Visual Recognition: Instances and Categories

Raquel Urtasun

TTI Chicago

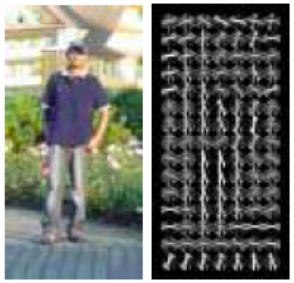
Jan 24, 2012

Which detectors?

Window-based

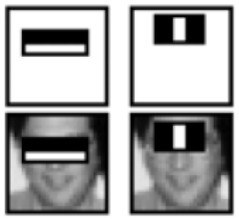

NN + scene Gist classification

e.g., Hays & Efros



SVM + person detection

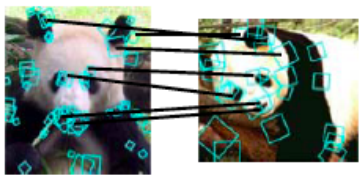
e.g., Dalal & Triggs



Boosting + face detection

Viola & Jones

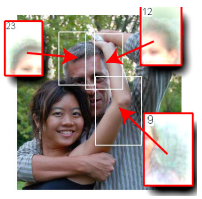
Part-based



BOW, pyramids
e.g., [Grauman et al.]

ISM: voting
e.g., [Leibe & Shiele]

deformable parts
e.g., [Felzenszwalb et al.]



poselets
[Bourdev et al.]

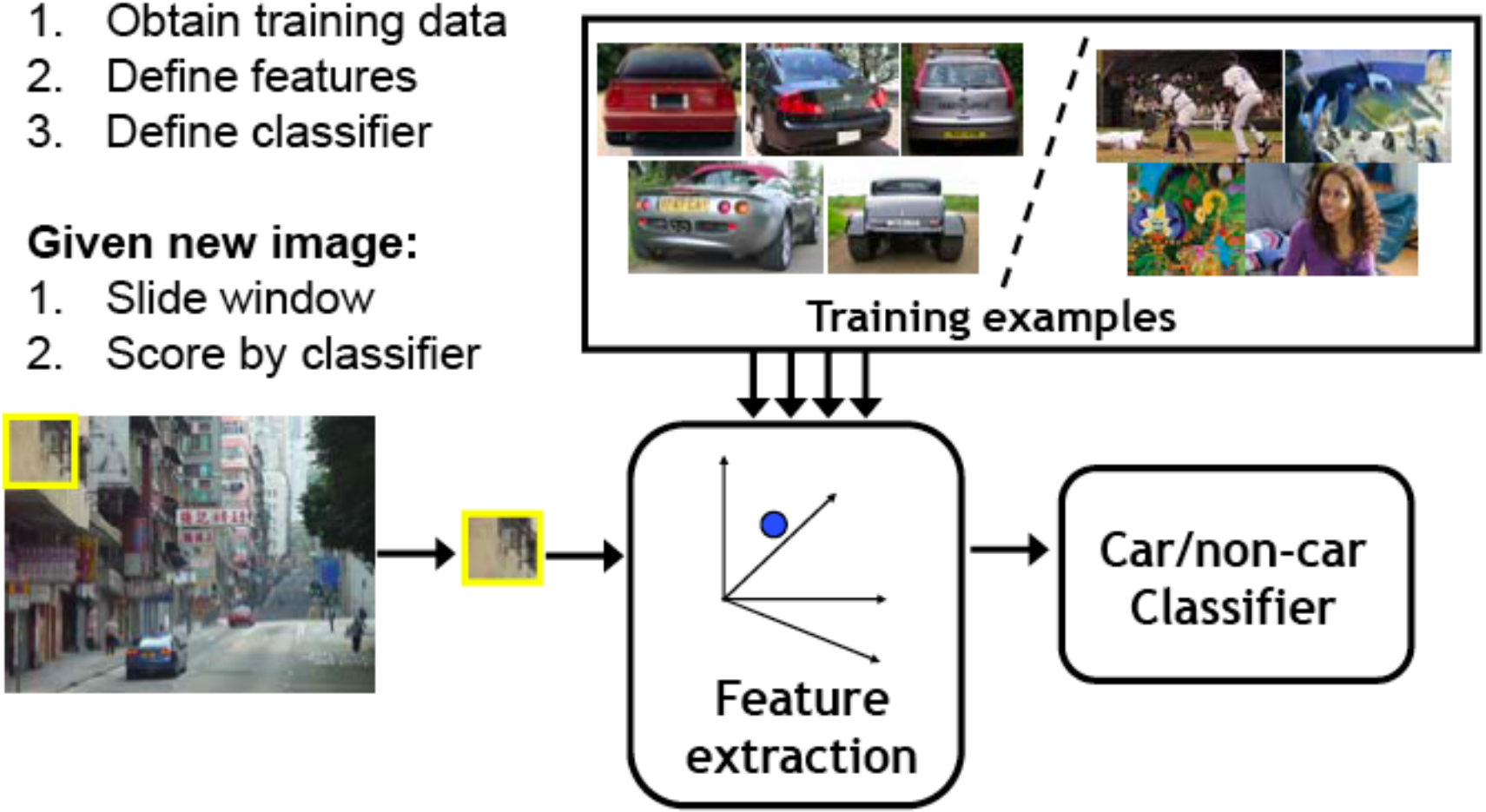
Sliding Window Recap

Training:

1. Obtain training data
2. Define features
3. Define classifier

Given new image:

1. Slide window
2. Score by classifier



[Source: K. Grauman]

Example: Dalal & Triggs

Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs
INRIA Rhône-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr

Abstract

We study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, we show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.

1 Introduction

We briefly discuss previous work on human detection in §2, give an overview of our method §3, describe our data sets in §4 and give a detailed description and experimental evaluation of each stage of the process in §5–6. The main conclusions are summarized in §7.

2 Previous Work

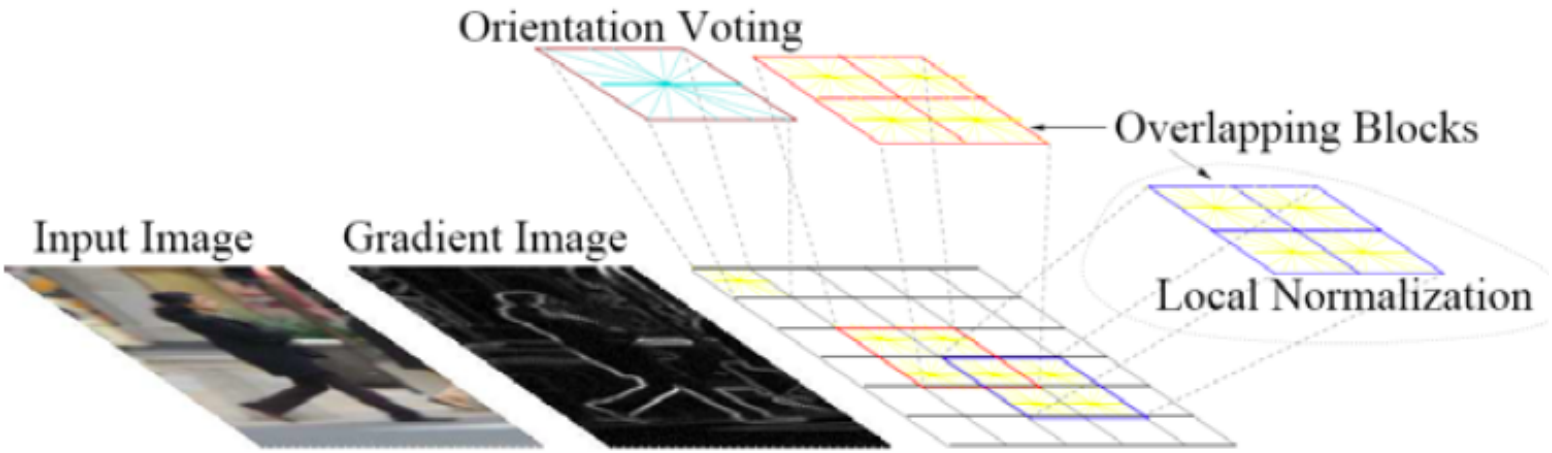
There is an extensive literature on object detection, but here we mention just a few relevant papers on human detection [18, 17, 22, 16, 20]. See [6] for a survey. Papageorgiou *et al* [18] describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in [17]. Depoortere *et al* give an optimized version of this [2]. Gavrila & Philomen [8] take a more direct approach, extracting edge images and matching them to a set of learned exemplars using chamfer distance. This has been used in a practical real-time pedestrian detection system [7]. Viola *et al* [22] build an efficient

Task to solve

- Pedestrian detection

Representation

- Histogram of gradients: [Schiele & Crowley, Freeman & Roth]
- Code available: http://pascal.inrialpes.fr/soft/olt/

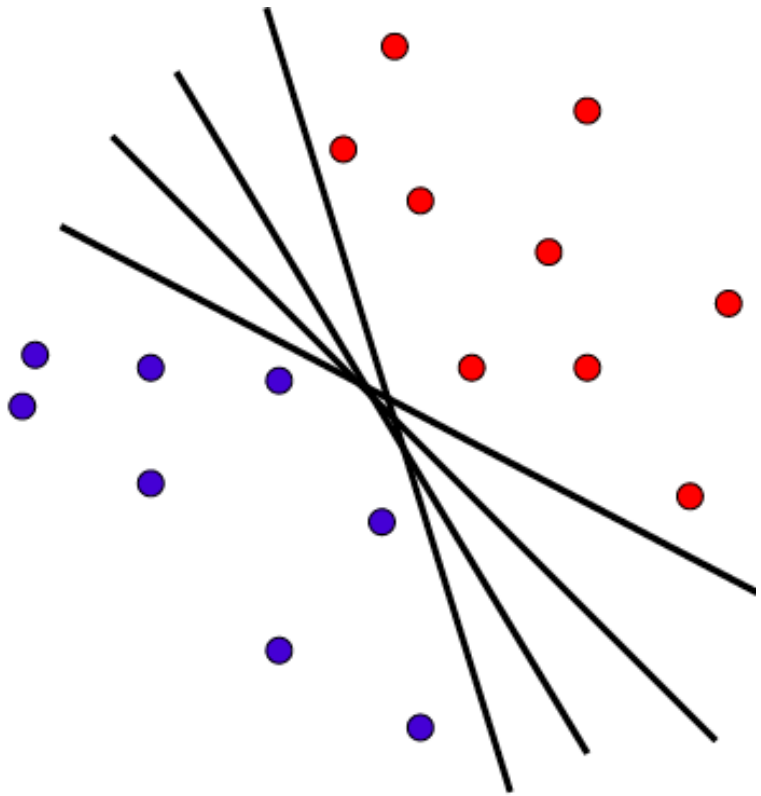


Linear Classifier

- Find linear function to separate positive and negative examples

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

- $f(\mathbf{x}) > 0$ if $\mathbf{x}$ is a positive example.
- $f(\mathbf{x}) < 0$ if $\mathbf{x}$ is a negative example.



Learning Setup

- Input $\mathbf{x} \in \mathcal{X}$, and outputs $y_i \in \{-1, 1\}$
- General setup: training set $\{\mathbf{x}_i, y_i\}$ sampled i.i.d. from $p(\mathbf{x}, y)$, we want to find parametric predictor $f \in \mathcal{F}$ that minimizes

$$R(f) = E_{\mathbf{x},y}\left[\bar{L}(f(\mathbf{x}_0;\theta), y)\right]$$

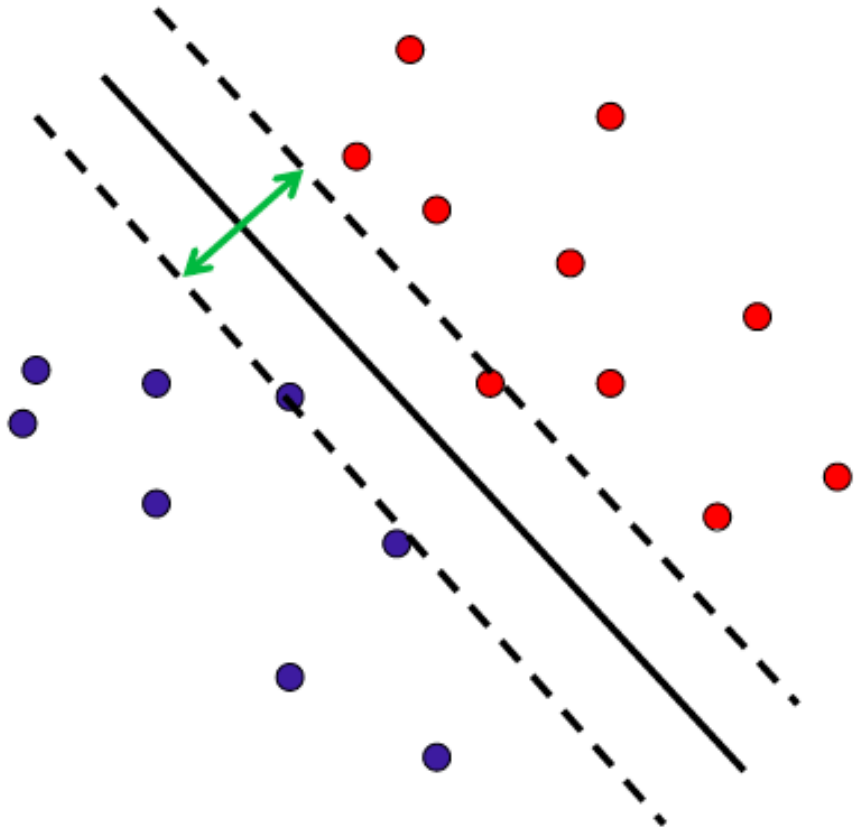
  with $\bar{L}$ the loss
- Regularized ERM:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} L(f(\mathbf{x}_i;\theta), y_i) + R(\theta)$$

