Object Recognition 2: Appearance-based, Tracking
Contour-based Object Recognition

1. SVM (Support Vector Machine)
2. Classifier-based Object Recognition
4. Kalman Filter-based Tracking, A Posteriori Probability
Support Vector Machine
Consider the problem of finding a separating hyperplane for a linearly separable dataset \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \), \( x \in \mathbb{R}^D \), \( y \in \{-1, +1\} \).

- Which of the infinite hyperplanes should we choose?
  - Intuitively, a hyperplane that passes too close to the training examples will be sensitive to noise and, therefore, less likely to generalize well for data outside the training set.
  - Instead, it seems reasonable to expect that a hyperplane that is farthest from all training examples will have better generalization capabilities.

Therefore, the optimal separating hyperplane will be the one with the largest margin, which is defined as the minimum distance of an example to the decision surface.
Optimal Separating Hyperplanes [6]

Distance between a plane and a point

\[ |AN| = |AB| \cos \theta = |AB| \frac{\overrightarrow{AB} \cdot \vec{w}}{|\overrightarrow{AB}| |\vec{w}|} = \frac{\overrightarrow{AB} \cdot \vec{w}}{|\vec{w}|} \]

\[ = \left( x_{1A} - x_{1B}, x_{2A} - x_{2B} \right)^T \left( w_1, w_2 \right) \]

\[ = \frac{w^T x_A - w^T x_B}{|\vec{w}|} = \frac{w^T x_A + b}{|\vec{w}|} \]
Optimal Separating Hyperplanes [6]

- Since we want to maximize the margin, let’s express it as a function of the weight vector and bias of the separating hyperplane
  - From basic trigonometry, the distance between a point $x$ and a plane $(w, b)$ is $\frac{|w^Tx + b|}{\|w\|}$

- Noticing that the optimal hyperplane has infinite solutions by simply scaling the weight vector and bias, we choose the solution for which the discriminant function becomes one for the training examples closest to the boundary $|w^Tx_i + b| = 1$
  - This is known as the canonical hyperplane

- Therefore, the distance from the closest example to the boundary is $\frac{|w^Tx + b|}{\|w\|} = \frac{1}{\|w\|}$

- And the margin becomes $m = \frac{2}{\|w\|}$
Optimal Separating Hyperplanes [6]

- Therefore, the problem of maximizing the margin is equivalent to

  minimize $J(w) = \frac{1}{2}||w||^2$

  subject to $y_i(w^T x_i + b) \geq 1 \ \forall i$

- Notice that $J(w)$ is a quadratic function, which means that there exists a single global minimum and no local minima

- To solve this problem, we will use classical Lagrangian optimization techniques

  - We first present the Kuhn-Tucker Theorem, which provides an essential result for the interpretation of Support Vector Machines
Consider the two-dimensional optimization problem:

maximize $f(x, y)$
subject to $g(x, y) = c$.

We can visualize contours of $f$ given by

$f(x, y) = d$

Find $x$ and $y$ to maximize $f(x, y)$ subject to a constraint (shown in red) $g(x, y) = c$. 

Lagrange Multipliers [9]

When \( f(x,y) \) becomes maximum on the path of \( g(x,y)=c \), the contour line for \( g=c \) meets contour lines of \( f \) \textbf{tangentially}. Since the gradient of a function is perpendicular to the contour lines, this is the same as saying that the gradients of \( f \) and \( g \) are parallel.

\[
\nabla_{x,y} f = -\lambda \nabla_{x,y} g,
\]
where
\[
\nabla_{x,y} f = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)
\]
and
\[
\nabla_{x,y} g = \left( \frac{\partial g}{\partial x}, \frac{\partial g}{\partial y} \right)
\]

Contour map. The red line shows the constraint \( g(x,y) = c \). The blue lines are contours of \( f(x,y) \). The point where the red line tangentially touches a blue contour is our solution.
Lagrange Multipliers [9]

To incorporate these conditions into one equation, we introduce an auxiliary function

\[ \Lambda(x, y, \lambda) = f(x, y) + \lambda \cdot (g(x, y) - c) \]

and solve

\[ \nabla_{x,y,\lambda} \Lambda(x, y, \lambda) = 0. \]
Kuhn–Tucker Theorem [6]

