Support Vector Machine: Face Detection in Still Gray Images

Written by:
- Bernd Heisele
- Tomaso Poggio
- Massimiliano Pontil

Presented by: Golan Maimon
Date: 30/03/05

Outline
- SVM
  - What is a good decision boundary?
  - Linear classifier
  - Non-linear decision by Kernel
- Face Detection
  - Extracting image features.
  - Feature Reduction Using PCA & ILC
  - Selecting Features Using SVM
  - Training Data
  - Component-based face detection

Feature extraction and Classification
- Feature extractor whose purpose is to reduce the data by measuring certain “properties”. Receive example $y$ and return $F(y)$.
- Classifier $C$ receive extract example $F(y)$ and decides which class to classify.

Introduction SVM & Kernel
Two class problem: Linear Separable Case

Many decision boundaries can separate these two classes.

Which one should we choose?

Linear classifier

- Function $Z$ classified which class the data belongs to.

$$z_i = \begin{cases} +1 & \text{if } y_i \text{ in class } C_1 \\ -1 & \text{in class } C_2 \end{cases}$$

Linear discriminant function:

$$g(y) = a^T y + b$$

Linear classifier with margin

- The decision boundary should be as far away from the data of both classes as possible.
- We should find the hyperplane that maximizes the margin $m$ with: $z_i g(y_i) = z_i (a^T y + b) \geq m$ for all $y_i \in y$

Vectors $y_i$ with $z_i g(y_i) = m$ are the support vectors

Maximal Margin Classifier

- Invariance: assume that the weight vector $a$ is normalized ($\|a\| = 1$)
  $$(a, b) \leftarrow (\lambda a, \lambda b), m \leftarrow \lambda m$$
  does not change the problem

Condition:

$$z_i = \begin{cases} +1 & a^T y_i + b \geq m \\ -1 & a^T y_i + b \leq -m \end{cases} \quad \forall i$$

Objective: maximize margin $m$ s.t. joint condition $z_i (a^T y_i + b) \geq m$ is met.

Learning problem: Find $a$ with $\|a\| = 1$, such that the margin $m$ is maximized.

$$\text{maximize } m$$
$$\text{subject to } \forall y_i \in y : z_i (a^T y_i + b) \geq m$$
Margin

What is margin \( m \)?

- Consider two points \( P_1 \) and \( P_2 \) of class 1,2 which located on both sides of the margin boundaries.

\[
2m = \frac{a^T}{\|a\|} (p_1 - p_2) = a^T P_1 + b = a^T P_2 + b = \frac{a^T p_1 - a^T p_2}{\|a\|} = \frac{1-b}{\|a\|} = 2
\]

- Maximizing the margin corresponds to minimizing the norm \( a \) for margin \( m = 1 \).

SVM Lagrangian (1)

- Minimize \( \|a\| \) for a given margin \( m = 1 \)
  
  \[
  \text{minimize} \quad T(a) = \frac{1}{2} a^T a
  \]
  
  subject to \( z_i(a^T y_i + b) \geq 1 \)

- Generalized Lagrange Function:

\[
L(a, b, \alpha) = \frac{1}{2} a^T a - \sum_{i=1}^{\alpha} \alpha_i [z_i(a^T y_i + b) - 1]
\]

SVM Lagrangian (2)

- Extremality condition:

\[
\frac{\partial L(a, b, \alpha)}{\partial a} = a - \sum_{i=1}^{\alpha} \alpha_i z_i y_i = 0 \quad \Rightarrow \quad a = \sum_{i=1}^{\alpha} \alpha_i z_i y_i
\]

\[
\frac{\partial L(a, b, \alpha)}{\partial b} = - \sum_{i=1}^{\alpha} \alpha_i z_i = 0
\]

- Resubstituting:

\[
\frac{\partial L}{\partial \alpha} = 0, \quad \frac{\partial L}{\partial b} = 0 \quad \text{into the Lagrangian function}
\]

SVM Lagrangian (3)

- After substituting:

\[
L(a, b, \alpha) = \frac{1}{2} a^T a - \sum_{i=1}^{\alpha} \alpha_i [z_i(a^T y_i + b) - 1]
\]

\[
= \frac{1}{2} \sum_{i=1}^{\alpha} \sum_{j=1}^{\alpha} \alpha_i \alpha_j z_i z_j y_i y_j + \sum_{i=1}^{\alpha} \alpha_i
\]

\[
= \frac{1}{2} \sum_{i=1}^{\alpha} \sum_{j=1}^{\alpha} \alpha_i \alpha_j z_i z_j y_i y_j
\]
The Optimization Problem

The dual of the problem is:

\[ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} z_i z_j \alpha_i \alpha_j y_i y_j \]

Subject to \( \forall i \quad \alpha_i \geq 0 \) and \( \sum_{i=1}^{n} z_i \alpha_i = 0 \)

- Data \( y_i \) with non-zero \( \alpha_i \) are called **Support Vector**.
- The decision boundary is determined only by the SV.
- All the other \( \alpha_i \) are zero therefore there is no contribution of the other vectors.