- Surrogate loss L: square loss (ridge regression, GP), hinge (SVM), log loss (logistic regression)

Support Vector Machine (SVM)

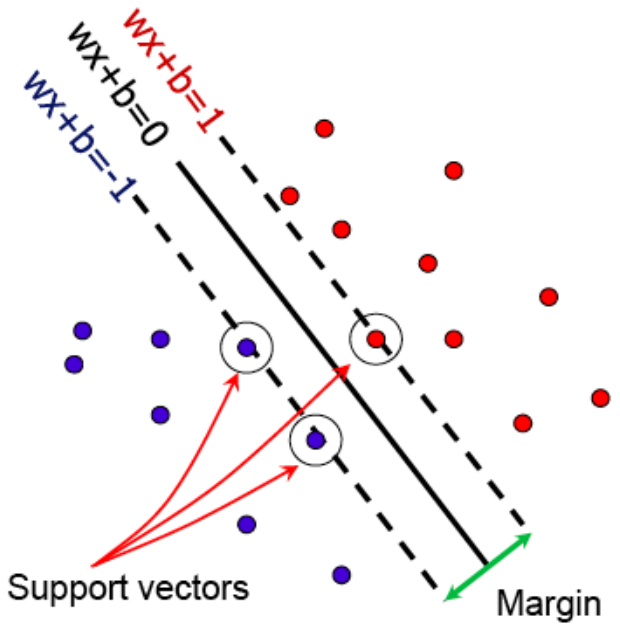
- Discriminative classifier based on optimal separating hyperplane
- Maximize the margin between the positive and negative training examples



[Source: G. Shakhnarovich]

Support Vector Machine (SVM)

- Maximize the margin between the positive and negative training examples



[Source: K. Grauman]

- Positive $\mathbf{y}_i = 1$: $\mathbf{w}^T\mathbf{x}_i + b \geq 1$
- Negative $\mathbf{y}_i = -1$: $\mathbf{w}^T\mathbf{x}_i + b \leq 1$
- Support vector: $\mathbf{w}^T\mathbf{x}_i + b = \pm 1$
- Point line distance: $\frac{\mathbf{y}(\mathbf{w}^T\mathbf{x}+b)}{||\mathbf{w}||}$
- For support vectors: $\frac{1}{||\mathbf{w}||}$
- Margin $M = \frac{2}{||\mathbf{w}||}$

Find the max margin hyperplane

- We want to maximize the margin

$$\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \right\}$$

- Hard optimization problem... but we can set

$$\min_i y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) = 1,$$

since we can rescale $\|\mathbf{w}\|$, b appropriately.

- Then, the optimization becomes a quadratic optimization problem

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \quad i = 1, \ldots, N. \end{aligned}$$

[Source: G. Shakhnarovich]

Find the max margin hyperplane

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad y_i(b + \mathbf{w}^T\mathbf{x}_i) - 1 \geq 0, \quad i = 1, \ldots, N.$$

- We can write the Lagrangian

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i(b + \mathbf{w}^T\mathbf{x}_i)\right]$$

- We can reformulate our problem now:

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i(b + \mathbf{w}^T\mathbf{x}_i)\right] \right\}$$

Max-margin optimization

- We want all the constraint terms to be zero:

$$\begin{aligned}
&\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i(b + \mathbf{w}^T \mathbf{x}_i)\right] \right\} \\
= &\min_{\mathbf{w}} \max_{\{\alpha_i \geq 0\}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i(b + \mathbf{w}^T \mathbf{x}_i)\right] \right\} \\
= &\max_{\{\alpha_i \geq 0\}} \min_{\mathbf{w}} \underbrace{\left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i(b + \mathbf{w}^T \mathbf{x}_i)\right] \right\}}_{J(\mathbf{w}, w_0; \alpha)}.
\end{aligned}$$

- We need to minimize $J(\mathbf{w}, b; \alpha)$ for any settings of $\alpha = [\alpha_1, \ldots, \alpha_N]^T$.

[Source: G. Shakhnarovich]

Strategy for optimization

- We need to find

$$\max_{\{\alpha_i \geq 0\}} \min_{\mathbf{w}} J(\mathbf{w}, b; \alpha)$$

- We will first fix α and treat $J(\mathbf{w}, b; \alpha)$ as a function of $\mathbf{w}$,b.
  - Find *functions* $\mathbf{w}(\alpha), b(\alpha)$ that attain the minimum.
- Next, treat $J(\mathbf{w}(\alpha), b(\alpha); \alpha)$ as a function of α.
  - Find α^* that attain the maximum.
- In the end, the solution is given by α^*, $\mathbf{w}(\alpha^*)$ and $b(\alpha^*)$.

[Source: G. Shakhnarovich]

Minimizing $J(\mathbf{w}, w_0; \alpha)$ with respect to $\mathbf{w}, b$

- For fixed α we can minimize

$$J(\mathbf{w}, b; \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i(b + \mathbf{w}^T \mathbf{x}_i)\right]$$

by setting derivatives w.r.t. $b, \mathbf{w}$ to zero:

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b; \alpha) = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0,$$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b; \alpha) = -\sum_{i=1}^{N} \alpha_i y_i = 0.$$

- Note that the bias term b dropped out but has produced a "global" constraint on α.

[Source: G. Shakhnarovich]

Solving for α

$$\underbrace{\mathbf{w}(\alpha) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i,}_{\text{Representer theorem!}} \qquad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

- Now can substitute this solution into

$$\max_{\{\alpha_i \geq 0, \sum_i \alpha_i y_i = 0\}} \left\{ \frac{1}{2} \|\mathbf{w}(\alpha)\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i (b(\alpha) + \mathbf{w}(\alpha)^T \mathbf{x}_i)\right] \right\}$$

$$= \max_{\{\alpha_i \geq 0, \sum_i \alpha_i y_i = 0\}} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right\}.$$

[Source: G. Shakhnarovich]

Max-margin and quadratic programming

- We started by writing down the max-margin problem and arrived at the *dual problem* in α:

$$\max \left\{ \sum_{i=1}^{N} \alpha_i \; - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right\}$$

$$\text{subject to} \sum_{i=1}^{N} \alpha_i y_i = 0, \; \alpha_i \geq 0 \text{ for all } i = 1, \ldots, N.$$

- Solving this *quadratic program* yields α^*.
- We substitute α^* back to get $\mathbf{w}$:

$$\hat{\mathbf{w}} \; = \; \mathbf{w}(\alpha^*) \; = \; \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i$$

[Source: G. Shakhnarovich]

Maximum margin decision boundary

$$\hat{\mathbf{w}} = \mathbf{w}(\alpha^*) = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i$$

- Suppose that, under the optimal solution, the margin (distance to the boundary) of a particular $\mathbf{x}_i$ is

$$y_i \left(b + \hat{\mathbf{w}}^T \mathbf{x}_i \right) > 1.$$

- Then, necessarily, $\alpha_i^* = 0 \Rightarrow$ not a support vector.
- The direction of the max-margin decision boundary is

$$\hat{\mathbf{w}} = \sum_{\alpha_i^* > 0} \alpha_i^* y_i \mathbf{x}_i.$$

- b is set by making the margin equidistant to two classes.

[Source: G. Shakhnarovich]

Support vectors

$$\hat{\mathbf{w}} = \sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i.$$

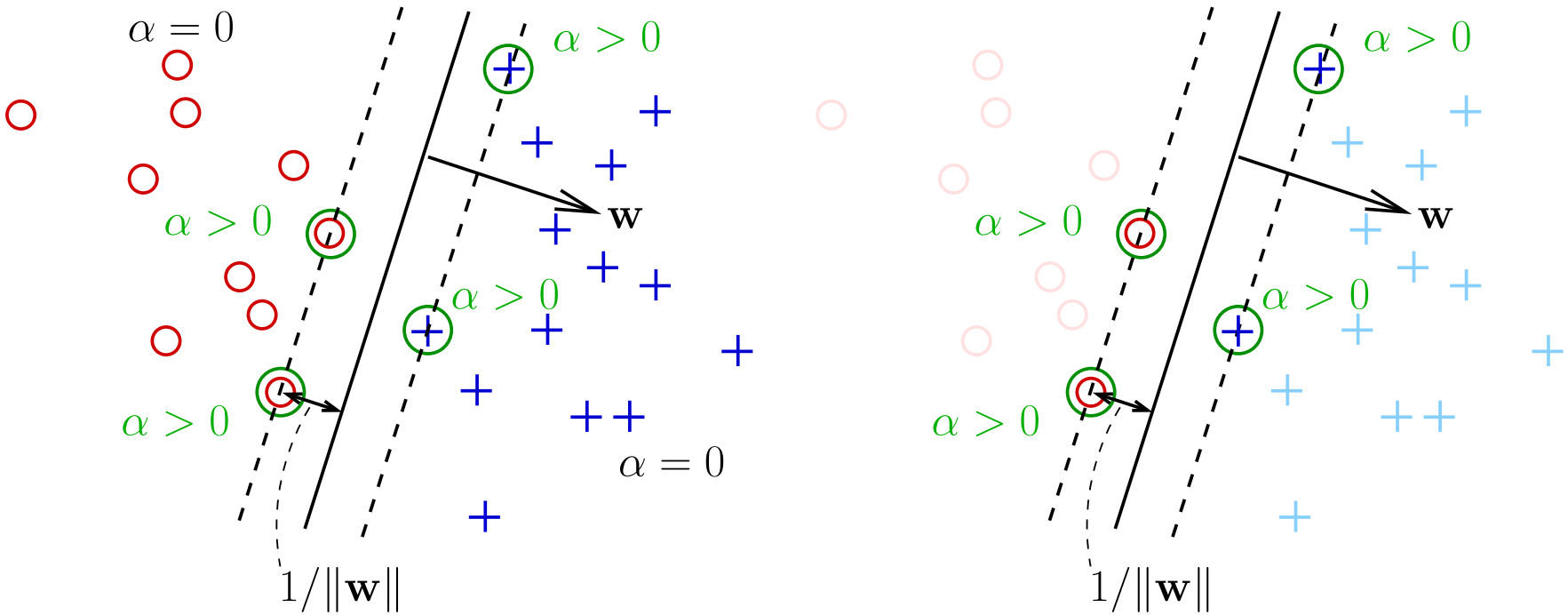
- Given a test example $\mathbf{x}$, it is classified by

$$\begin{aligned} \hat{y} &= \text{sign}\left(\hat{b} + \hat{\mathbf{w}}^T \mathbf{x}\right) \\ &= \text{sign}\left(\hat{b} + (\sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i)^T \mathbf{x}\right) \\ &= \text{sign}\left(\hat{b} + \sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}\right) \end{aligned}$$

- The classifier is based on the expansion in terms of dot products of $\mathbf{x}$ with support vectors.