- Given an optimization problem with convex domain $\Omega \subseteq \mathbb{R}^N$

  minimize $f(z)$ $\quad z \in \Omega$

  subject to $g_i(z) \leq 0 \quad i = 1, \ldots, k$

  $h_i(z) = 0 \quad i = 1, \ldots, m$

- with $f \in C^1$ convex and $g_i$, $h_i$ affine, necessary and sufficient conditions for a
  normal point $z^*$ to be an optimum are the existence of $\alpha^*$, $\beta^*$ such that

  $\frac{\partial L(z^*, \alpha^*, \beta^*)}{\partial z} = 0$

  $\frac{\partial L(z^*, \alpha^*, \beta^*)}{\partial \beta} = 0$

  $\alpha_i^* g_i(z^*) = 0 \quad i = 1, \ldots, k$

  $g_i(z^*) \leq 0 \quad i = 1, \ldots, k$

  $\alpha_i^* \geq 0 \quad i = 1, \ldots, k$

- $L(z, \alpha, \beta)$ is known as a generalized Lagrangian function

- The third condition is known as the Karush-Kuhn-Tucker (KKT) complementary
  condition. It implies that for active constraints $\alpha_i \geq 0$; and for inactive constraints $\alpha_i = 0$.

  - As we will see in a minute, the KKT condition allows us to identify the training examples that
    define the largest margin hyperplane. These examples will be known as Support Vectors.
The Lagrangian Dual Problem [6]

- Constrained minimization of \( J(w) = \frac{1}{2} ||w||^2 \) is solved by introducing the Lagrangian

\[
L_p(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{N} \alpha_i [y_i (w^T x_i + b) - 1]
\]

- which yields an unconstrained optimization problem that is solved by:
  - minimizing \( L_p \) with respect to the primal variables \( w \) and \( b \), and
  - maximizing \( L_p \) with respect to the dual variables \( \alpha_i \geq 0 \) (the Lagrange multipliers)
- Thus, the optimum is defined by a saddle point (see below for illustration)

- This is known as the **Lagrangian primal problem**

A saddle point
Dual Problem [10]

Minimize (in $w, b$)

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

Subject to (for any $i=1,...,n$)

$$c_i(w \cdot x_i - b) \geq 1.$$ 

One could be tempted to expressed the previous problem by means of non-negative Lagrange multipliers $\alpha_i$ as

$$\min_{w,b,\alpha} \left\{ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [c_i(w \cdot x_i - b) - 1] \right\}$$

we could find the minimum by sending all $\alpha_i$ to $\infty$. Nevertheless the previous constrained problem can be expressed as

$$\min_{w,b} \max_{\alpha} \left\{ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [c_i(w \cdot x_i - b) - 1] \right\}$$

This is we look for a saddle point.
The Lagrangian Dual Problem [6]

- To simplify the primal problem, we eliminate the primal variables \((w,b)\) using the first Kuhn-Tucker condition \(\partial J/\partial z=0\)
  
  - Differentiating \(L_p(w,b,\alpha)\) with respect to \(w\) and \(b\), and setting to zero yields
    
    \[
    \frac{\partial L_p(w,b,\alpha)}{\partial w} = 0 \implies w = \sum_{i=1}^{N} \alpha_i y_i x_i \\
    \frac{\partial L_p(w,b,\alpha)}{\partial b} = 0 \implies \sum_{i=1}^{N} \alpha_i y_i = 0
    \]
  
  - Expansion of \(L_p\) yields
    
    \[
    L_p(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i
    \]

  - Using the optimality condition \(\partial J/\partial w=0\), the first term in \(L_p\) can be expressed as
    
    \[
    w^T w = w^T \sum_{i=1}^{N} \alpha_i y_i x_i = \sum_{i=1}^{N} \alpha_i y_i w^T x_i = \sum_{i=1}^{N} \alpha_i y_i \left( \sum_{j=1}^{N} \alpha_j y_j x_j \right)^T x_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j
    \]
  - The second term in \(L_p\) can be expressed in the same way
  - The third term in \(L_p\) is zero by virtue of the optimality condition \(\partial J/\partial b=0\)
The Lagrangian Dual Problem [6]

- Merging these expressions together we obtain
  \[ L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \]
  - Subject to the (simpler) constraints \( \alpha_i \geq 0 \) and \( \sum_{i=1}^{N} \alpha_i y_i = 0 \)

