Face Detection in Still Gray Images

Heisele, Poggio, Pontil - 2000

presented by David Ross and Karolina Büchner

Overview

- Support Vector Machines: a brief tutorial
  - Linear SVM's, separable data
  - Non-separable data
  - Non-linear SVM's - kernel trick
  - Choice of kernel
- Face Detection in Still Gray Images
  - Preprocessing (Extracting Features)
  - Feature Reduction
  - Whole Face Detection
  - Component-based face detection
  - Related component-based detection

Linear SVM's, Separable Data

- given labeled data
  $\{\mathbf{x}_i, y_i\}, i = 1, \ldots, l$
  $\mathbf{x}_i \in \mathbb{R}^d, \quad y_i \in -1, +1$

- want a hyperplane that separates positive and negative examples

- equation of plane: $\mathbf{w} \cdot \mathbf{x} + b = 0$

- classify new instance according to which side of the plane it's on

$$\mathrm{sign}(\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$

- data correctly classified if

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0 \quad \forall i$$

Linear SVM's, continued

- further constrain the hyperplane so it lies halfway between the nearest *+ve* examples and the nearest *-ve* examples

- choose scale of $\mathbf{w}$ and b so that

  $\mathbf{w} \cdot \mathbf{x}_+ + b = +1$
  for nearest *+ve* examples $\mathbf{x}_+$

  $\mathbf{w} \cdot \mathbf{x}_- + b = -1$
  for nearest *-ve* examples $\mathbf{x}_-$

- define $margin = d_+ + d_- = \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$

- want to maximize the margin, hence we will minimize $\|\mathbf{w}\|$

Quadratic Program

- minimize $\frac{1}{2}||\mathbf{w}||^2$
- subject to $y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \forall i$
- using Lagrange multipliers α_i we get the primal objective

$$L_P = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{\ell} \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1)$$

$\alpha_i \geq 0$

Quadratic Program, continued

- taking derivatives w.r.t. b and $\mathbf{w}$ respectively gives

$$\sum_i \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i}$$

- substituting into primal objective L_P gives the dual

$$L_D = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{x_i} \cdot \mathbf{x_j})$$

which we maximize subject to

$$\alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0$$

SVM Training

- maximize L_D subject to constraints
- $\forall i$, Lagrange Multiplier α_i corresponds to training point $\mathbf{x_i}$
- $\alpha_i > 0 \Rightarrow \mathbf{x}_i$ is a "support vector" and lies on one of H_1 or H_2

  $\alpha_i = 0 \Rightarrow \mathbf{x}_i$ lies on or beyond either H_1 or H_2
- support vectors specify location of the decision boundary (since $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$)

Non-Separable Data - Soft Margin

- suppose data cannot be separated by a (hyper)plane, due to noise and outliers
- allow positive points to be on negative side of plane H_1 (& vice versa), but penalize for this
- relax constraints $\forall i$:

$$y_i(\mathbf{x} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- objective to minimize becomes

$$\frac{||\mathbf{w}||^2}{2} + C\left(\sum_i \xi_i\right)$$

- C - how much to penalize errors
- dual same, but now has additional restriction

$$0 \leq \alpha_i \leq C \quad \forall i$$

Non-Linear SVM's

- suppose decision function is a non-linear function of the data

- idea: first map data to a high-dimensional space $\mathcal{H}$ via

$$\Phi : \mathbb{R}^d \to \mathcal{H}$$

  then separate data with a hyperplane in $\mathcal{H}$

- note: in training, data appears only in dot products $\mathbf{x}_i \cdot \mathbf{x}_j$

- training algorithm now depends only on dot products of data in $\mathcal{H}$

$$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Kernel Trick

- if there is a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

  then replace $\mathbf{x}_i \cdot \mathbf{x}_j$ in our original algorithm with $K(\mathbf{x}_i, \mathbf{x}_j)$

- decision function becomes

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{\ell} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b \\ &= \sum_{i=1}^{\ell} \alpha_i y_i \, K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

  (actually: only sum over SV's, i.e. $\mathbf{x}_i$ where $\alpha_i > 0$)