[Source: G. Shakhnarovich]

SVM classification



[Source: G. Shakhnarovich]

SVM: summary so far

- We started with $\underset{\mathbf{w},b}{\text{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \right\}$
- In linearly separable case, we get a quadratic program

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right\}$$

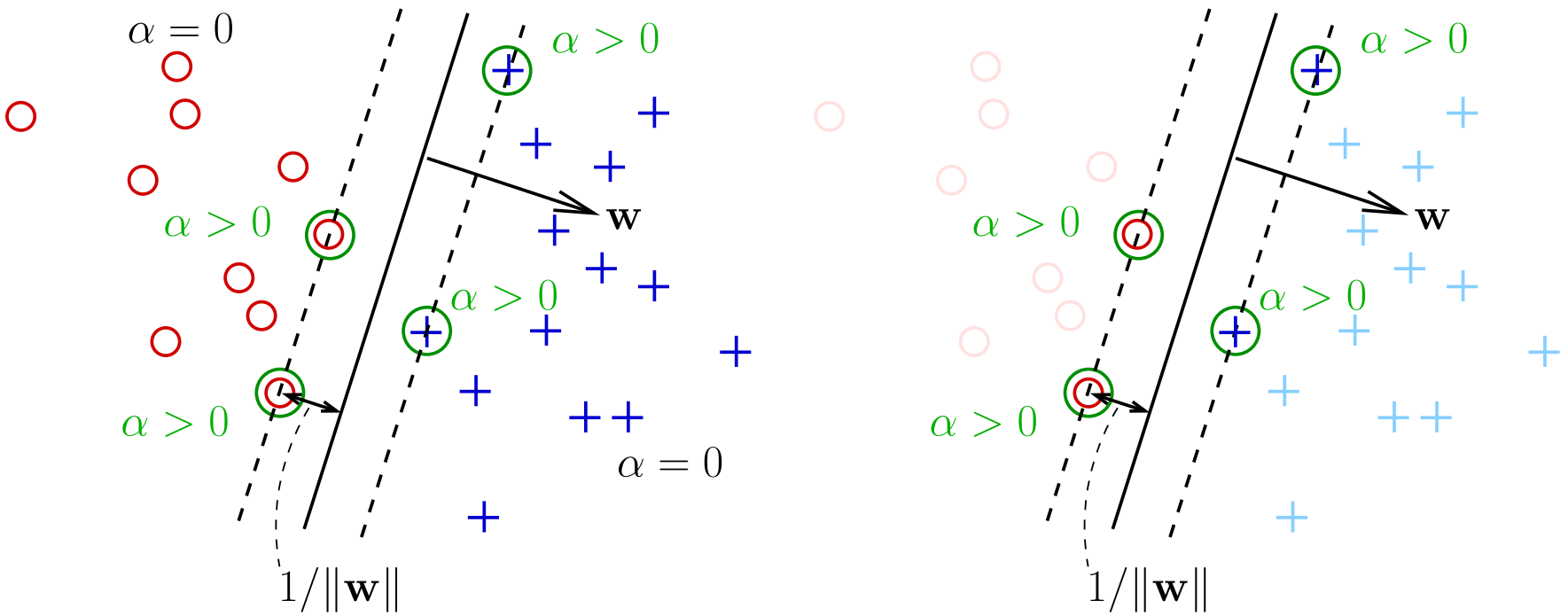
$$\text{subject to} \sum_{i=1}^{N} \alpha_i y_i = 0, \ \alpha_i \geq 0 \text{ for all } i = 1, \ldots, N.$$

- Solving it for α we get the SVM classifier

$$\hat{y} = \text{sign} \left(\hat{b} + \sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} \right).$$

[Source: G. Shakhnarovich]

SVM classification



- Only support vectors (points with $\alpha_i > 0$) determine the boundary

[Source: G. Shakhnarovich]

Non-separable case

- Not linearly separable data: we can no longer satisfy $y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1$ for all i.
- We introduce *slack variables* $\xi_i \geq 0$:

$$y_i \left(b + \mathbf{w}^T \mathbf{x}_i\right) - 1 + \xi_i \geq 0.$$

- Whenever the original constraint is satisfied, $\xi_i = 0$.

[Source: G. Shakhnarovich]

Non-separable case

- Not linearly separable data: we can no longer satisfy $y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1$ for all i.
- We introduce *slack variables* $\xi_i \geq 0$:

$$y_i \left(b + \mathbf{w}^T \mathbf{x}_i\right) - 1 + \xi_i \geq 0.$$

- Whenever the original constraint is satisfied, $\xi_i = 0$.
- The updated objective:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i.$$

- The parameter C determines the penalty paid for violating margin constraints.
- This is applicable even when the data *are* separable!

[Source: G. Shakhnarovich]

Non-separable case

- Not linearly separable data: we can no longer satisfy $y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1$ for all i.
- We introduce *slack variables* $\xi_i \geq 0$:

$$y_i \left(b + \mathbf{w}^T \mathbf{x}_i\right) - 1 + \xi_i \geq 0.$$

- Whenever the original constraint is satisfied, $\xi_i = 0$.
- The updated objective:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i.$$

- The parameter C determines the penalty paid for violating margin constraints.
- This is applicable even when the data *are* separable!

[Source: G. Shakhnarovich]

Non-separable case: solution

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i.$$

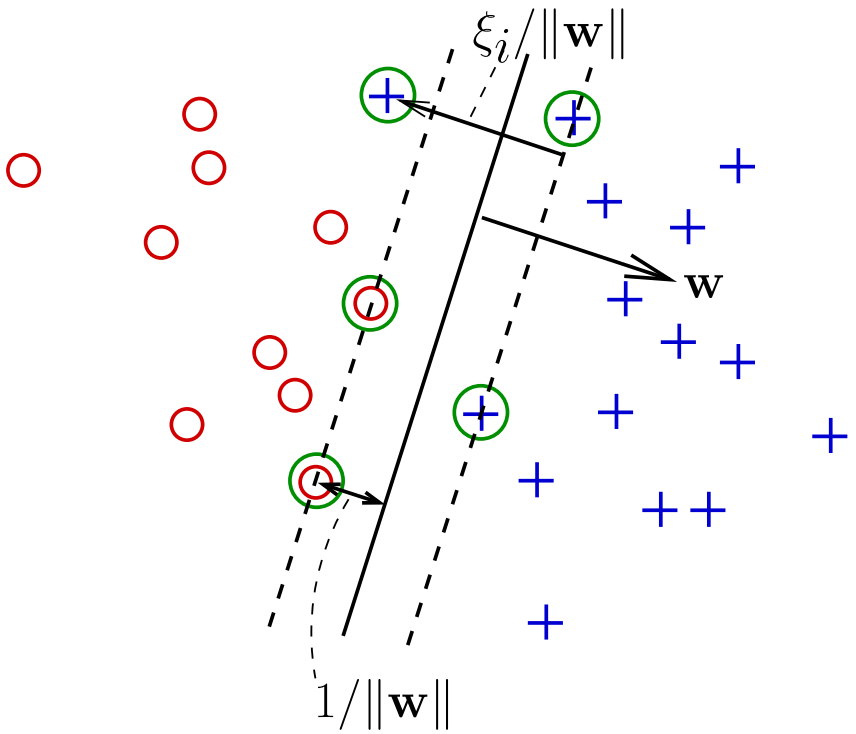
- We can solve this using Lagrange multipliers
  - Introduce additional multipliers for the ξs, as they have to be positive.
- The resulting dual problem:

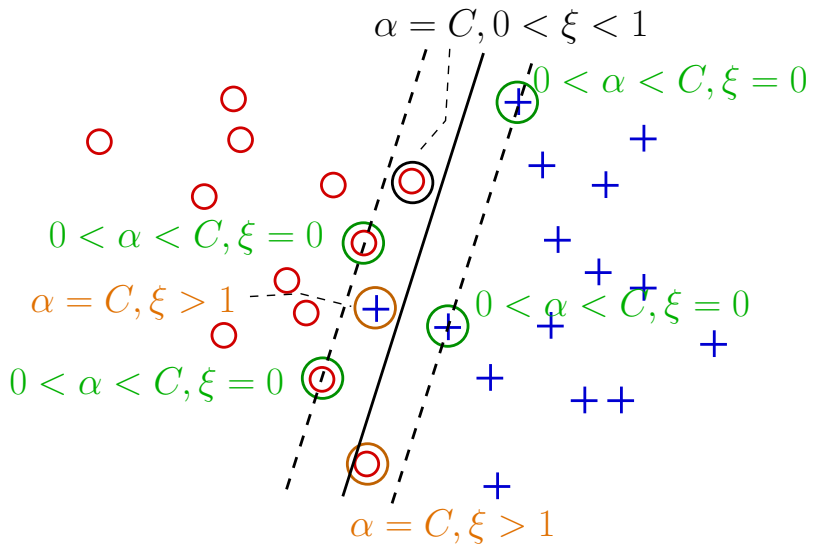
$$\max\left\{\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j\right\}$$

$$\text{subject to } \sum_{i=1}^{N}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C \text{ for all } i = 1,\ldots,N.$$

[Source: G. Shakhnarovich]

SVM with slack variables





- Support vectors: points with $\alpha > 0$
- If $0 < \alpha < C$: SVs on the margin, $\xi = 0$.
- If $0 < \alpha = C$: SVs over the margin, either misclassified ($\xi > 1$) or not ($0 < \xi \leq 1$).

[Source: G. Shakhnarovich]

SVM and regularization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

- C is a regularization parameter, controlling penalty for imperfect fit to training labels.
- Larger $C \Rightarrow$ more reluctant to make mistakes
- How do we select value of C? Cross validation is a common practical way to do that.

[Source: G. Shakhnarovich]

SVM and regularization

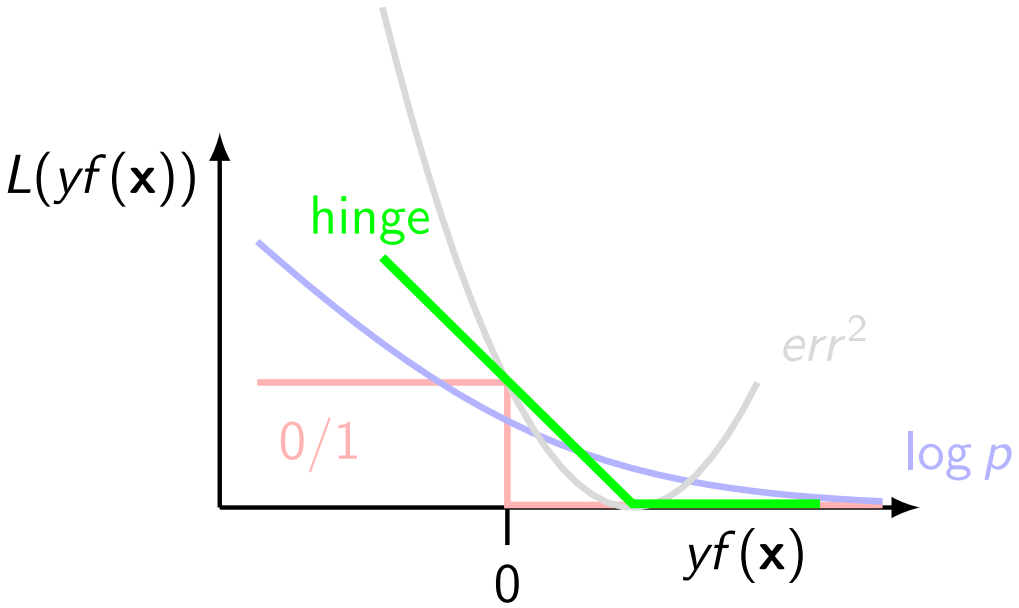
$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

- C is a regularization parameter, controlling penalty for imperfect fit to training labels.
- Larger $C \Rightarrow$ more reluctant to make mistakes
- How do we select value of C? Cross validation is a common practical way to do that.

[Source: G. Shakhnarovich]

Loss in SVM

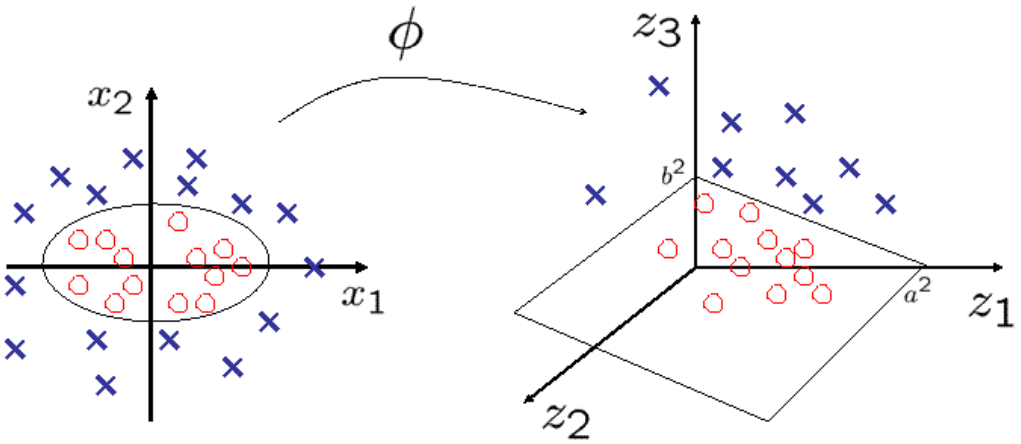
- Loss is measured as $\sum_{i=1}^{N} \xi_i$
- This surrogate loss is known as *hinge loss*



[Source: G. Shakhnarovich]

Nonlinear features

- We can move to nonlinear classifiers by mapping data into nonlinear *feature space*.



$$\phi : [x_1, x_2]^T \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$

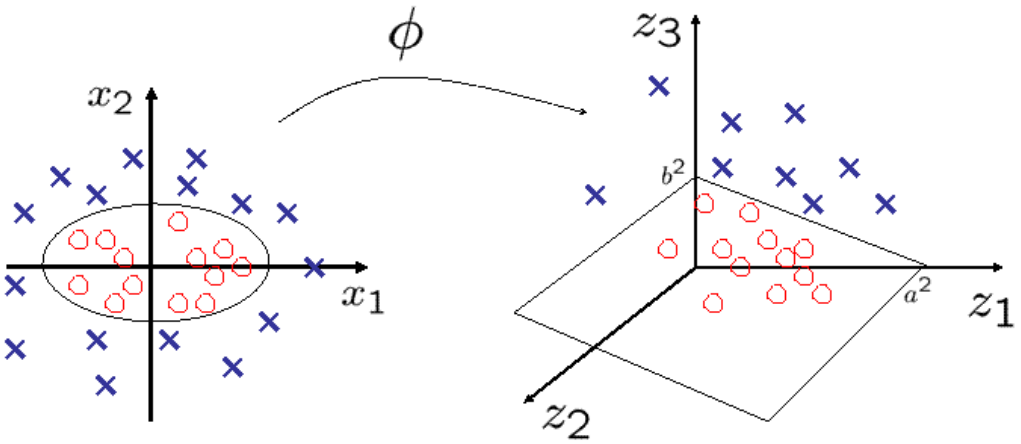
- Elliptical decision boundary in the input space becomes linear in the feature space $\mathbf{z} = \phi(\mathbf{x})$:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = c \Rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = c.$$

[Source: G. Shakhnarovich]

Nonlinear features

- We can move to nonlinear classifiers by mapping data into nonlinear *feature space*.



$$\phi : [x_1, x_2]^T \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$

- Elliptical decision boundary in the input space becomes linear in the feature space $\mathbf{z} = \phi(\mathbf{x})$:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = c \Rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = c.$$

[Source: G. Shakhnarovich]

Representer theorem

- Consider the optimization problem

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \; \forall i$$

- Theorem: the solution can be represented as

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$$

- This is the "magic" behind Support Vector Machines!