  This is known as the **Lagrangian dual problem**

**Comments**
- We have transformed the problem of finding a saddle point for \( L_P(w,b) \) into the easier one of maximizing \( L_D(\alpha) \)
  - Notice that \( L_D(\alpha) \) depends on the Lagrange multipliers \( \alpha \), not on \( (w,b) \)
- The primal problem scales with dimensionality (\( w \) has one coefficient for each dimension), whereas the dual problem scales with the amount of training data (there is one Lagrange multiplier per example)
- Moreover, in \( L_D(\alpha) \) the training data appears only as dot products \( x_i^T x_j \)
  - As we will see in the next lecture, this property can be cleverly exploited to perform the classification in a higher (e.g., infinite) dimensional space
Support Vectors [6]

- The KKT complementary condition states that, for every point in the training set, the following equality must hold:

  \[ \alpha_i [y_i (w^T x_i + b) - 1] = 0 \quad \forall i = 1 \ldots N \]

- Therefore, for each example, either \( \alpha_i = 0 \) or \( y_i (w^T x_i + b - 1) = 0 \) must hold.
- Those points for which \( \alpha_i > 0 \) must then lie on one of the two hyperplanes that define the largest margin (only at these hyperplanes the term \( y_i (w^T x_i + b - 1) \) becomes zero).
  - These points are known as the Support Vectors.
- All the other points must have \( \alpha_i = 0 \).
- Note that only the support vectors contribute to defining the optimal hyperplane:

  \[ \frac{\partial J(w, b, \alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \]
  - NOTE: the bias term \( b \) is found from the KKT complementary condition on the support vectors.
- Therefore, the complete dataset could be replaced by only the support vectors, and the separating hyperplane would be the same.
Non-separable Case [6]

- The solution for the non-separable case is to introduce slack variables $\xi_i$ that relax the constraints of the canonical hyperplane equation:
  \[ y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i = 1...N \]

- The slack variables measure deviation from the ideal condition:
  - For $0 \leq \xi \leq 1$, the data point falls on the right side of the separating hyperplane but within the region of maximum margin.
  - For $\xi > 1$, the data point falls on the wrong side of the separating hyperplane.
Non-separable Case [6]

How does the optimization problem change with the introduction of slack variables?

- Our goal is to find a hyperplane with minimum misclassification rate
- This may be accomplished by minimizing the following objective function

$$\Phi(\xi) = \sum_{i=1}^{N} l(\xi_i - 1)$$

where

$$l(\xi_i) = \begin{cases} 0 & \text{if } \xi_i \leq 0 \\ 1 & \text{if } \xi_i > 0 \end{cases}$$

- subject to the constraints on $||w||^2$ and the perceptron equation
- $\Phi(\xi)$ represents the total number of misclassified samples

- Unfortunately, minimization of $\Phi(\xi)$ is a difficult combinatorial problem (NP-complete) due to the non-linearity of the indicator function $l(\xi_i)$
Non-separable Case [6]

- Instead, we approximate $\Phi(\xi)$ by
  \[ \Phi'(\xi) = \sum_{i=1}^{N} \xi_i \]

- which is an upper bound on the number of misclassifications, and minimize the joint objective function
  \[ J(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \]

- subject to
  \[
  \begin{cases}
  y_i(w^Tx_i + b) \geq 1 - \xi_i & \forall i \\
  \xi_i \geq 0 & \forall i
  \end{cases}
  \]

- Interpretation of $C$
  - Parameter $C$ represents a trade-off between misclassification and capacity
    - Large values of $C$ favor solutions with few misclassification errors
    - Small values of $C$ denote a preference towards low-complexity solutions
  - Therefore, this parameter can be viewed as a regularization parameter (recall ridge-regression in Lecture 17), the difference being that the minimization problem is now subject to constraints
  - A suitable value for $C$ is typically determined through cross-validation
Non-separable Case [6]

Using a procedure similar to the one in the previous lecture, we can derive the dual problem as

\[
L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

- subject to the constraints
  \[
  \begin{align*}
  &\sum_{i=1}^{N} \alpha_i y_i = 0 \\
  &0 \leq \alpha_i \leq C \quad i = 1 \ldots N
  \end{align*}
  \]