Non-linear Decision Boundary (1)

- Key idea:
  - transform \( y_i \) to a higher dimensional space to “make life easier”.
  - Input space: the space \( y_i \) are in.
  - Feature space: the space of \( \phi(y_i) \) after transformation

Why transform?

- Linear operation in the feature space is equivalent to non-linear operation in input space
- The classification task can be “easier” with a proper transformation.

Classifier Complexity

- The classifier is:
  \[ \sum_{i=1}^{n} (\alpha_i z_i y_i) y + b \]

- \( M \) - number of SV, \( N \times N \) - image size
- Classifier Complexity \( (M) \times (N \times N) \)

Non-linear Decision Boundary (2)

- Possible problem of the transformation
  - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
  - Kernel tricks for efficient computation
  - Minimize \( ||\alpha||^2 \) can lead to a “good” classifier

The support vector with non-zero \( \alpha_i \)
**Kernel**

- Define the kernel function $K(x, y)$ as $k(x, y) = (1 + x_1y_1 + x_2y_2)^2$.

- Consider the following transformation:
  $$
  \phi \left( \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \right) = \begin{bmatrix} \sqrt{2} x_1 \\ \sqrt{2} y_1 \\ \sqrt{2} x_1^2 \\ \sqrt{2} y_1^2 \end{bmatrix}, \quad \phi \left( \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \right) = \begin{bmatrix} \sqrt{2} x_2 \\ \sqrt{2} y_2 \\ \sqrt{2} x_2^2 \\ \sqrt{2} y_2^2 \end{bmatrix}
  $$

- The inner product can be computed by $K$ without going through the map $\phi(.)$.

  $$
  \left\langle \phi \left( \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \right), \phi \left( \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \right) \right\rangle = \begin{bmatrix} 1 \\ \sqrt{2} x_1 y_1 \\ \sqrt{2} x_1^2 y_1^2 \\ \sqrt{2} y_1^2 \end{bmatrix} = (1 + x_1y_1 + x_2y_2)^2 = k(x, y)
  $$

**Kernel Trick**

- The relationship between the kernel function $K$ and the mapping $\phi(.)$ is $k(X, Y) = \langle \phi(X), \phi(Y) \rangle$ this is known as the kernel trick.

  - In practice, we specify $K$ instead of choosing $\phi(.)$.

  - We do not know, how the feature space looks like, we just need the kernel function as a measure of similarity.

- $K(x, y)$ needs to satisfy a technical condition (Mercer condition) in order for $\phi(.)$ to exist.

**Examples of Kernel Functions**

- Polynomial kernel with degree $d$
  $$
  K(x, y) = (x^T y + 1)^d
  $$

- Radial basis function kernel with width $\sigma$
  $$
  K(x, y) = \exp(-||x - y||^2/(2\sigma^2))
  $$

- Sigmoid with parameter $\kappa$ and $\theta$
  $$
  K(x, y) = \tanh(\kappa x^T y + \theta)
  $$

**Modification Due to Kernel Function**

- Change all inner products to kernel functions.

  Original

  Subject to $\forall i \quad C \geq \alpha_i \geq 0$ and $\sum_{i=1}^n z_i \alpha_i = 0$

  $$
  W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n z_i z_j \alpha_i \alpha_j y_i^T y_j
  $$

  With kernel function

  Subject to $\forall i \quad C \geq \alpha_i \geq 0$ and $\sum_{i=1}^n z_i \alpha_i = 0$

  $$
  W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n z_i z_j \alpha_i \alpha_j K(y_i, y_j)
  $$
Example (1)

- Suppose we have 5 data points 1 Dimension.
  - $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 5, y_5 = 6$, with 1, 2, 6 as class 1 and 4, 5 as class 2
  - $z_1 = 1, z_2 = 1, z_3 = -1, z_4 = -1, z_5 = 1$

- We use the polynomial kernel of degree 2.
  - $K(y_i, y_j) = (y_i y_j + 1)^2$
  - $C$ - parameter is set to 100.