- now, given K, we don't need to know what Φ is explicitly

Choice of Kernel

- kernel must define a dot product in some space
- typical examples:
  - $K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$
  - $K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2})$
  - $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa(\mathbf{x_i} \cdot \mathbf{x_j}) - \delta)$

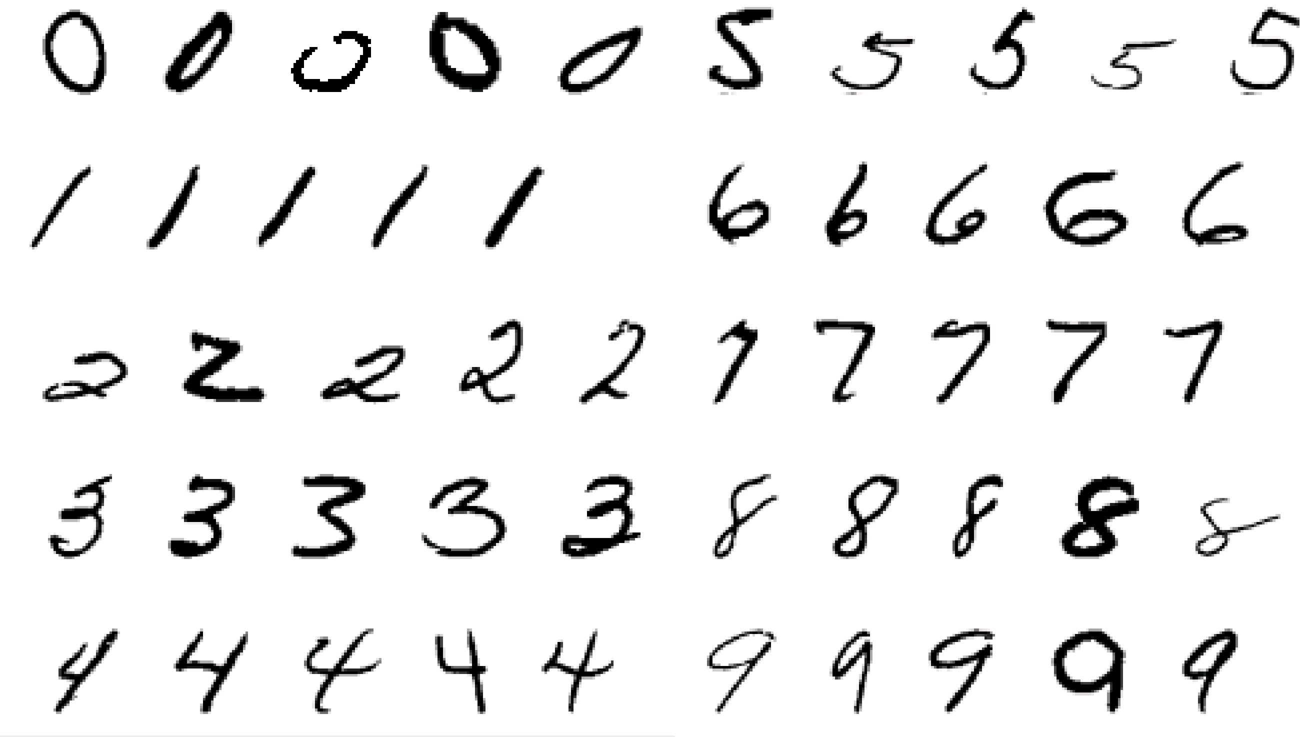
. . . which kernel to choose?!

Does it matter much?

- experiments suggest perhaps *NOT*
- task: hand-written digit recognition
- SVM's using different kernels
  $\Rightarrow$ many common support vectors (approx. 80%)

	Poly	RBF	NN	Common
total # of SV's	1677	1727	1611	1377
% of common SV's	82	80	85	100

Hand-Written Digit Data



Face Detection: Motivation

1. interesting applications

2. most faces very visually similar
   $\Rightarrow$ easier detection task

- key issue: pose invariance
- 2 methods:
  1. whole-face detection with single classifier
  2. component-based face detection with multiple classifiers $\Rightarrow$ better w.r.t. pose invariance

Preprocessing

extract image features such that

- within-class variations reduced
- between-class variations kept

1. trim image to eliminate background

2. reduce effect of cast shadows (subtract *"best-fit intensity plane"*)