[Source: G. Shakhnarovich]

Example of nonlinear mapping

- Consider the mapping: $\phi : [x_1, x_2]^T \rightarrow [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$.
- The (linear) SVM classifier in the feature space:

$$\hat{y} = \text{sign}\left(\hat{b} + \sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})\right)$$

- The dot product in the feature space:

$$\begin{aligned}\phi(\mathbf{x})^T \phi(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= \left(1 + \mathbf{x}^T\mathbf{z}\right)^2.\end{aligned}$$

[Source: G. Shakhnarovich]

Dot products and feature space

- We defined a non-linear mapping into feature space

$$\phi : [x_1, x_2]^T \rightarrow [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$$

  and saw that $\phi(\mathbf{x})^T\phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z})$ using the *kernel*

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T\mathbf{z})^2.$$

- I.e., we can calculate dot products in the feature space implicitly, without ever writing the feature expansion!

[Source: G. Shakhnarovich]

The kernel trick

- Replace dot products in the SVM formulation with kernel values.
- The optimization problem:

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

  - Need to compute the *kernel matrix* for the training data

- The classifier:

$$\hat{y} = \text{sign} \left(\hat{b} + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

  - Need to compute $K(\mathbf{x}_i, \mathbf{x})$ for all SVs $\mathbf{x}_i$.

[Source: G. Shakhnarovich]

Mercer's kernels

- What kind of function K is a valid kernel, i.e. such that there exists a feature space $\Phi(\mathbf{x})$ in which $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$?
- Theorem due to Mercer (1930s): K must be
  - Continuous;
  - symmetric: $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})$;
  - positive definite: for any $\mathbf{x}_1, \ldots, \mathbf{x}_N$, the *kernel matrix*

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & K(\mathbf{x}_1, \mathbf{x}_N) \\ \cdots & \cdots & \cdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

  must be positive definite.

[Source: G. Shakhnarovich]

Some popular kernels

- The linear kernel:

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}.$$

  This leads to the original, linear SVM.

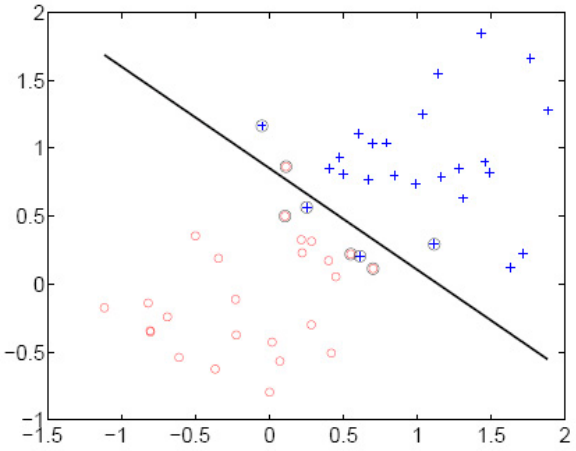
- The polynomial kernel:

$$K(\mathbf{x}, \mathbf{z};\, c, d) = (c + \mathbf{x}^T \mathbf{z})^d.$$

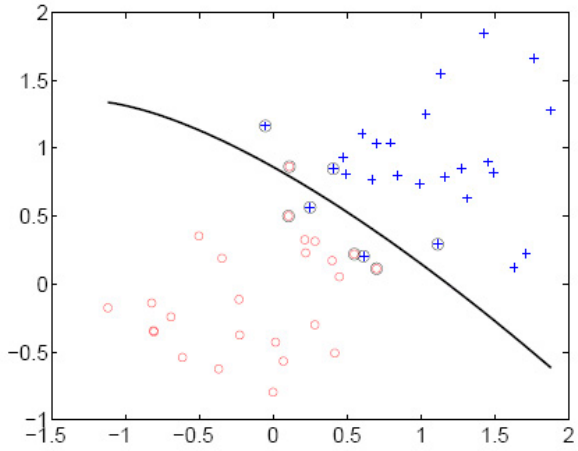
  We can write the expansion explicitly, by concatenating powers up to d and multiplying by appropriate weights.

[Source: G. Shakhnarovich]

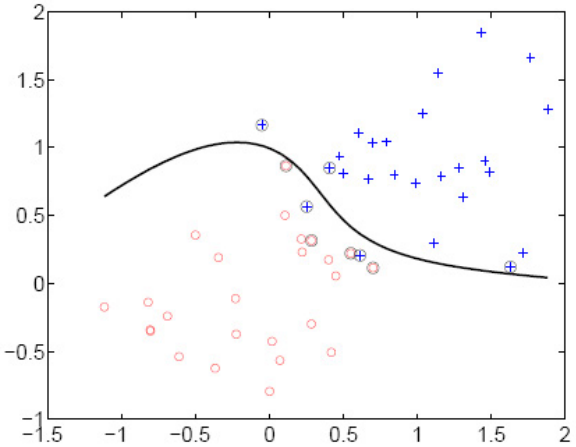
Example: SVM with polynomial kernel



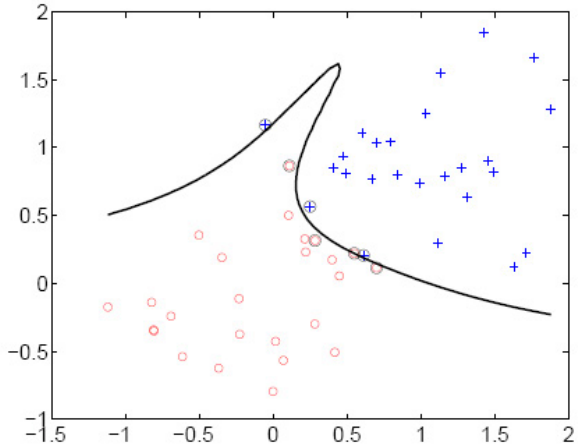
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

(using $C < \infty$)

[Source: G. Shakhnarovich]

Radial basis function kernel

$$K(\mathbf{x}, \mathbf{z}; \sigma) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right).$$

- The RBF kernel is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ? Consider $\sigma \to 0$.

Radial basis function kernel

$$K(\mathbf{x}, \mathbf{z}; \sigma) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right).$$

- The RBF kernel is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ? Consider $\sigma \to 0$.

$$K(\mathbf{x}_i, \mathbf{x}; \sigma) \to \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}_i. \end{cases}$$

- All examples become SVs $\Rightarrow$ likely overfitting.

[Source: G. Shakhnarovich]

Radial basis function kernel

$$K(\mathbf{x}, \mathbf{z}; \sigma) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right).$$

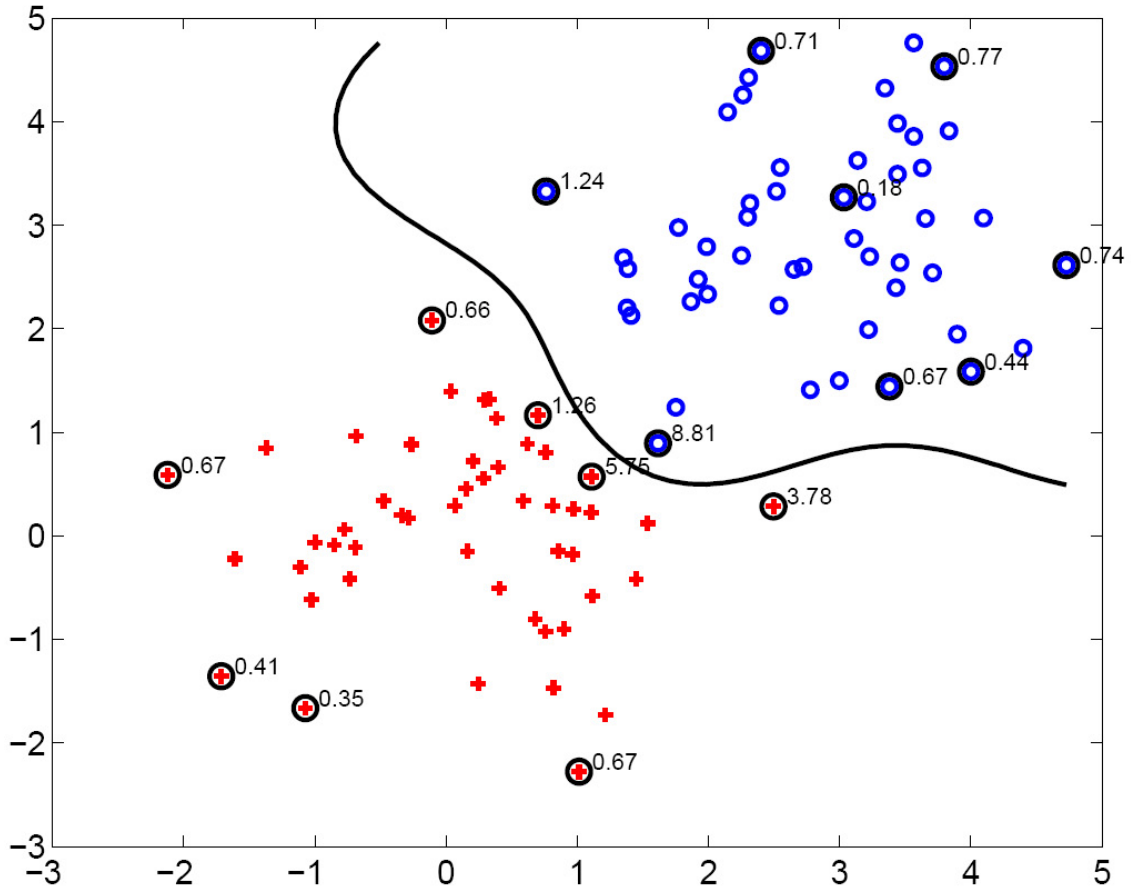
- The RBF kernel is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ? Consider $\sigma \to 0$.

$$K(\mathbf{x}_i, \mathbf{x}; \sigma) \to \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}_i. \end{cases}$$

- All examples become SVs $\Rightarrow$ likely overfitting.

[Source: G. Shakhnarovich]

SVM with RBF (Gaussian) kernels



- Data are linearly separable in the (infinite-dimensional) feature space
- We don't need to explicitly compute dot products in that feature space – instead we simply evaluate the RBF kernel.

SVM: summary

- Two main ideas:
  - large margin classification,
  - the kernel trick.
- Complexity of classifier depends on the number of SVs.
  - Controlled indirectly by C and kernel parameters.
- One of the most successful ML techniques applied to computer vision!
- Recommended off-the-shelf package: SVM^{light}
  `http://svmlight.joachims.org`

[Source: G. Shakhnarovich]

SVM for Visual Recognition

1. Define your representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples
4. Use this kernel matrix to solve for SVM support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.

[Source: K. Grauman]

Multi-class SVMs

Achieve multi-class classifier by combining a number of binary classifiers

- One vs. all
  - Training: learn an SVM for each class vs. the rest
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM votes for a class to assign to the test example

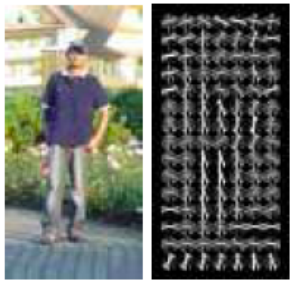
[Source: K. Grauman]

Which detectors?

Window-based

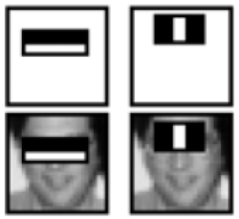
NN + scene Gist classification

e.g., Hays & Efros



SVM + person detection

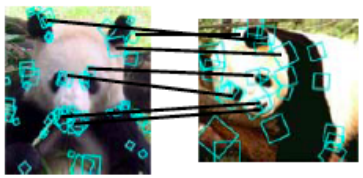
e.g., Dalal & Triggs



Boosting + face detection

Viola & Jones

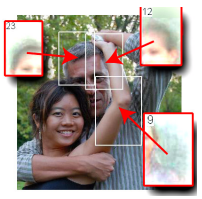
Part-based



BOW, pyramids
e.g., [Grauman et al.]