- We find $\alpha_i, i = 1, 2, ..., 5$ by

\[
\max \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j z_i z_j (y_i y_j + 1)^2
\]
subject to $0 \leq \alpha_i \leq 100, \sum_{i=1}^{5} \alpha_i z_i = 0$

- By using a Quadratic Programming solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{y_2=2, y_4=5, y_5=6\}$

- The discriminant function is

\[
f(y) = 2.5 \cdot (2y + 1)^2 + 7.333 \cdot (-1)(5y + 1)^2 + 4.833 \cdot -1(6y + 1)^2 + b = 0.6667y^2 - 5.333y + b
\]

Example (2)

- We first find $\alpha_i, i = 1, 2, ..., 5$ by

\[
f(x) = \phi(x) = \phi(x) = \phi(x) = \phi(x) = \phi(x)
\]

- $C$ - parameter is set to 100.

- We use the polynomial kernel of degree 2.
  - $K(y_i, y_j) = (y_i y_j + 1)^2$

Example (3)

- $b$ is recovered by solving:
  - $f(2) = 1$ or $f(5) = 1$ or $f(6) = 1$, as $x_2, x_4, x_5$ lie on $z_i (x, \phi(x) + b) = 1$ and all give $b = 9$.
  - $f(y) = 0.6667y^2 - 5.333y + 9$

Summary: Steps for Classification

- Prepare the pattern matrix.
- Select the kernel function to use.
- Select the parameter of the kernel function and the value of $C$.
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter.
- Execute the training algorithm and obtain the $\alpha_i$.
- Unseen data (test set) can be classified using the $\alpha_i$ and the support vectors.
Strengths and Weaknesses of SVM

- **Strengths**
  - Training is relatively easy
  - Tradeoff between classifier complexity and error can be controlled explicitly

- **Weaknesses**
  - Need a “good” kernel function

Face Detection in still gray image

Extracting image features

- The goal of feature extraction is reduced the objects space with keeping the relevant variations for separating to classes.

Three preprocessing steps

- Remove pixels close to the boundary of the 19x19 images in order to eliminate the background.
- Best fit intensity plane was subtracted from the gray values to compensate for cast shadows.
- Histogram equalization was applied to remove variations in the image brightness and contrast.
Example - Histogram equalization

Gradients

- The gradients were computed from the histogram equalized 19x19 images, using 3 x 3 and x,y- Sobel filters.

Haar wavelets

- A features takes a scalar value by summing up the white region and subtracting the dark region.
- Three orientation tuned masks in two different scales were convoluted with the 19X19 image to compute the Haar wavelets.

Comparing gray, gray gradient & Haar wavelets (HW)(2)

- Gray, gray gradient and H.W features were rescaled to be in range between 0 and 1 before used for training an SVM with 2nd degree polynomial kernel.
- Training data consisted of:
  - 2429 face images.
  - 19932 non-face images.
- Test set consisted of:
  - 118 gray images.
  - 479 frontal images.
The images were rescaled 14 times by factor between 0.1 and 1.2 to detect images at different scales.
- A 19x19 window was shifted pixel by pixel over each image.
- About 57,000,000 windows were processed.

Result: For a fixed FP rate the detection rate for gray values was about 10% higher than for HW and about 20% higher than for gray gradients.

The Goal is:
- Improve the detection rate.
- Speed up the classification process by removing class irrelevant features.

Investigated two ways of feature reduction:
- Linear combination of features
- Selecting feature

Two techniques were evaluated, which generate new features sets by linearly combining the original features:
- Principal Component Analysis (PCA)
- Iterative Linear Classification (ILC)
Determines the most class discriminant, orthogonal features by iteratively training a linear classifier on the labeled training samples.

The algorithm consists of two steps:

1. Determine the direction for separating the two classes by training a linear classifier on the current training samples.
2. Generate a new sample set by projecting the samples into a subspace that is orthogonal to the direction calculated in (a) and continue with step (a).

Both techniques were applied to 283 gray value features.

Training data consisted of:
- 2429 face images.
- 4550 non-face images.

Test set consisted of:
- 479 face patterns.
- 23,570 non-face patterns.

An SVM with a 2nd degree polynomial kernel was trained on the reduced feature sets.
Feature Reduction ILC

Increasing the number of ILC features up to 10 did not improve the performance. This is because ILC does not generate uncorrelated features.