3. histogram normalization

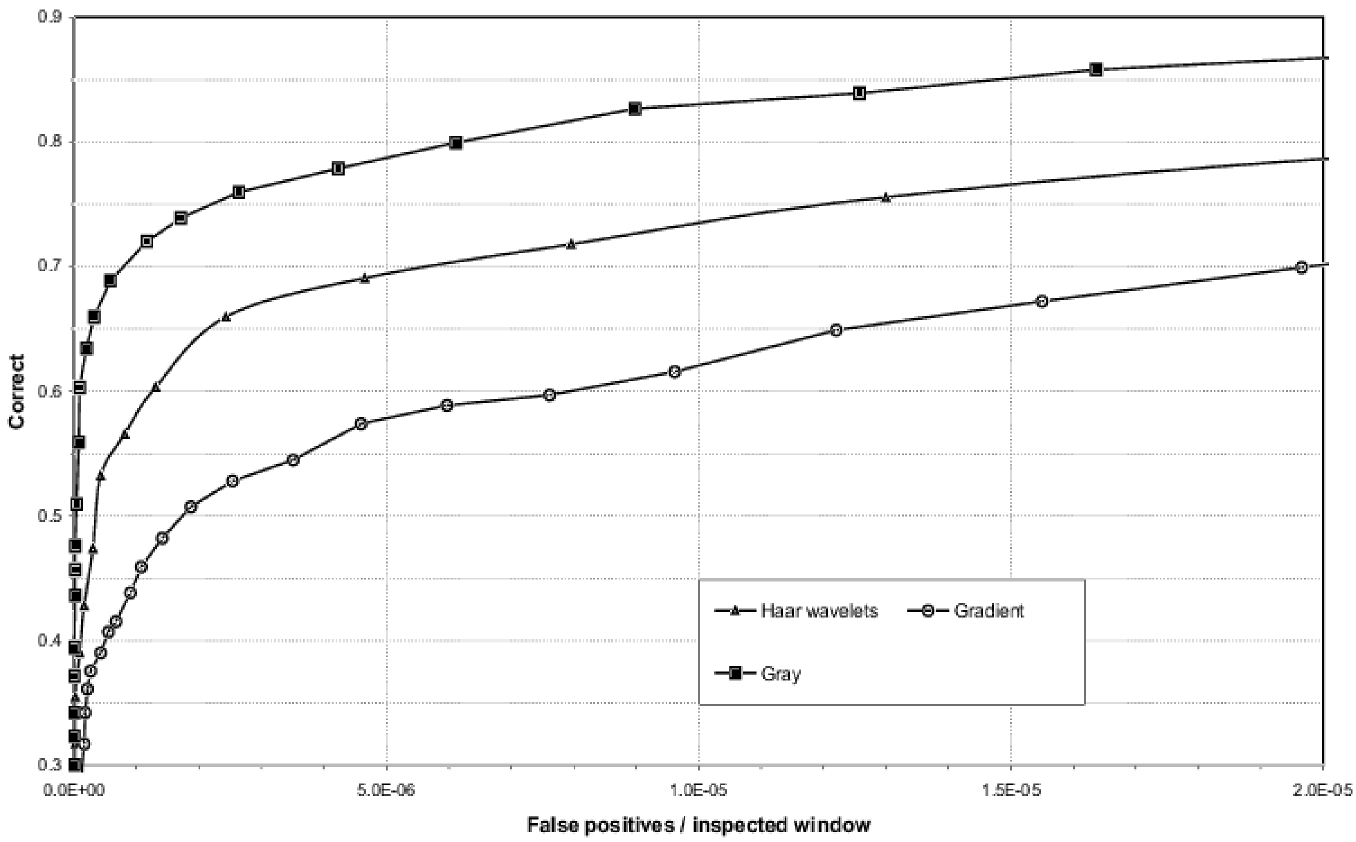
other techniques: gradients, Haar wavelets

Comparison

- training images: 2429 faces, 19932 non-faces
- pixel values scaled to [0,1]
- 19×19 window shifted pixel-by-pixel over each image
  $\rightarrow$ 57 million windows
- SVM with 2^{nd} degree polynomial kernel
- ROC curve generated by varying b (SVM threshold)