ISM: voting
e.g., [Leibe & Shiele]

deformable parts
e.g., [Felzenszwalb et al.]



poselets
[Bourdev et al.]

Viola-Jones face detector

ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

Abstract

This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small num-

tected at 15 frames per second on a conventional 700 MHz Intel Pentium III. In other face detection systems, auxiliary information, such as image differences in video sequences, or pixel color in color images, have been used to achieve high frame rates. Our system achieves high frame rates working only with the information present in a single grey scale image. These alternative sources of information can also be integrated with our system to achieve even higher frame rates.

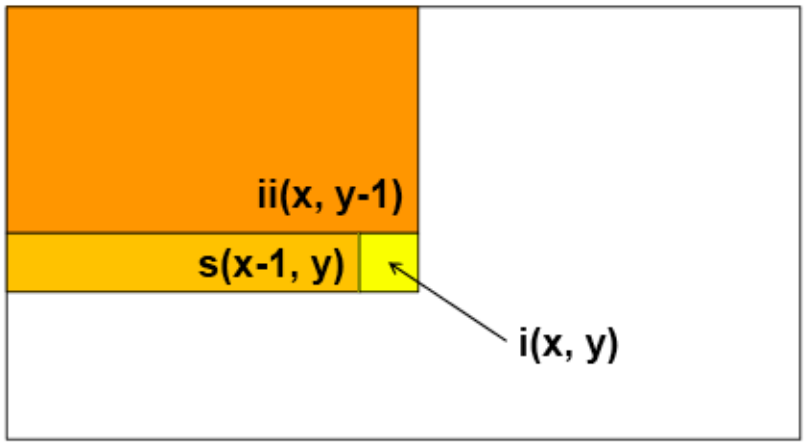
There are three main contributions of our object detec-

Main idea

- Represent local texture with rectangular features within window of interest
- Use integral images to compute the features efficiently
- Select discriminative features to be weak classifiers
- Use boosted combination of them as final classifier
- Form a cascade of such classifiers, rejecting clear negatives quickly

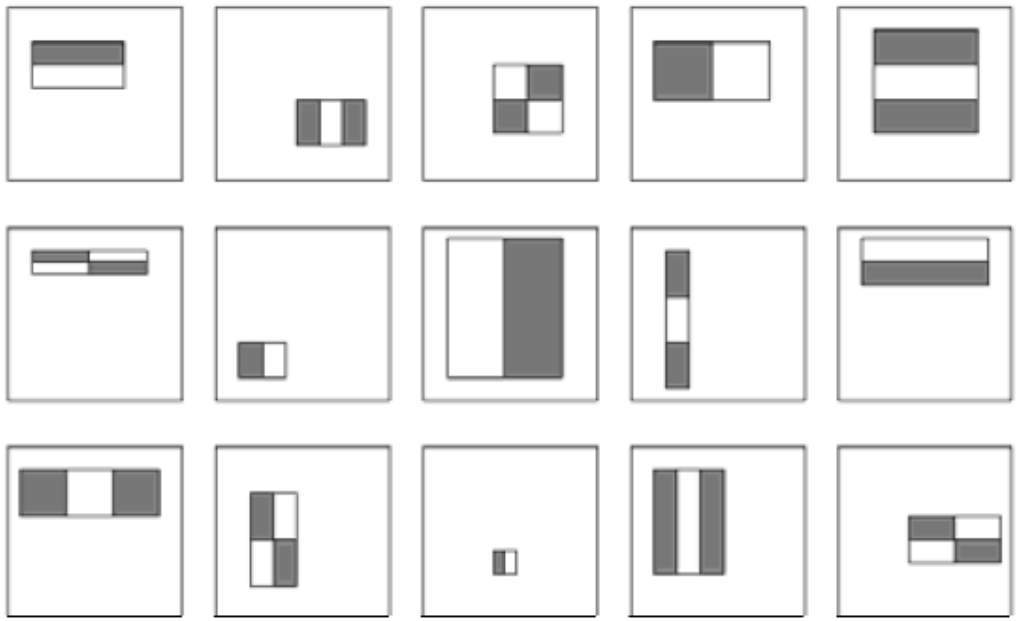
Efficient computation of the features

- **Rectangular filters**: Feature output is the difference between adjacent regions
- Can be computed efficiently with integral images: any sum can be computed in constant time
- Cumulative row sum $s(x, y) = s(x - 1, y) + i(x, y)$
- Integral image: $ii(x, y) = ii(x, y - 1) + s(x, y)$
- Avoid scaling images: scale features directly for same cost.



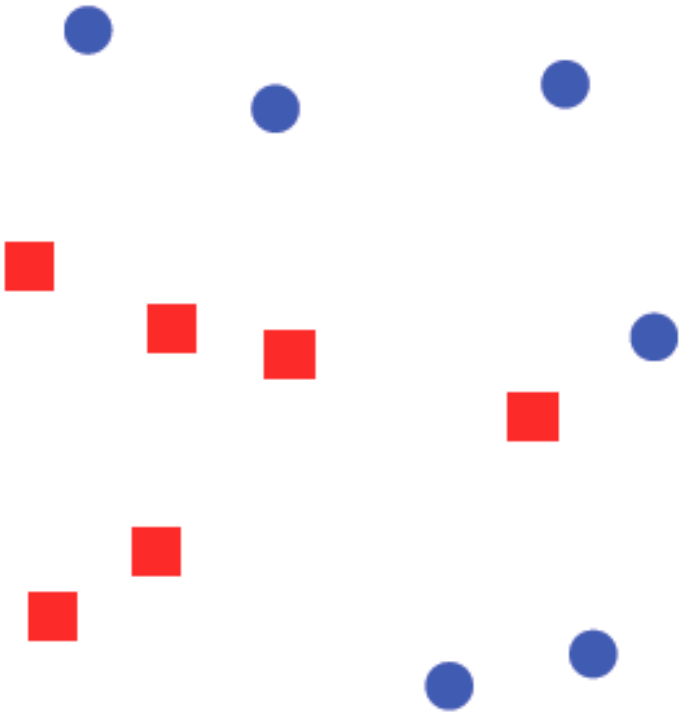
Representation

- **Rectangular filters**: difference between adjacent regions
- Consider all possible filter parameters: position, scale, and type. 180,000 features for 24×24 window.
- Which subset of these features should we use?
- Use AdaBoost to select the informative features and to form the classifier



Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



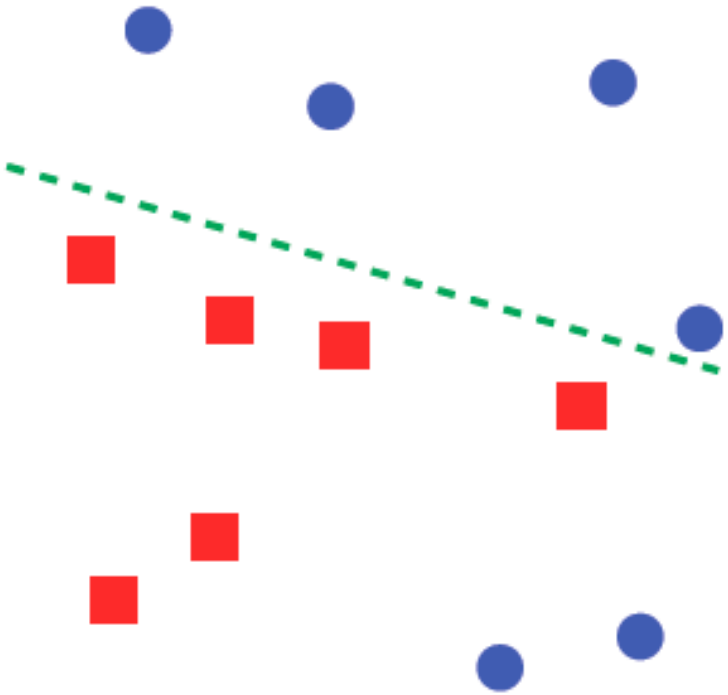
Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



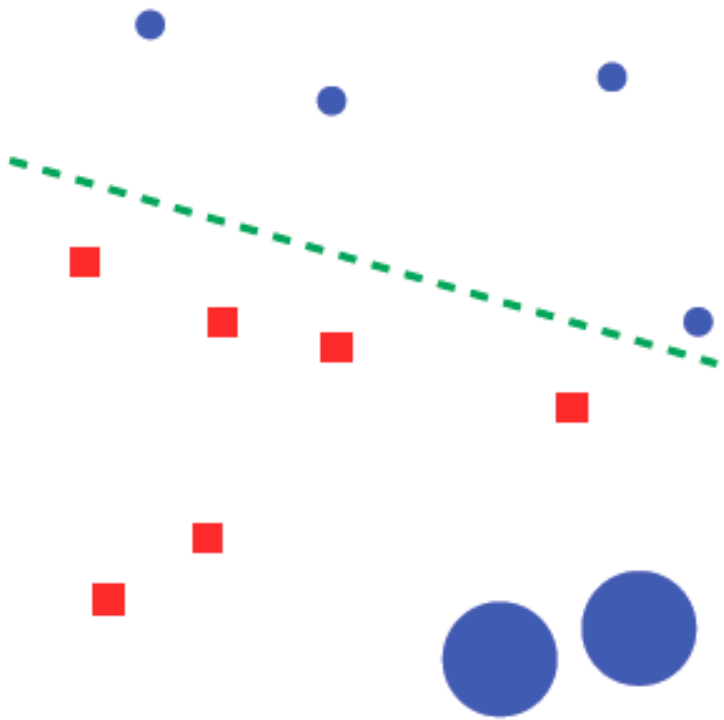
Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



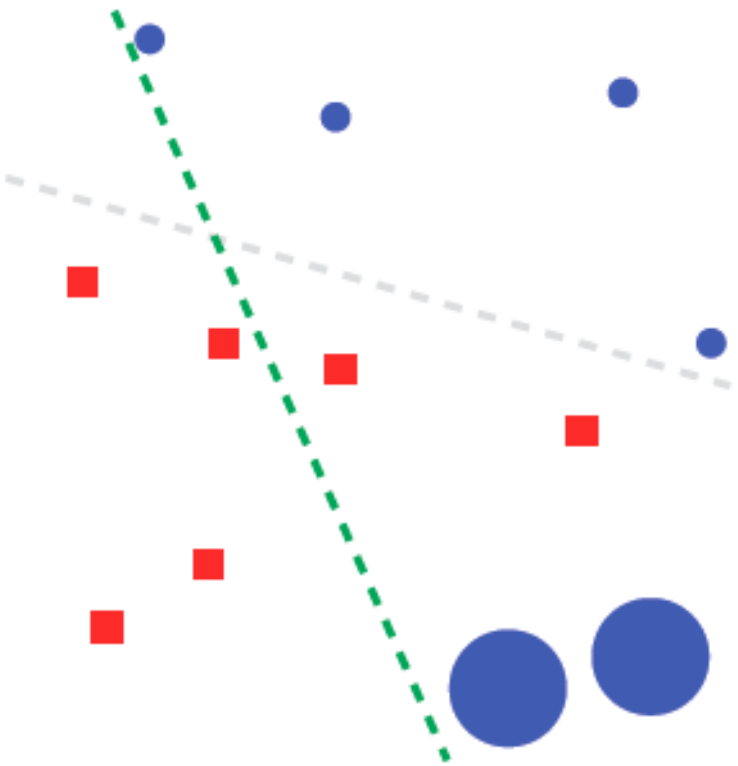
Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



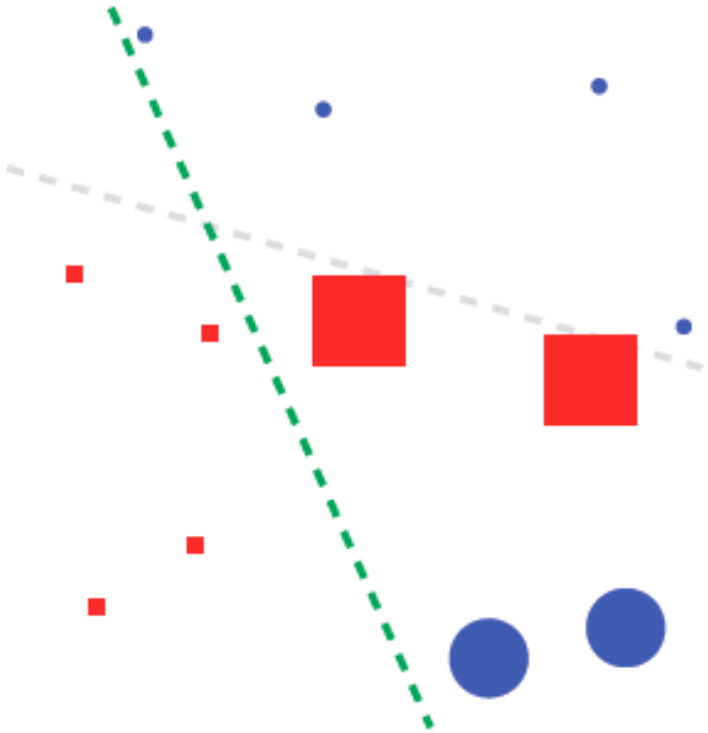
Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

[Source: G. Shakhnarovich]