Comments

- Notice that neither the slack variables nor their associated Lagrange multipliers appear in the formulation of the dual problem.
- Therefore, this represents the same optimization problem as the linearly separable case, with the exception that the constraints \( \alpha_i \geq 0 \) have been replaced by the more restrictive constraints \( 0 \leq \alpha_i \leq C \).
  - The optimum solution for the weight vector remains the same:
    \[
    w = \sum_{i=1}^{N} \alpha_i y_i x_i
    \]
  - and the bias can be found by choosing a training point for which \( 0 < \alpha_i < C \) (\( \xi_i = 0 \)), and solving the KKT condition:
    \[
    \alpha_i [y_i (w^T x_i + b) - 1 + \xi_i] = 0
    \]
Non-linear SVMs [6]

- The power of SVMs resides in the fact that they represent a robust and efficient implementation of the principle in Cover’s theorem on the separability of patterns
  - “A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space”

- Based on this principle, SVMs operate in two stages
  - Perform a non-linear mapping of the feature vector $x$ onto a high-dimensional space that is hidden from the inputs or the outputs
  - Construct an optimal separating hyperplane in the high-dimensional space

$$x \xrightarrow{\varphi(x)} z \xrightarrow{w^Tz} y$$
Non-linear SVMs [6]

\[ \varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]

\[ (x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \]
Non-linear SVMs [6]

- Naïve application of this concept by simply projecting to a high-dimensional non-linear manifold has two major problems
  - **Statistical**: operation on high-dimensional spaces is ill-conditioned due to the "curse of dimensionality" and the subsequent risk of overfitting
  - **Computational**: working in high-dimensions requires higher computational power, which poses limits on the size of the problems that can be tackled
- SVMs bypass these two problems in a robust and efficient manner
  - First, generalization capabilities in the high-dimensional manifold are ensured by enforcing a **largest margin** classifier
    - Recall that generalization in SVMs is strictly a function of the margin (or the VC dimension), regardless of the dimensionality of the feature space
  - Second, projection onto a high-dimensional manifold is only **implicit**
    - Recall that the SVM solution depends only on the dot product $\langle x_i, x_j \rangle$ between training examples
    - Therefore, operations in high dimensional space $\varphi(x)$ do not have to be performed explicitly if we find a function $K(x_i, x_j)$ such that $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$
    - $K(x_i, x_j)$ is called a **kernel** function in SVM terminology

E-mail: hogijung@hanyang.ac.kr
http://web.yonsei.ac.kr/hgjung
Implicit Mappings: An Example [6]

Consider a pattern recognition problem in $\mathbb{R}^2$

- Assume we choose a kernel function $K(x_i, x_j) = (x_i^T x_j)^2$
- Our goal is to find a non-linear projection $\varphi(x)$ such that $(x_i^T x_j)^2 = \varphi(x_i)^T \varphi(x_j)$
- Performing the expansion of $K(x_i, x_j)$

$$K(x_i, x_j) = (x_i^T x_j)^2 = \left(\begin{array}{c} x_{1,1} x_{2,1} \\ x_{1,1} x_{2,2} \end{array}\right) \left(\begin{array}{c} x_{2,1} \\ x_{2,2} \end{array}\right)$$

$$= (x_{1,1} x_{2,1} + x_{1,2} x_{2,2})^2 = x_{1,1}^2 x_{2,1}^2 + 2 x_{1,1} x_{2,1} x_{1,2} x_{2,2} + x_{1,2}^2 x_{2,2}^2 =$$

$$= (x_{1,1}^2, \sqrt{2} x_{1,1} x_{1,2}, x_{1,2}^2) \left(\begin{array}{c} x_{2,1}^2, \sqrt{2} x_{2,1} x_{2,2}, x_{2,2}^2 \end{array}\right)$$

- where $x_{i,k}$ denotes the $k$-th coordinate of example $x_i$
- So in using the kernel $K(x_i, x_j) = (x_i^T x_j)^2$, we are implicitly operating on a higher-dimensional non-linear manifold defined by

$$\varphi(x_i) = \left(\begin{array}{c} x_{i,1}^2 \\ \sqrt{2} x_{i,1} x_{i,2} \\ x_{i,2}^2 \end{array}\right)$$

- Notice that the inner product $\varphi(x_i)^T \varphi(x_j)$ can be computed in $\mathbb{R}^2$ the original space by means of the kernel $(x_i^T x_j)^2$ without ever having to project onto $\mathbb{R}^3$!
Kernel Methods [6]

Let's now see how to put together all these concepts.