Selecting features – Using SVM (1)

- Used a technique for selecting class relevant based on the decision function $f(y)$ of an SVM.
  \[ f(y) = \sum \alpha_i z_i K(y_i, y) + b \]
- $Y_i$ - are the support vector.
- $\alpha_i$ - The Lagrange multipliers.
- $Z_i$ - the labels of the support vector (-1 or 1)

The transformation from the original feature space $F$ to $F^*$ by $\phi(y)$ is: $f(y) = a \cdot \phi(y) + b$

Selecting features – Using SVM (2)

For a 2nd degree polynomial kernel with:

\[ K(x, y) = (1 + x \cdot y)^2 \]

Transformed feature space $F^*$ with:

- dimension \( \frac{(N + 3)N}{2} \)
- $y^* = \phi(y) = (\sqrt{z_1}, \sqrt{z_2}, ..., \sqrt{z_i}, y_{1x_i}, y_{2x_i}, y_{3x_i}, ..., y_{ix_i})$

The contribution of a feature $y_i^*$ to the decision function $f(y) = a \cdot \phi(y) + b$ depends on $\alpha_i$.

\[ a = \sum \alpha_i z_i \phi(y_i) \]

Selecting features – Using SVM (3)

- Two ways to order the features by ranking
  1. A straightforward by decreasing $|\alpha_i|$.
  2. **Weighted** By SV for account different distributions of the features in the TD.

- Ordered by decreasing $|a| \sum z_i y_{ij} |$  where $y_{ij}$ denotes the n-th component of SV $i$ in feature space $F^*$.

- Both ways of feature ranking were applied to an SVM with 2nd-degree polynomial kernel, trained on 20 PCA features corresponding to 230 features in $F^*$.
Selecting features – Using SVM (2)

- In a first evaluation of the rankings they calculated for all M support vector: \( \frac{1}{m} \sum_{i=1}^{m} |f(y) - f_s(x_i)| \)
- \( f_s(x) \) - is the decision function using the S first features according to the ranking.

Partial Sum for Support Vectors

The results show that ranking by the weighted components of W lead to faster convergence of the error towards 0.

Feature Selection

For 100 features on the TS was about the same as for the complete feature set.

Different Kernel Function

The SVM with Gaussian kernel (\( \sigma^2 = 5 \)) was slightly better but required about 1.5 times more Support Vectors (738 versus 458) than the polynomial SVM.
The detection performance slightly increases with $C$ until $C=1$. For $C \geq 1$ the error rate on the training data was 0 and the decision boundary did not change any more.

An alternative way to generate artificial images for training the classifier, can be generated by:
- Rendering 3-D head models.
- Modified the pose and the illumination of the head.
- Morphed between different head models.
- Heads were rotated between $-15^\circ$ and $15^\circ$.
- Heads were rotated between $-8^\circ$ and $8^\circ$ in the image plane.
- The position of the light varied between $-30^\circ$ and $30^\circ$.
- Elevation varied between $30^\circ$ and $60^\circ$.
- Overall, they generated about 5000 face images.

The image variations captured in the synthetic data do not cover the variations present in real face images.

Classifiers trained on real and synthetic faces.
Non-face patterns are abundant and can be automatically extracted from images that do not contain faces.

- Target – increase the false positive (FP).
- An SVM trained on:
  - 19,932 randomly selected non-face patterns.
  - 7,065 non-face pattern.
- Determined in three bootstrapping iterations.

Bootstrapping
- Bootstrapped – FP rate was about 1 FP per image.
- Non-bootstrapped – FP rate was about 3 times higher.

Component-based face detection (1)
- Matching with a single template
  - The schematic template of a frontal face is shown in (a).
  - Slight rotations of the face in the image plane (b) and in depth (c)
- Lead to considerable discrepancies between template and face.

Component-based face detection (2)
- Matching with a set of component templates.
  - The schematic component templates for a frontal face are shown in (a).
  - Shifting the component templates can compensate for slight rotations of the face in the image plane (b) and in depth (c).
The component-based approach tries to avoid this problem by independently detecting parts of the face.

Two level component-based classifier is shown:
- **First level**: Component classifiers independently detect the eyes, the nose and the mouth.
- **Second level**: Classifiers perform the final face detecting by combining the results of the component classifiers. Search regions are fed into the geometrical configuration classifier.

The component classifiers were SVM with 2nd degree polynomial kernels and the geometrical configuration classifier was a linear SVM.

The four component system performs worse than the whole face classifier.

Added the whole face as a fifth component.

They performed tests on synthetic face 3D models consisted of two groups:
- 4574 faces rotated in the image plane.
- 15,865 faces rotated in depth.

The best performance was achieved by the five component.
Synthetic faces rotated in the image plane

Synthetic faces rotated in depth

Conclusions

- Gray value are better input features for a face detector than Haar wavelets and gradients value.

- By combining PCA with SVM based feature selection we sped-up the detection system by two orders of magnitude without loss in classification performance.

- Bootstrapping the classifier with non-face patterns increased the detection rate by more than 5%.

- A component-based face detector is more robust against face rotations than comparable whole face detector.