{x}_i^f > \theta)}{\sum_{c \in S(n)} \sum_i w_i^c \delta(\boldsymbol{x}_i^f > \theta)} \tag{2.47}$$

$$k^c = \prod_{c \notin S(n)} \frac{\sum_i w_i^c z_i}{\sum_i w_i^c} \tag{2.48}$$

b はしきい値以下の区間における確率密度を，a はしきい値以上の区間における確率密度をそれぞれ反映し，負例が多い区間では負方向，正例が多い区間では正方向の強度を示す．k^c は弱識別器 h_m が対象としないクラスについて識別器は k^c を応答として返し，識別関数は介在しない．

2.4 統計的学習による物体検出を困難とする要因

本節では物体検出が困難となる要因について整理する．

現在主流である物体検出は，画像局所特徴量と統計的学習法の組み合わせにより実現され，統計的学習法による識別器にはSVMやBoostingなどの利用が提案されている．特に，Boostingは弱識別器として採用された特徴量のみを計算すればよいため計算コストが低く，特徴量の共起関係の表現が容易であるため，人，顔，車両等の物体検出に多く用いられている [13][14][15].

検出対象となる物体の見えは，照明，視点によって大きく異なり，検出対象の向きや姿勢により，形状，テクスチャなどが全く異なる画像が得られる．人検出は，検出対象が非剛体であることと，その個体差の多様性から物体検出の中でも特に困難とされている．文献 [16] では，人検出を困難とする要因について，以下のように[1]まとめられている．

1. 見えの個体差
   「見えの個体差とは，人の衣服や体格など同一人クラス内における各個人の差である．衣服の色は人によって異なるため，人検出への有効な情報として利用しにくい．また，体格は大人と子供，性別によっても大きく異なるため，頭部や胴体，足などの見えや位置が人画像毎に異なる.」
2. 向きの変化
   「人を正面から撮影する場合と横から撮影する場合では人の見えが異なる.」
3. 姿勢の変化
   「人は非剛体な物体であり，自由な姿勢をとることができるため，姿勢により人の形状は大きく変化する.」
4. 視点の変化
   「人の向きの違いと似ているが，ここではカメラの俯角の違いによる人の見えの違いを表わす．人を正面から撮影した人画像と斜め上から撮影した人画像では，人の見えは大きく異なる.」
5. 人領域の隠れ
   「人とカメラの間に物体が存在する場合，画像上の人領域が部分的に観測できなくなる．そのため，部分的に欠損した人画像から人を検出することになる.」
6. 複雑な背景
   「背景画像が複雑なテクスチャを含むことがある．特に，人画像のように縦エッジが連続しているような背景画像は，部分的な領域のみを見た場合には人画像に似ているため，誤識別する問題がある.」

これらの問題に対し，Dalalらは，Histograms of Oriented Gradients(HOG)特徴量とSVMによる人検出を提案している [3]．HOG特徴量は，輝度勾配情報に基づき形状の人らしさを抽出する特徴であり，体型，服装等の個体差による問題を大きく低減することに寄与している．山内らは単純なHOGだけでなく，特徴間の共起表現を導入したBoostingによるJoint特徴量を提案している [13]．高木らも同様に，Geometric Contextによる共起表現を導入したBoostingにより高精度な人と車両の検出を実現している [14]．しかし，人のように変動を多く含む物体を検出する際，変化の大きい学習サンプルを同一のクラスとして扱い識別器を構築するのは困難となる場合がある．文献 [13][14] の手法は特徴量の表現力を向上させることで精度向上しているが，学習自体はReal AdaBoostなど

[1]原文では番号付けされていないが，本論文では参照のため番号を付けた．

の既存の 2 クラス学習を用いており，統計的学習法の改良については検討されていない．

2.4.1　物体の多様性に起因する問題

図 2.7 は人検出用データベースとして公開されている INRIA person dataset[3] に含まれている画像である．全ての人画像は人が中心となるよう画像上で正規化されているが，体幹の傾き，顕著な年齢差，複雑な姿勢等の変化を含んでいる．このような変化は，人検出を困難とする要因のうち 1，2，3 に該当する．これらの変化（変動）を持つ複雑な対象物体を一つの識別器で学習すると，全体に共通する情報が減少し，識別性能の低下を招くことになる．

図 2.7: 物体の見えの変化 (INRIA person dataset)．

2.4.2　カメラの設置環境に起因する問題

カメラの設置環境が異なると，以下に挙げる変化が生じる．

≤ 外乱等による特徴量の変化

学習サンプルと設置環境における光量の差や背景の違いなどのノイズにより，色やエッジの出力される値が変化する．これにより，しきい値や区間関数を用いた弱識別器が誤識別を起こす．

≤ 視点の変化

物体の向きの違いと似ているが，ここではカメラの俯角の違いによる物体の見えの違いを表わす．図 2.8(a)(b) に示す例のように，物体を正面から撮影した物体画像と斜め上から撮影した物体画像では，物体の見えは大きく異なり，未検出が発生する．