- Assume that our original feature vector $x$ lives in a space $\mathbb{R}^D$.
- We are interested in non-linearly projecting $x$ onto a higher dimensional implicit space $\varphi(x) \in \mathbb{R}^{D_1}$ ($D_1 > D$) where classes have a better chance of being linearly separable.
  - Notice that we are not guaranteeing linear separability, we are only saying that we have a better chance because of Cover’s theorem.
- The separating hyperplane in $\mathbb{R}^{D_1}$ will be defined by
  \[ \sum_{j=1}^{D_1} w_j \varphi_j(x) + b = 0 \]
- To eliminate the bias term $b$, let's augment the feature vector in the implicit space with a constant dimension $\varphi_0(x) = 1$.
  - Using vector notation, the resulting hyperplane becomes
    \[ w^T \varphi(x) = 0 \]
- From our previous results, the optimal (maximum margin) hyperplane in the implicit space is given by
  \[ w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i) \]
Kernel Methods [6]

- Merging this optimal weight vector with the hyperplane equation

\[ w^T \varphi(x) = 0 \Rightarrow \]
\[ \left( \sum_{i=1}^{N} a_i y_i \varphi(x_i) \right)^T \varphi(x) = 0 \Rightarrow \]
\[ \sum_{i=1}^{N} a_i y_i \varphi(x_i)^T \varphi(x) = 0 \]

- and, since \( \varphi(x_i)^T \varphi(x_j) = K(x_i, x_j) \), the optimal hyperplane becomes

\[ \sum_{i=1}^{N} a_i y_i K(x_i, x) = 0 \]

- Therefore, classification of an unknown example \( x \) is performed by computing the weighted sum of the kernel function with respect to the support vectors \( x_i \) (remember that only the support vectors have non-zero dual variables \( \alpha_i \) )
Kernel Methods [6]

- How do we compute the dual variables $\alpha_i$ in the implicit space?
  - Very simple: we use the same optimization problem as before, except for now we replace the dot product $\varphi(x_i)^T\varphi(x_j)$ by the kernel $K(x_i, x_j)$

- The Lagrangian dual problem for the non-linear SVM is simply

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i^T, x_j)$$

- subject to the constraints

$$\left\{ \begin{array}{l}
\sum_{i=1}^{N} \alpha_i y_i = 0 \\
0 \leq \alpha_i \leq C \quad i = 1 \ldots N
\end{array} \right.$$
Kernel Methods [6]

Kernel Functions

- Polynomial kernels
  \[ K(x, x') = (x^T x' + 1)^p \]
  - The degree of the polynomial is a user-specified parameter

- Radial basis function kernels
  \[ K(x, x') = \exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right) \]
  - The width \(\sigma\) is a user-specified parameter, but the number of radial basis functions and their centers are determined automatically by the number of support vectors and their values

- Two-layer perceptron
  \[ K(x, x') = \tanh(\beta_0 x^T x' + \beta_1) \]
  - The number of hidden neurons and their weight vectors are determined automatically by the number of support vectors and their values, respectively. The hidden-to-output weights are the Lagrange multipliers \(\alpha_i\)
  - However, this kernel will only meet Mercer’s condition for certain values of \(\beta_0\) and \(\beta_1\)
Architecture of an SVM [6]

- This figure below illustrates the operation of the SVM during recall in the form of a neural network architecture.
Case Study: XOR [6]

- Decision function defined by the SVM
  - Notice that the decision boundaries are non-linear in the original space $\mathbb{R}^2$, but linear in the implicit space $\mathbb{R}^6$
References

1. Frank Masci, “An Introduction to Principal Component Analysis,”
   http://web.ipac.caltech.edu/staff/fmasci/home/statistics.refs/PrincipalComponentAnalysis.pdf
Classifier-Based Object Recognition
Two Phases of Object Recognition

1. Hypothesis Generation (HG) Phase: it should include all possibilities and it is expected to be relatively lighter to implement.

2. Hypothesis Verification (HV) Phase: it should be able to distinguish the real objects and it is expected to be heavier to implement ➔ Based on classifier.

HV consists of feature and classification.

- Features: PCA, ICA, Local Receptive Field (LRF), Histogram Of Gradient (HOG), Gabor Filter Bank (GFB)
- Classifier: LDA, Neural Network, Support Vector Machine (SVM)
GFB−SVM−Based Pedestrian Recognition [1][2][3]
**Gabor Filter**

Gabor filter is the product of 2D Gaussian function and 2D sine wave. It can detect the orientation-specific, frequency-specific components of image at a specific spatial location.