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



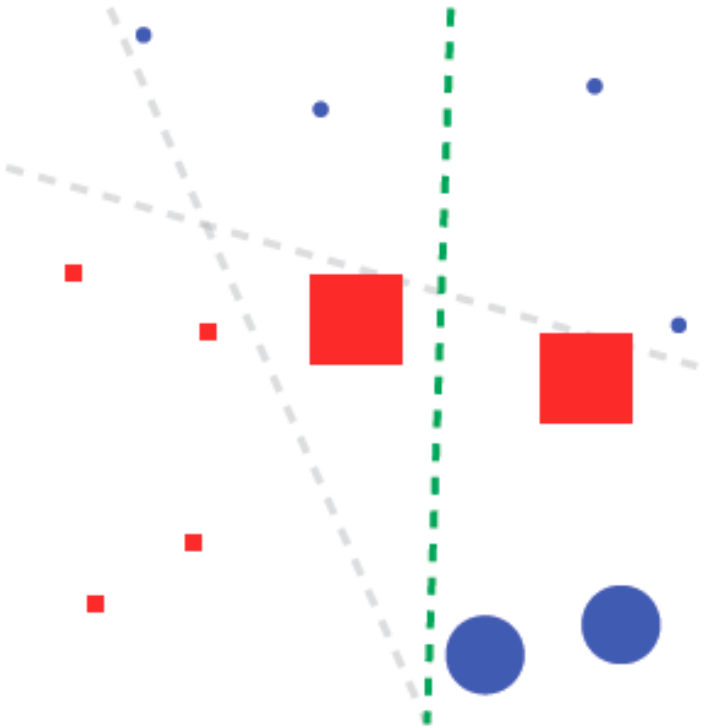
[Source: G. Shakhnarovich]

Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



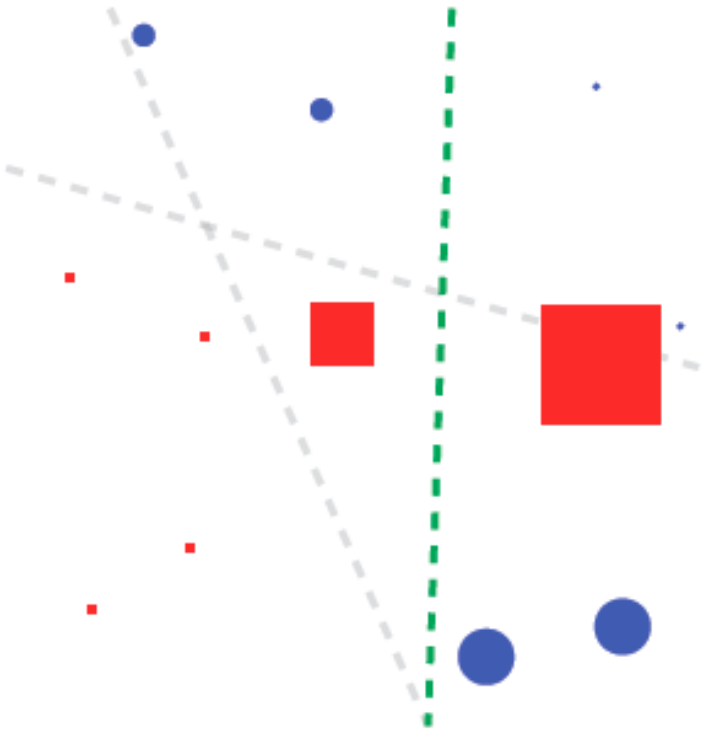
[Source: G. Shakhnarovich]

Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

Boosting: intuition

Want to pick weak classifiers that contribute something to the ensemble.



[Source: G. Shakhnarovich]

Greedy algorithm: for $m = 1, \ldots, M$

- Pick a weak classifier h_m
- Adjust weights: misclassified examples get "heavier"
- α_m set according to weighted error of h_m

Boosting: training

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest weighted training error
  - Raise weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

[Source: S. Lazebnik]

AdaBoost: algorithm summary

1. Initialize weights: $W_i^{(0)} = 1/N$
2. Iterate for $m = 1, \ldots, M$:
   - Find (any) "weak" classifier h_m that attains weighted error

$$\epsilon_m = \frac{1}{2}\left(1 - \sum_{i=1}^{N} W_i^{(m-1)} y_i h_m(\mathbf{x}_i)\right) < \frac{1}{2}$$

   - Let $\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}$.
   - Update the weights and normalize so that $\sum_i W_i^{(m)} = 1$:

$$W_i^{(m)} = \frac{1}{Z} W_i^{(m-1)} e^{-\alpha_m y_i h_m(\mathbf{x}_i)},$$

3. The combined classifier: $\text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$

[Source: G. Shakhnarovich]

Boosting: pros and cons

Advantages

- Integrates classification with feature selection: only one feature in VJ detector
- Complexity of training is linear in the number of training examples
- Flexibility in the choice of weak learners, boosting scheme
- Testing is fast
- Easy to implement

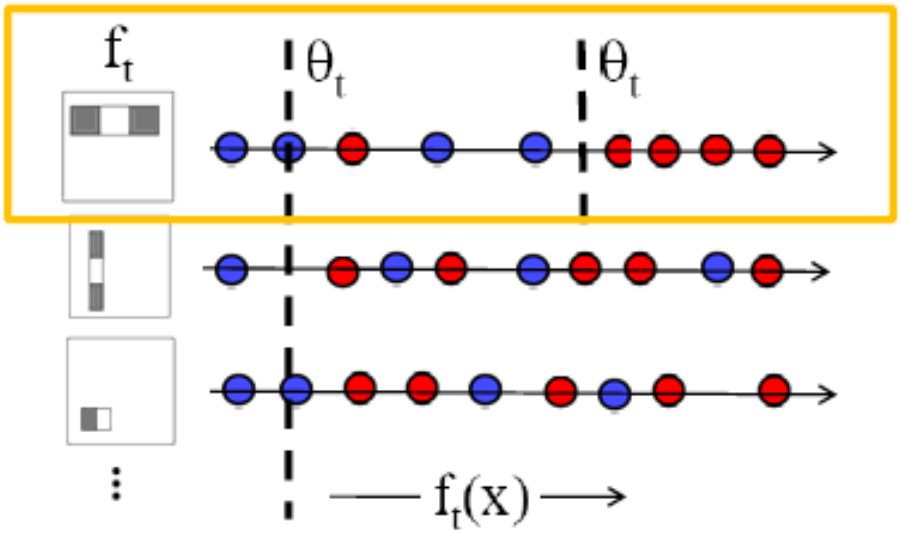
Disadvantages

- Needs many training examples
- Often found not to work as well as SVMs, GPs, etc.

Viola Jones Adaboost

- Select the single feature and threshold that best separates positive (faces) and negative (nonfaces) training examples, in terms of weighted error.
- Weak classifier is defined as

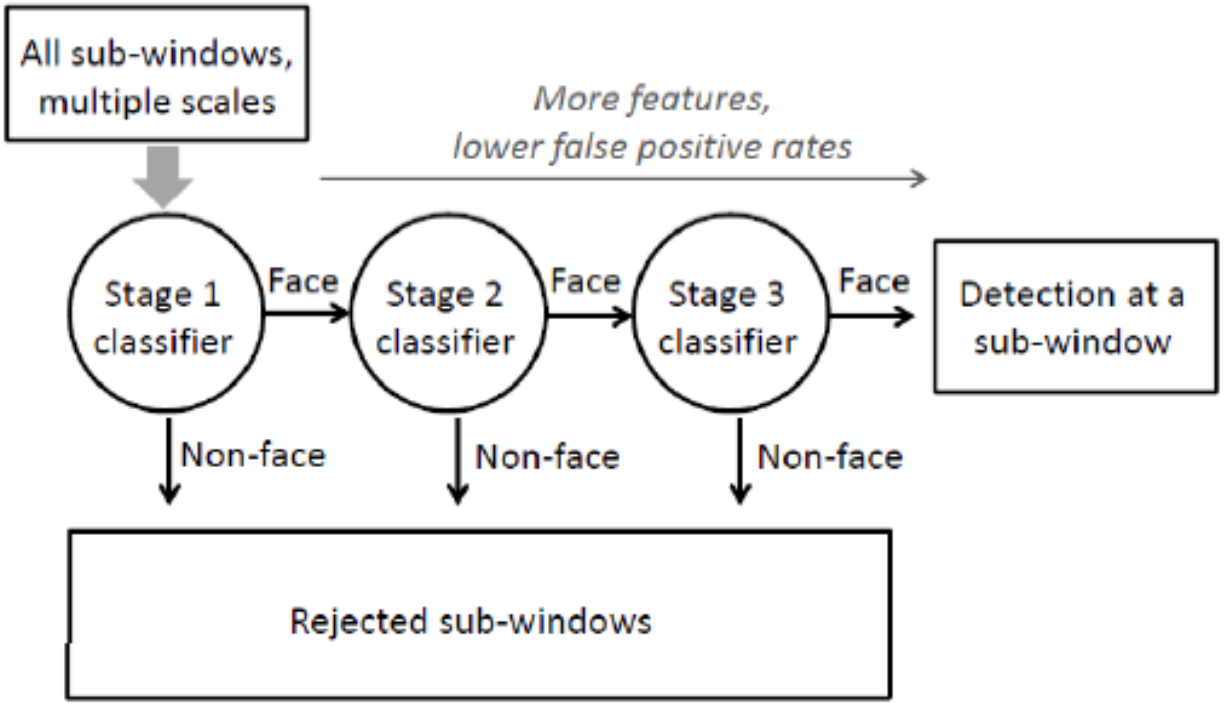
$$h_i(x) = \begin{cases} +1 & \text{if } f_i(x) > \theta_i \\ -1 & \text{Otherwise} \end{cases}$$



- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Cascading Classifiers

- Form a cascade with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative



[K. Grauman]

Training the cascade

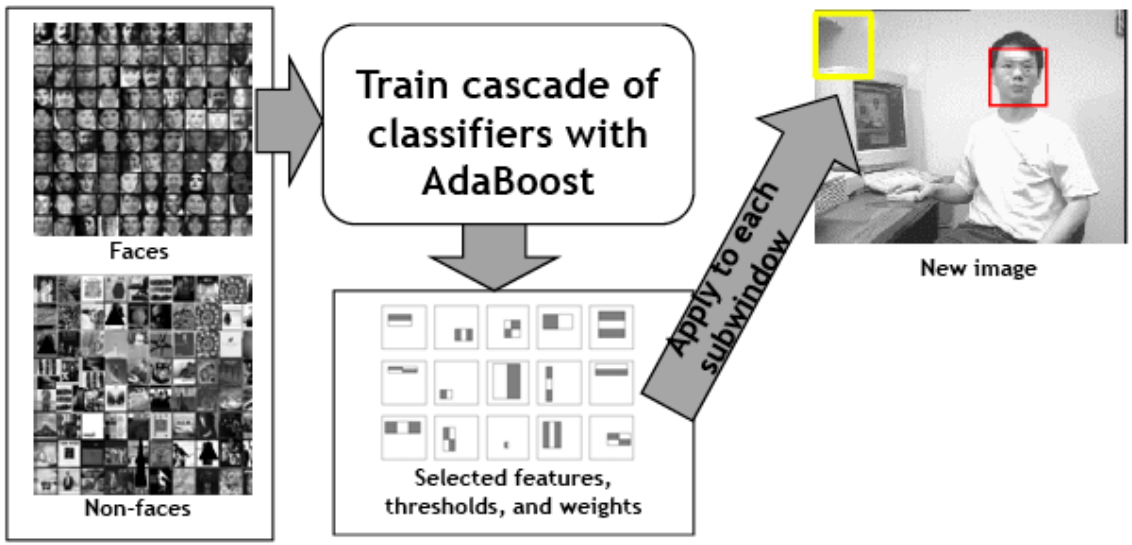
- Set target detection and false positive rates for each stage
- Keep adding features until its target rates have been met
  - Low AdaBoost threshold to max detection (don't minimize total classification error)
  - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

First classifier

- Only uses 2 features
- Detects 100% of the faces, and only 40% false positives.