Features

(Training: 2,429 faces, 19,932 non-faces, Test: 118 images, 479 faces, 56,774,966 windows)



Feature Reduction

- improve detection rate & speed up classification by removing irrelevant features
- generating new features which are linear combinations of original ones
  - Principle Component Analysis
  - Iterative Linear Classification
    iteratively train a linear classifier (SVM)
    1. determine direction $\approx$ feature
    2. *"subtract out effects of this direction"*
    3. repeat ...

    new N-dimensional feature space spanned by first N directions

Comparison

- training data: 2429 faces, 4550 non-faces (most similar to faces)
- SVM with 2^{nd} degree polynomial kernel
  trained on reduced feature sets
- 10 ILC features highly correlated
  $\Rightarrow$ no improvement
- 20 PCA features
  $\Rightarrow$ classification performance improves and is sped up by a factor of 196 (vs. all features)

Selecting features in *feature space*

select features in feature space $\mathcal{F}^\star$ that contribute most to decision function

$$f(x) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$$

1. order features, i.e. components of $\mathbf{x}^\star = \Phi(\mathbf{x})$, by decreasing $|\mathbf{w_n}|$

2. alternatively, order by $|\mathbf{w_n}(\sum_i y_i \mathbf{x}^\star_{\mathbf{i},\mathbf{n}})|$

use only the first of the ordered features when determining f(x)

Comparison

$$\frac{1}{M}\sum_i |f(x_i) - f_S(x_i)|$$

$\approx$ average difference between original f and new f_S when classifying support vectors

input

$\downarrow_{PCA}$

20 components in input space

$\downarrow_{\Phi}$

230 features in $\mathcal{F}^\star$

$\downarrow$feature selection

50, 100, 150 features in $\mathcal{F}^\star$

This brings up some issues . . .

- the authors select a subset of the features $\mathbf{x}^\star = \Phi(\mathbf{x})$
- kernel trick no longer applies for testing
- have to know Φ explicitly; Φ can't have infinite dimensionality
- **does** speed up testing

Parametrization of SVM's

- choice of kernel, kernel parameters, C
- Gaussian kernel ($\sigma^2 = 5$) best, but required 1.5 times more support vectors than polynomial kernel ($n = 2$)
- $C \in [0.1, 1]$ showed slight improvement
- $C \geq 1 \Rightarrow$
  training error = 0; no change in decision boundary

Training Data

positive data:

- real images
- rendered images (7 heads, differing illumination, position)

+: control over lighting, orientation
-: not enough variation in models

negative data:

- non-face set augmented with false positives using boot-strapping
  $\Rightarrow$ boot-strapping improves performance by 5%

Whole Face Detection

- uses heuristics to suppress multiple detections

System	Subset of test set 1 23 images, 155 faces		Test set 1 130 images, 507 faces	
	Det. Rate	FPs	Det. Rate	FPs
[Sung 96] Neural Network	84.6%	13	N/A	N/A
[Osuna 98] SVM	74.2%	20	N/A	N/A
[Rowley et al. 98] Single neural network	N/A	N/A	90.9%	738
[Rowley et al. 98] Multiple neural networks	84.5%	8	84.4%	79
[Schneiderman & Kanade 98][3] Naïve Bayes	91.1%	12	90.5%	33
[Yang et al. 99][4] SNoW, multi-scale	94.1%	3	94.8%	78
Our system[5]	84.7% 90.4%	11 26	85.6% 89.9%	9 75

Table 1: Comparison between face detection systems.

- problem: very sensitive to small rotations . . .