Gabor filter in spatial domain is defined like Eq. (1)

\[
g(x, y) = \left( \frac{1}{2\pi\sigma_x\sigma_y} \right) \exp \left[ -\frac{1}{2} \left( \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) + 2\pi jWx \right]
\]  

(1)

Its definition in frequency domain is like Eq. (2)

\[
G(u, v) = \exp \left[ -\frac{1}{2} \left( \frac{(u - W)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2} \right) \right]
\]  

(2)

Gaussian wavelet can be defined by rotating the basic form.

\[
g_r(x, y) = a^{-r} \left( \frac{1}{2\pi\sigma_x\sigma_y} \right) \exp \left[ -\frac{1}{2} \left( \frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2} \right) + 2\pi jU'x' \right]
\]

\[
\begin{align*}
x' &= x \cos \left( \frac{k\pi}{K} \right) + y \sin \left( \frac{k\pi}{K} \right) \\
y' &= -x \sin \left( \frac{k\pi}{K} \right) + y \cos \left( \frac{k\pi}{K} \right)
\end{align*}
\]
Garbor Filter Bank

[Spatial Domain]

[Frequency Domain]

Real part of first filter
Support Vector Machine [2]

![Graph showing ROC comparison](image)

**Table 1** Complexity comparison: feature dimension and support vector number

<table>
<thead>
<tr>
<th>Feature dimension</th>
<th>LQR–quadratic SVM [2, 21]</th>
<th>GFB–RBF SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support vector no.</td>
<td>5,160</td>
<td>2,379</td>
</tr>
<tr>
<td></td>
<td>L792</td>
<td>648</td>
</tr>
</tbody>
</table>

Fig. 3 ROC comparison: the *dotted line* is ROC of LRF-quadratic SVM shown in Fig. 5d of [1] and the *solid line* is ROC of GFB–RBF SVM ($C = 461095126710589.4$ and $\sigma = 0.00017627$)
Support Vector Machine [2]

Histogram of Oriented Gradients (HOG) are feature descriptors used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

## Support Vector Number Reduction (SVNR)

### Taxonomy of SVNR

<table>
<thead>
<tr>
<th>Method</th>
<th>Target</th>
<th>Performance</th>
<th>Representative</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customized Optimization</td>
<td>- SV number cost [43][44][45] &lt;br&gt; - SV number constraint [46][47]</td>
<td>- Cluster inner sample, SVM via clustering [48][49][50][51] &lt;br&gt; - Random sampling, RSV [52][53][54]</td>
<td>- Nearest neighbor-based [55][56][57][58][59][60][61] &lt;br&gt; - Cluster outer sample [62][63][30]</td>
<td></td>
</tr>
<tr>
<td>Learning Sample Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post-pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduced Set Selection</td>
<td>- Approximation error [65][66][32][67][68][69][25][33] &lt;br&gt; - SV number cost [29] &lt;br&gt; - Noisy SV [70]</td>
<td>- Linearly dependent SV: [71][72][73][74] &lt;br&gt; - Cluster center [75] &lt;br&gt; - SVR [76]</td>
<td>- Classification function-based [85][86]</td>
<td></td>
</tr>
<tr>
<td>Reduced Set Construction</td>
<td>- Approximation error [27][28][64][39][77][78]</td>
<td>- Cluster center [79][80][81][82][83] &lt;br&gt; - KPCA [84]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning Sample Selection</td>
<td>- Approximation error [89][90] &lt;br&gt; - Noisy SV [91]</td>
<td></td>
<td>- Classification function-based [87][31][88]</td>
<td></td>
</tr>
</tbody>
</table>

---


E-mail: hogijung@hanyang.ac.kr  
http://web.yonsei.ac.kr/hgjung
Support Vector Number Reduction (SVNR)

In particular, this paper confirms that the support vector number of a pedestrian classifier using a histogram of oriented gradient (HOG)-based feature and radial basis function (RBF) kernel-based SVM can be reduced by more than 99.5% without any accuracy degradation using iterative preimage addition which can be downloaded from the Internet.

![Graphs showing accuracy vs. SV remaining ratio](image)

**Fig. 2.** Experimental results of HOG-based pedestrian classifier. (a) Accuracy with respect to SV remaining ratio 0-100%. (b) Accuracy with respect to SV remaining ratio 0-3%.