Viola Jones detector

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade 6061 features in all layers
- Code: `http://www.intel.com/technology/computing/opencv/`

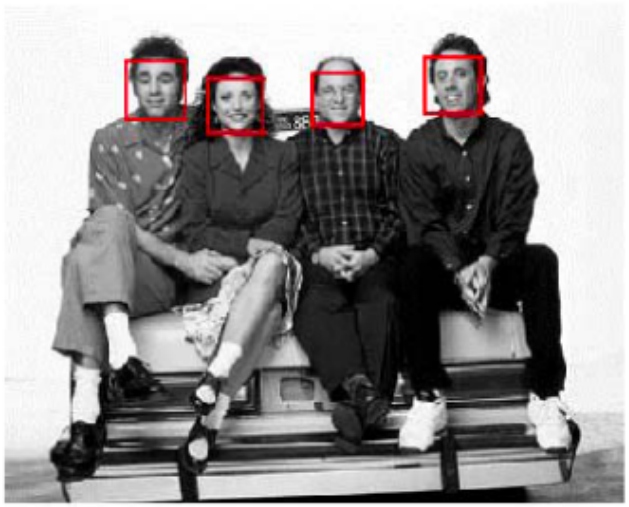


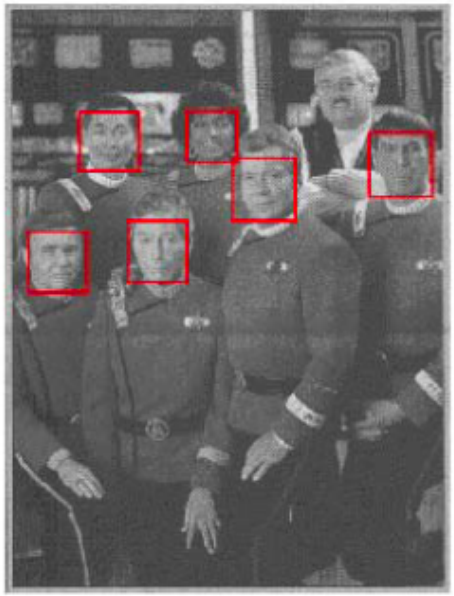
[K. Grauman]

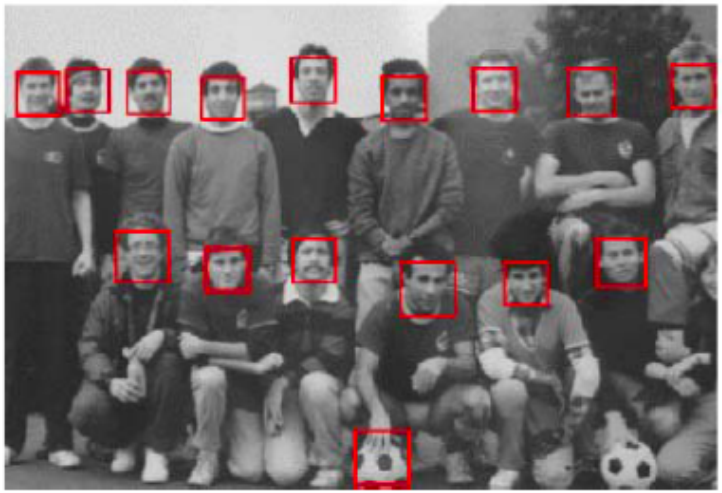
Viola Jones Summary

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  1. Integral images for fast feature evaluation
  2. Boosting for feature selection
  3. Attentional cascade of classifiers for fast rejection of non-face windows

Results

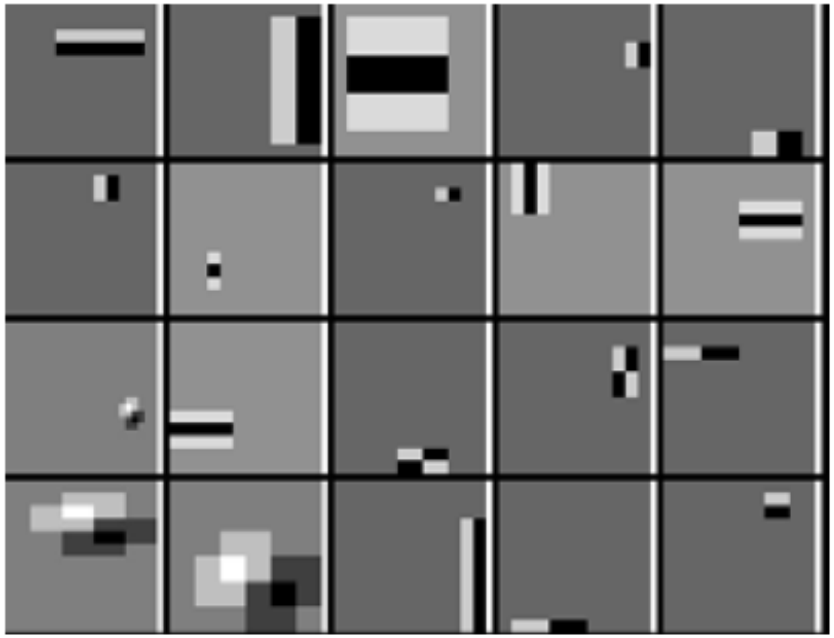






Results

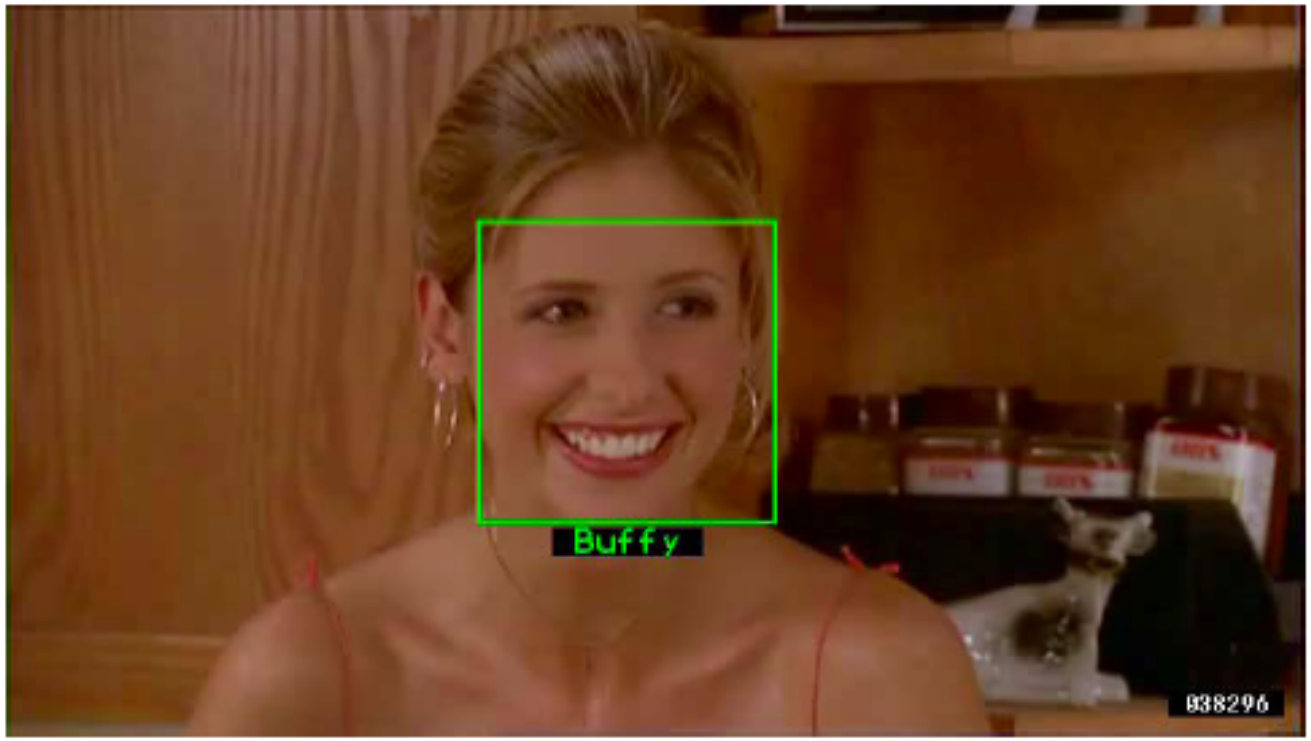
Also works for profile faces



[Source: K. Grauman]

Example using Viola-Jones detector

- Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles, [Everingham et al, BMVC 2006]

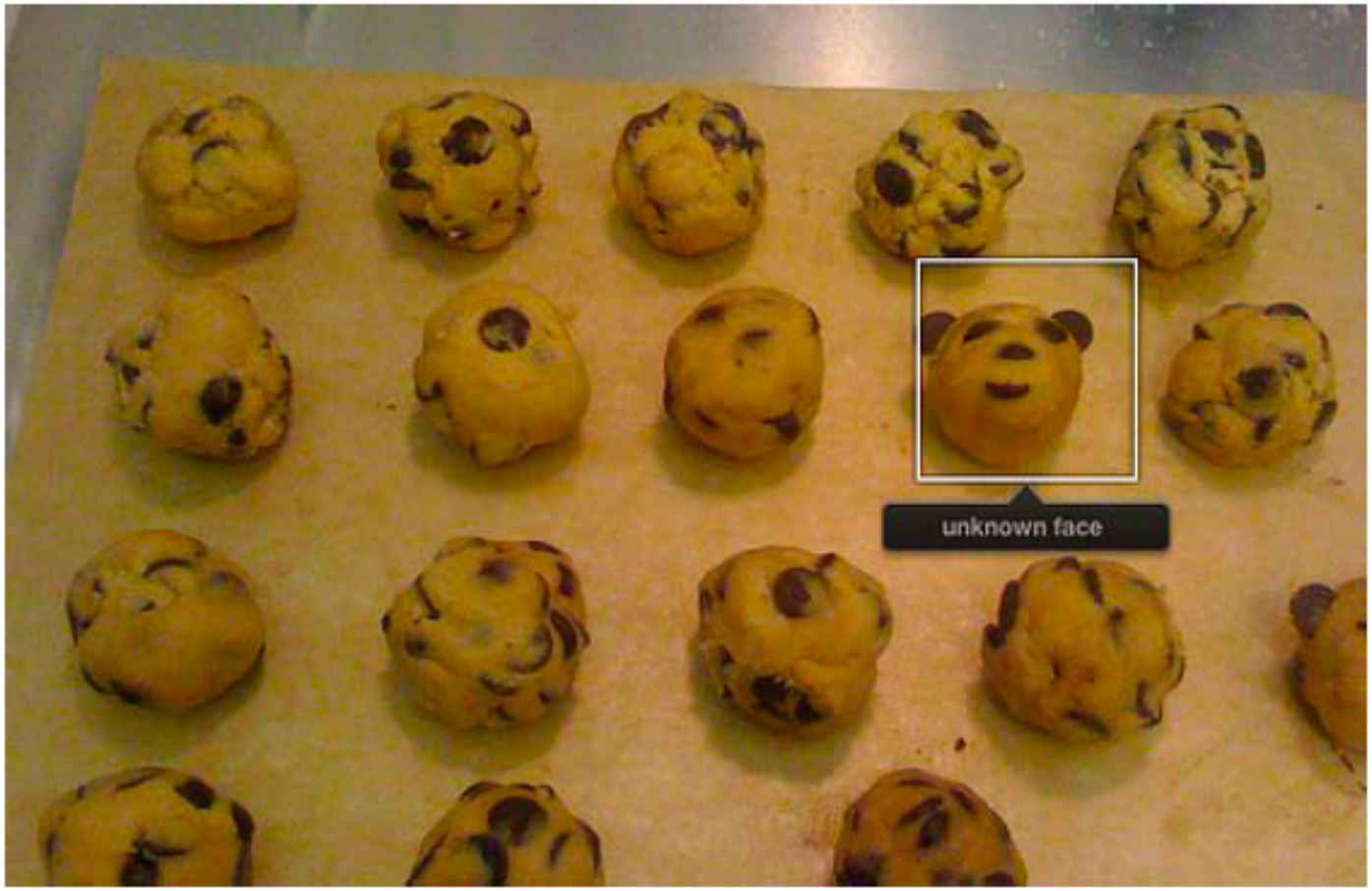


[Source: K. Grauman]

Commercial Applications

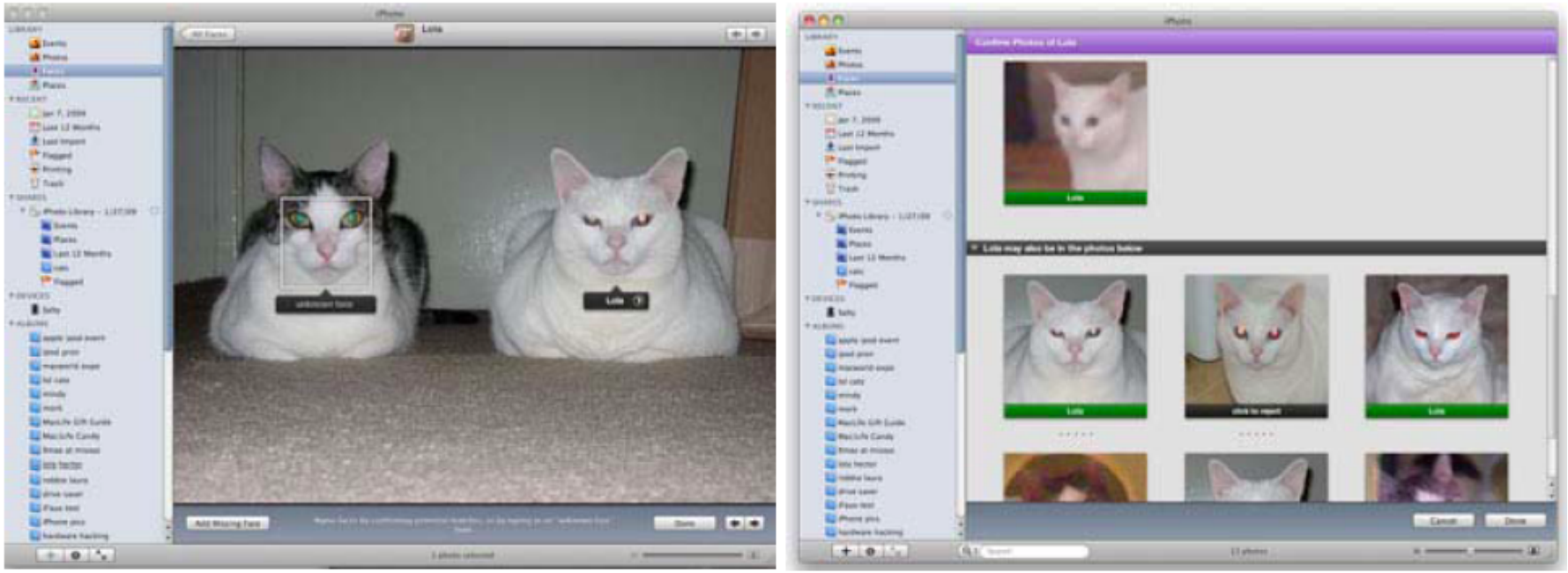
[Source: S. Lazebnik]

Interesting cases....