これらは人検出を困難とする要因の 4，5，6 に該当し，設置環境に依存した問題といえる．

(a)INRIA評価画像　(b)特定シーン画像

図 2.8: 視点の変化により発生する未検出．

特に姿勢の変化は深刻な問題で，俯角の大きさやカメラの高さなどによって縦横比やパーツのサイズなどは全く異なってしまう．しかし，実店舗や屋外環境への検出システム設置を考えた場合には，設置やメンテナンスのコスト面，また利用者の心理面からカメラの台数は可能な限り少なくすることが望ましいため，カメラをずらりと並べるのではなく俯角をつけて一台でできるだけ広範囲の視野をカバーすることが求められる．このような場合，システムの設置箇所毎，あるいはカメラ毎に異なる俯角に合せてシステムを構築し直すことが必要となる．しかし，各俯角や背景毎に大量のサンプルを収集することは非常にコストが大きく，困難であるため，設置環境において簡易な調整により十分な性能を得られる識別器の調整手法が必要である．

また，まるで見えが異なる場合，調整ではなく識別器の再構築を求められる．しかし，統計的学習法で必要とされる学習サンプルは，図 2.9 に示す例のように，切り出した画像中の人の位置やサイズなどが標準化されている必要があり，技術と人的コストを要する．もし学習サンプル中の人画像位置にズレがあれば，人を中心に検出することができず，十分に統計的な学習が得られない．サイズについても同様で，様々なサイズで映り込んでいる場合，学習や検出を困難とする要因となる．これらの要件を満たす学習サンプルを人手で収集するためには，多くの専門的な人間が多大な時間を費やす必要があり，非常にコストが高い作業であるといえる．

これらの問題を解決するアプローチとして，少数の学習サンプルからスケール変化や回転，ノイズの付加などの実環境で測定されうる変動を含むように変形させた学習サンプルを生成し，生成したサンプルを用いて識別器を学習する生成型学習法 [17] が提案されている．文献 [18] では，認識対象である車載カメラから見える路面上の標識に対して，光学ぼけや動きぼけ等を考慮したサンプルを生成して学習に利用した．また，文献 [19] では道路標識に対して，位置ずれや回転などの形状の

(a) 学習サンプル画像　　(b) 平均勾配画像

図 2.9: 標準的な学習サンプルの例（INRIA Person Dataset）.

変化，背景などのテクスチャの変化，反射や影などの色の変化を考慮した生成モデルを用いて学習サンプルを生成した．しかし，これらの手法が認識対象とするのは比較的簡単な 2 次元パターンであり，人のような非剛体で複雑な形状や姿勢を持つ物体に対しては，同様のアプローチで学習サンプルを生成するのは困難である．これらのことから，特定環境における統計的学習法に用いるサンプル，特に非剛体の対象物体の学習サンプルを低コストで収集する手法が望まれている．さらに，近年では検出技術の向上に伴い特徴量の次元数は数千 $\in$ 数千万と膨大な量を用いることが多く，これら多くの次元から大量に得た多くの画像に共通する特徴量を探し学習を行うという，統計的学習による識別器構築自体に必要なコストも無視できなくなりつつある．これらの要因が物体検出の実用化を困難にしている．

2.5　まとめ

本章では物体検出技術において重要な要素である統計的学習法に基づく識別器について述べた後，それを実用化する際に困難となる要因について述べた．

統計的学習法は入力された学習サンプルについて正しい判別を行うよう学習し，非常に高い精度で正解する．しかし，入力された学習サンプルが多くのバリエーションを持つ複雑な問題であった場合，それを解決する識別器もまた複雑となり，精度は低下する．3 章では，この問題を解決するために統計的学習法への分割統治戦略の導入を提案する．

物体検出を実用化する際に問題となるのは，設置環境への適応である．一般に学習サンプルと実際の設置環境で得られる画像は外乱などにより異なっており，その相違に起因して識別性能が低下する．そこで，4 章では特徴量がどれだけ識別に貢献しているかに基づき，特徴量の削減や調整により簡易に設置環境へ適応する手法を提案する．

また，それでも適応不可能な相違がある場合は再学習が必要となるが，その際に学習サンプルを再収集する人的コストの大きさが問題である．基本的に，検出性能と良質なサンプルの量は不可分の関係にあり，設置環境に適したサンプルを大量に用意するための人的コストを低減する必要があ

る．そこで，5 章では学習サンプルの自動生成を用いて，再学習に使用するサンプルを大量かつ安価に用意し，設置環境に対して高精度な識別器を構築する手法を提案する．さらに，学習自体に必要なコストを削減するために，6 章では転移学習を導入し，高速化に伴い発生する識別性能の低下を防ぐハイブリッド型転移学習による識別器を提案する．