E-mail: hogijung@hanyang.ac.kr
http://web.yonsei.ac.kr/hgjung
References


Viola Jones Method: AdaBoost-based Face Detection
Face Detector by AdaBoost [4]
AdaBoost [4]

- Given a set of weak classifiers
  
  originally: \( h_j(x) \in \{+1, -1\} \)
  
  - None much better than random

- Iteratively combine classifiers
  
  - Form a linear combination

\[
C(x) = \theta \left( \sum_t h_t(x) + b \right)
\]

  - Training error converges to 0 quickly
  - Test error is related to training margin

E-mail: hogijung@hanyang.ac.kr
http://web.yonsei.ac.kr/hgjung
Face Detector by AdaBoost [4]

Weak Classifier 1

Weights Increased

Weak Classifier 2

Weak classifier 3

Final classifier is linear combination of weak classifiers
Haar–like Feature [7]

The simple features used are reminiscent of Haar basis functions which have been used by Papageorgiou et al. (1998).

Three kinds of features: two-rectangle feature, three-rectangle feature, and four-rectangle feature

Given that the base resolution of the detector is 24x24, the exhaustive set of rectangle feature is quite large, 160,000.
Rectangular features can be computed very rapidly using an intermediate representation for the image which we call the integral image.

The integral image at location $x, y$ contains the sum of the pixels above and to the left of $x, y$, inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image (see Fig. 2). Using the following pair of recurrences:

1. $$s(x, y) = s(x, y - 1) + i(x, y)$$
2. $$ii(x, y) = ii(x - 1, y) + s(x, y)$$

(where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$, and $ii(-1, y) = 0$) the integral image can be computed in one pass over the original image.

*Figure 2.* The value of the integral image at point $(x, y)$ is the sum of all the pixels above and to the left.
Haar-like Feature: Integral Image [7]

Using the integral image any rectangular sum can be computed in four array references (see Fig. 3).

![Figure 3](image)

*Figure 3. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4 + 1 − (2 + 3).*

Our hypothesis, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features.
Feature Selection [4]

• For each round of boosting:
  – Evaluate each rectangle filter on each example
  – Sort examples by filter values
  – Select best threshold for each filter (min $Z$)
  – Select best filter/threshold (= Feature)
  – Reweight examples

• $M$ filters, $T$ thresholds, $N$ examples, $L$ learning time
  – $O(MTL(MN))$ Naïve Wrapper Method
  – $O(MN)$ Adaboost feature selector
Cascasted Classifier [4]

- Given a nested set of classifier hypothesis classes

- Computational Risk Minimization

(Image of cascade classifier diagram with three classifiers and decision tree)
Cascaded Classifier [4]

• A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
• A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) – using data from previous stage.
• A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)
References

1. Frank Masci, “An Introduction to Principal Component Analysis,”
   http://web.ipac.caltech.edu/staff/fmasci/home/statistics.refs/PrincipalComponentAnalysis.pdf
Kalman Filter–based Tracking, A Posteriori Probability
2D Kalman Filtering [1]

- State vector: Light blob 무게 중심의 x 좌표, y 좌표, x축 속도, y축 속도

\[
\begin{bmatrix}
    x \\
    y \\
    v_x \\
    v_y
\end{bmatrix} = \begin{bmatrix}
    x(k+1) \\
    y(k+1) \\
    v_x(k+1) \\
    v_y(k+1)
\end{bmatrix} = \begin{bmatrix}
    x(k) + v_x(k) \\
    y(k) + v_y(k) \\
    v_x(k) \\
    v_y(k)
\end{bmatrix} = \begin{bmatrix}
    1 & 0 & 1 & 0 \\
    0 & 1 & 0 & 1 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x(k) \\
    y(k) \\
    v_x(k) \\
    v_y(k)
\end{bmatrix}
\]

- 2차원 평면에 대한 LTI(Linear Time-Invariant) model을 이용한 Kalman filter

State transition matrix
\[
A = \begin{bmatrix}
    1 & 0 & 1 & 0 \\
    0 & 1 & 0 & 1 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

Measurement matrix
\[
H = \begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0
\end{bmatrix}
\]