[Source: S. Lazebnik]

Can even recognize cats

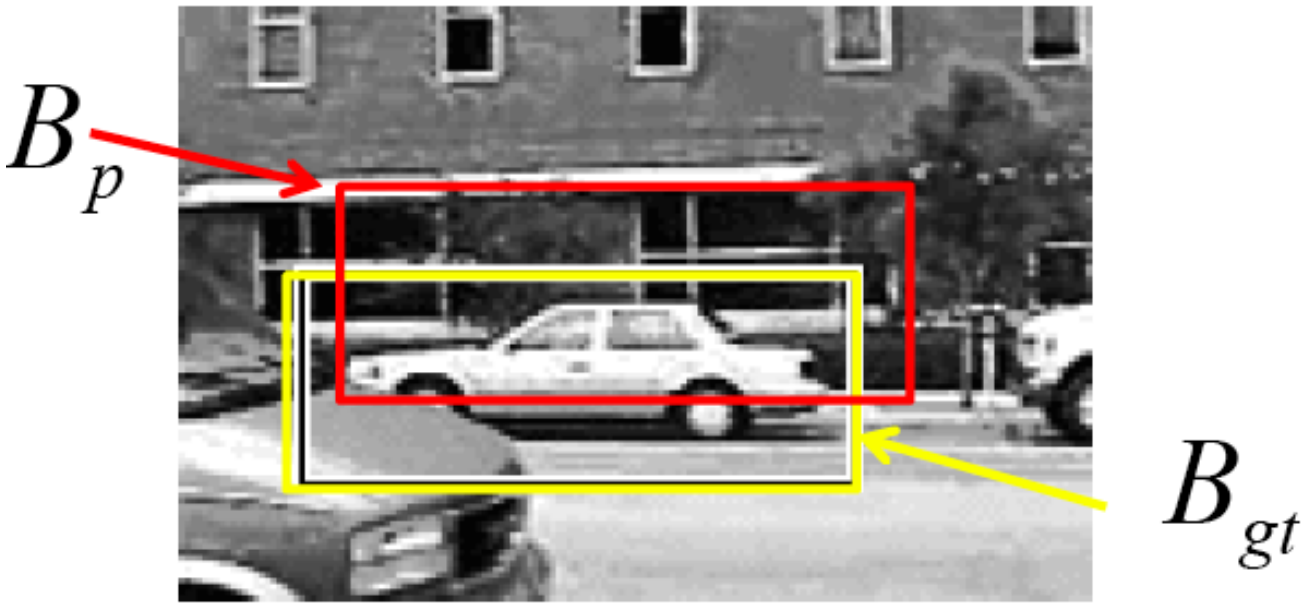


[Source: S. Lazebnik]

Scoring a sliding window detector

- Detection is correct if the intersection of the bounding boxes, divided by their union, is $> 50\%$.

$$a_0 = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$



[Source: K. Grauman]

Performance

- If the detector can produce a confidence score on the detections, then we can plot the rate of true vs. false positives as a threshold on the confidence is varied.
- Plot precision-recall curves
- Plot ROC curves.

Window-based detection

Strengths

- Simple detection protocol to implement
- Good feature choices critical
- Past successes for certain classes

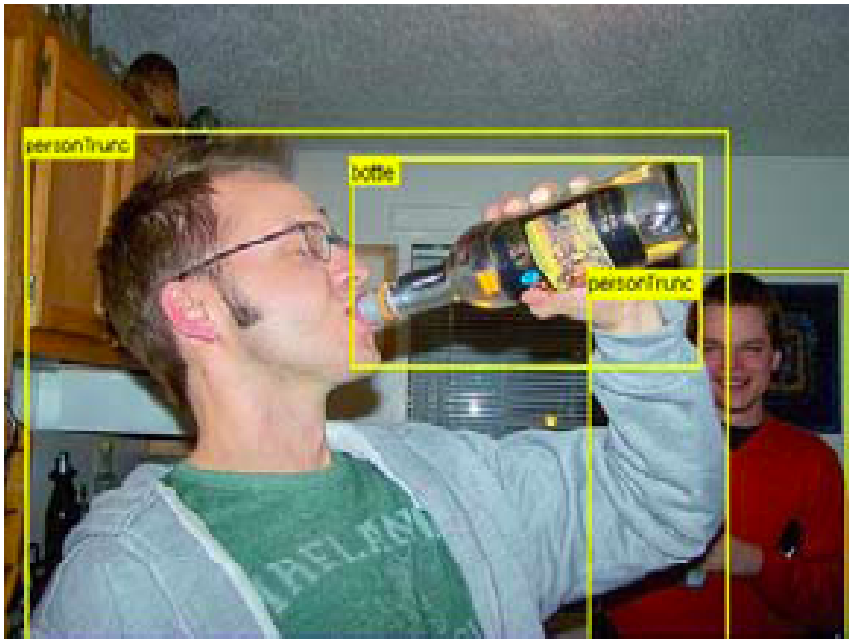
Limitations

- High computational complexity: For example: 250,000 locations x 30 orientations x 4 scales =30,000,000 evaluations!
- If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

[Source: K. Grauman]

More Limitations ...

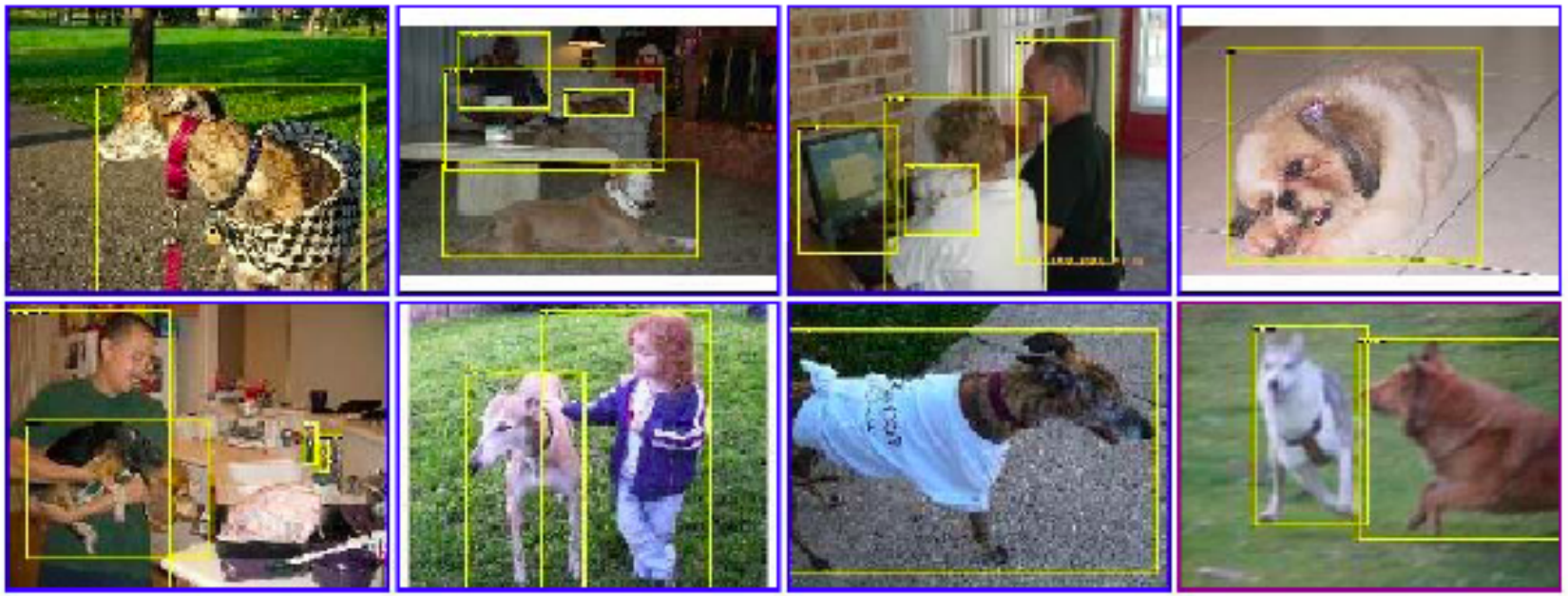
- Not all objects are box shaped: Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint



[Source: K. Grauman]

More Limitations

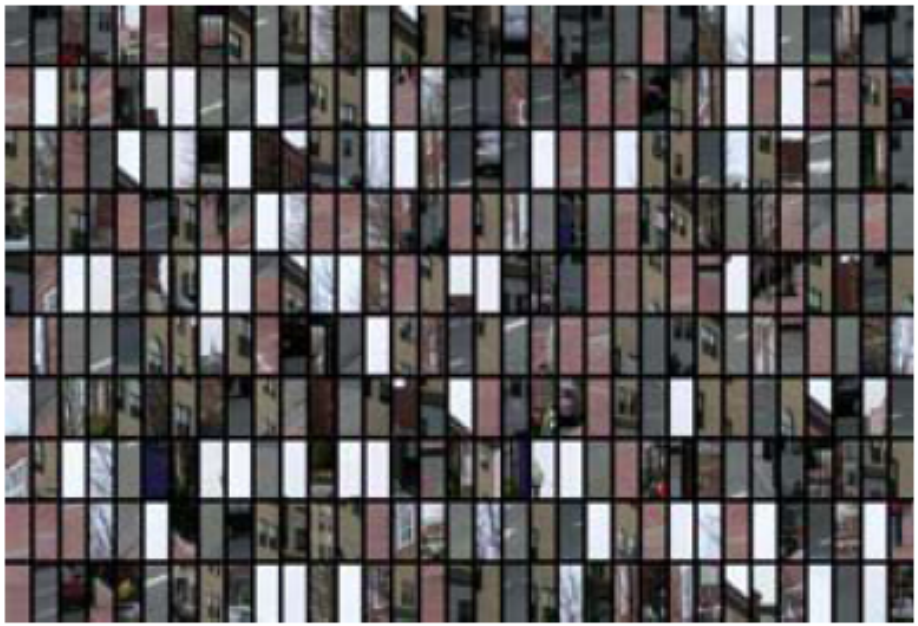
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



[Source: K. Grauman]

Even more...

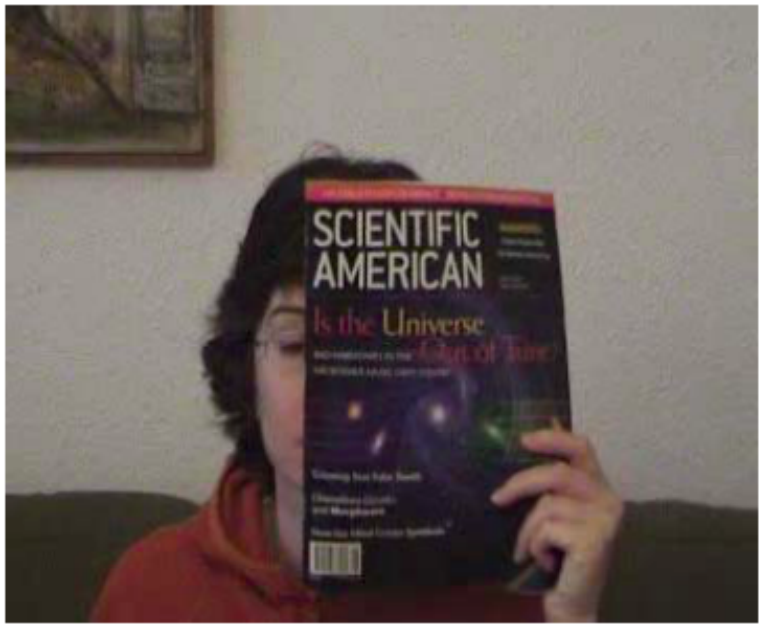
- If considering windows in isolation, context is lost



[Source: K. Grauman]

And more...

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions



[Source: K. Grauman]

Summary

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Discriminative classifiers for window-based representations:
  - Boosting: Viola-Jones face detector example
  - Nearest neighbors: Scene recognition example
  - Support vector machines: HOG person detection example
- Pros and cons of window-based detection

[K. Grauman]