Component-based Face Detection

- avoid problem of pose-sensitivity by independently detecting eyes, nose, mouth
- 2-level classifier:

  1. detect eyes, nose, mouth
     using 2^{nd} degree polynomial K SVM's
     trained on manually selected facial components

  2. geometrical configuration classifier
     combine outputs of component detectors to see if face template is matched

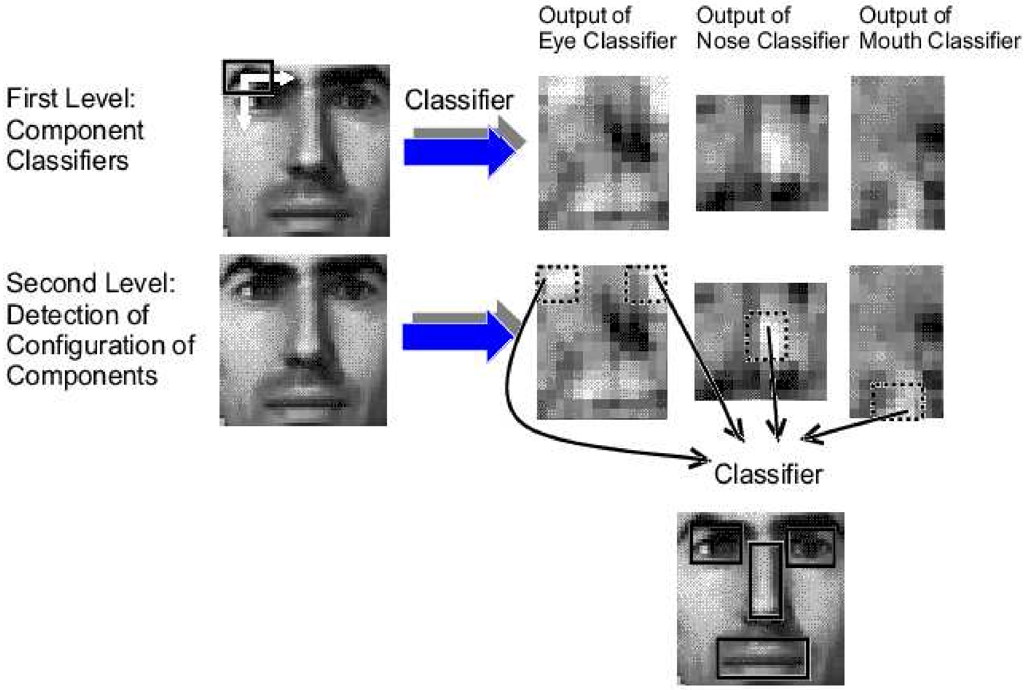


Figure 16: System overview of the component-based classifier. On the first level, windows of the size of the components (solid lined boxes) are shifted over the face image and classified by the component classifiers. On the second level, the maximum outputs of the component classifiers within predefined search regions (dotted lined boxes) are fed into the geometrical configuration classifier.

Results

- performance worse on frontal views than whole-face detector
- however, superior performance on faces rotated in depth and in image plane
- drawback: components were chosen manually
  $\Rightarrow$ lots of work to make training data
  $\Rightarrow$ are these really the best components to use?
- augmented with whole face as “component”

Determining Components

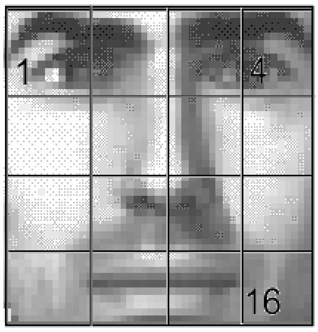


Figure 20: Partitioning the face pattern into 16 non-overlapping components.

- arbitrarily define a uniform grid of components
- during training, 2^{nd} level detector determines which components are useful

Figure 22: Automatically generated components.

- alternatively, manually select component centres, enlarge components based on estimate of SVM performance error

$$\rho = R^2 \mathbf{w}^2 = \frac{R^2}{margin^2}$$

R = radius of smallest sphere in $\mathcal{F}^\star$ containing the data

A Similar Component-Based Detector

"Example-Based Object Detection in Images by Components" (Mohan, Papageorgiou, Poggio - 2001)

- allows for variations in limb positions
- allows detection of partially occluded objects
- learn component detectors (SVM's)
- learn a geometric classifier
  $\Rightarrow$ input is location and scale of component detector results
- results much better than whole-body detector

Discussion

- paper presents a “grab-bag” of techniques
- addition of whole-face as “component” in component-based detector
- time/space complexity
  - quadratic programming: naive approach takes $O(n^3)$, $n =$ number of training points
  - training involves millions of image windows
  - number of support vectors used = ?