State covariance
\[
Q = \begin{bmatrix}
    10 & 0 & 0 & 0 \\
    0 & 10 & 0 & 0 \\
    0 & 0 & 10 & 0 \\
    0 & 0 & 0 & 10
\end{bmatrix}
\]

Measurement covariance
\[
R = \begin{bmatrix}
    5 & 0 & 0 & 0 \\
    0 & 5 & 0 & 0
\end{bmatrix}
\]

System dynamics 정의
System noise level 정의

E-mail: hogijung@hanyang.ac.kr
http://web.yonsei.ac.kr/hgjung
2D Kalman Filtering [1]

1) Prediction 단계
\[ \hat{x}^{-}_k = A \cdot \hat{x}_{k-1} \]
\[ P^{-}_k = A \cdot P_{k-1} \cdot A^T + Q \]

2) Matching 단계
- Predicted position \( \hat{x}^{-}_k \) 에서 가장 가까운 측정치 \( z_k \) 검색
- 그 object 까지의 거리가 \( P_k \) 를 활용한 임계치보다 작으면 대응 성립 \( \rightarrow \) correction 단계로 대응 없으면 \( \rightarrow \) 소멸

3) Correction 단계
\[ K_k = P^{-}_k \cdot H^T (H \cdot P^{-}_k \cdot H^T + R)^{-1} \]
\[ \hat{x}_k = \hat{x}^{-}_k + K_k \left( z_k - H \cdot \hat{x}^{-}_k \right) \]
\[ \hat{x}_k = (I - K_k H) \hat{x}^{-}_k + K_k z_k \]
\[ P_k = P^{-}_k - K_k \cdot H \cdot P^{-}_k \]

4) 생성 단계: matching 되지 않은 \( z_k \)에 대해 track 생성
C개의 class가 있는 classification을 N회 tracking된 object에 적용한 결과가 O이다.
즉, 어떤 object의 class c는 1 ≤ c ≤ C
특정 시간 n의 어떤 object에 대한 classifier의 출력 o(n)은 1 ≤ o_n ≤ C
N회 tracking된 어떤 object에 대한 classifier 출력 O는 O = [o_1, o_2, ..., o_N]

N회 tracking된 어떤 object에 classifier를 적용하여 classifier 출력 O를 얻은 후, 그 object의 class를 알아내는 것은

\[
\arg \max_{c} P(c|O) \quad \text{Maximum A Posteriori Probability}
\]

Class c인 샘플들에 대해서 classifier를 적용하여 얻은 결과는 \( P(O|c) \)이다.
⇒ 미리 측정할 수 있는 값이다.
Bayes’ Rule을 이용하면,

\[
P(c \mid O) = \frac{P(O \mid c)P(c)}{P(O)} = \frac{P(O \mid c)P(c)}{\sum_{c=1}^{C} P(O \mid c)P(c)}
\]

이때, classifier의 출력이 서로 independent하다면,

\[
P(O \mid c) = P(o_1 \mid c)P(o_2 \mid c) \cdots P(o_{n-1} \mid c)P(o_n \mid c) = \{P(o_1 \mid c)P(o_2 \mid c) \cdots P(o_{n-1} \mid c)\}P(o_n \mid c)
\]

누적해서 곱해서 구할 수 있다.
**Classification of Tracked Object [2]**

classifier output sequence $X$

$$P(X|\text{Ped}) = \binom{N}{k}P(T|\text{Ped})^k \cdot P(F|\text{Ped})^{N-k}$$

$$= \frac{N!}{k!(N-k)!}P(T|\text{Ped})^k \cdot P(F|\text{Ped})^{N-k}$$

$$P(X) = P(\text{Ped}) \binom{N}{k}P(T|\text{Ped})^k \cdot P(F|\text{Ped})^{N-k}$$

$$+ P(\text{Non}) \binom{N}{k}P(T|\text{Non})^k \cdot P(F|\text{Non})^{N-k}$$

$P(\text{Ped}) = P(\text{Non}) = 0.5$

$$P(\text{Ped}|X) = \frac{\binom{N}{k}P(T|\text{Ped})^k \cdot P(F|\text{Ped})^{N-k}}{\binom{N}{k}P(T|\text{Ped})^k \cdot P(F|\text{Ped})^{N-k} + \binom{N}{k}P(T|\text{Non})^k \cdot P(F|\text{Non})^{N-k}}$$
Classification of Tracked Object [2]
Classification of Tracked Object [2]
References