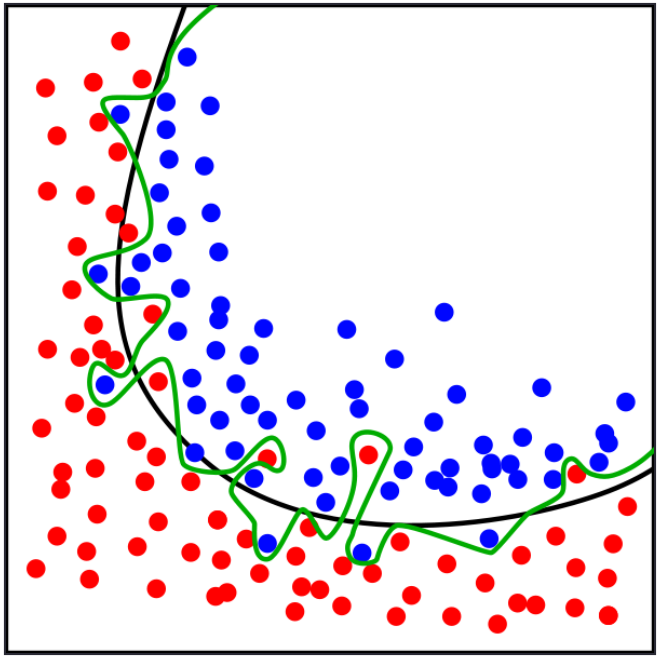
CS 4495 Computer Vision
Classification 2

Aaron Bobick

School of Interactive Computing

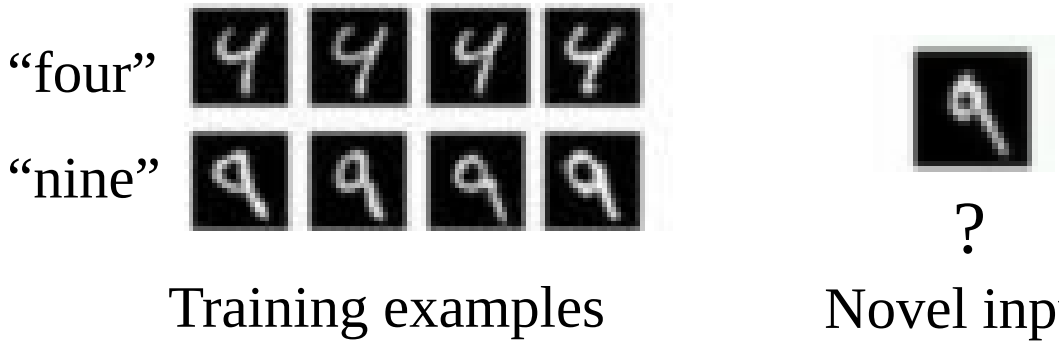


Administrivia

- PS5 (still) due on Friday Nov 14, 11:55pm
  - Remember, MSCS and undergrad are on different curves
  - We will not be unfair
- Hopfully PS6 out Friday Nov 14, due Nov 23rd
- Reminder: Problem set resubmission policy:
  - Full questions only
  - Be email to me and the TAs.
  - You get 50% credit to replace whatever you got last time on that question.
  - Must be submitted by: DEC 1. NO EXCEPTIONS.

Last time: Supervised classification

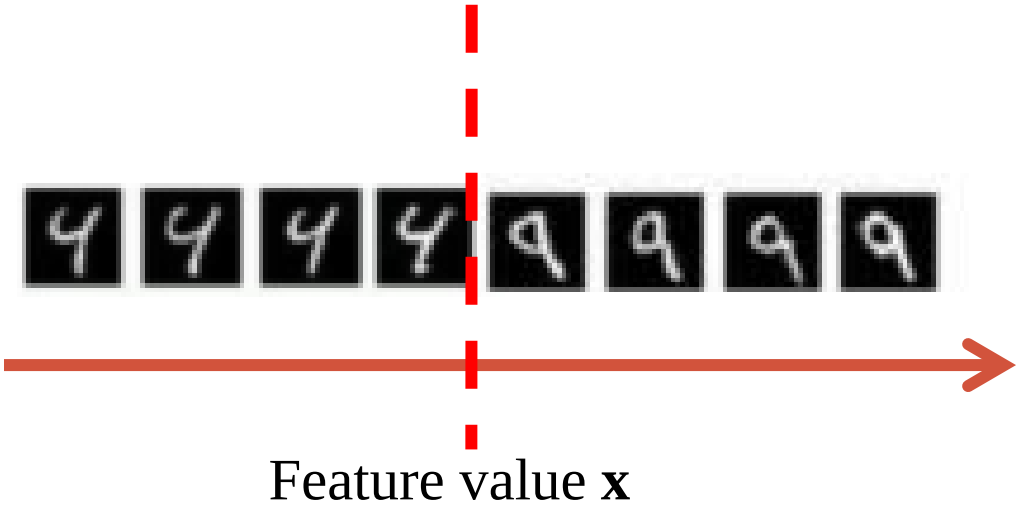
- Given a collection of *labeled* examples, come up with a function that will predict the labels of new examples.



Supervised classification

- Since we know the desired labels of training data, we want to minimize the expected misclassification
- Two general strategies
  - Use the training data to build representative probability model; separately model class-conditional densities and priors (***Generative***)
  - Directly construct a good decision boundary, model the posterior (***Discriminative***)

Supervised classification: minimal risk



Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

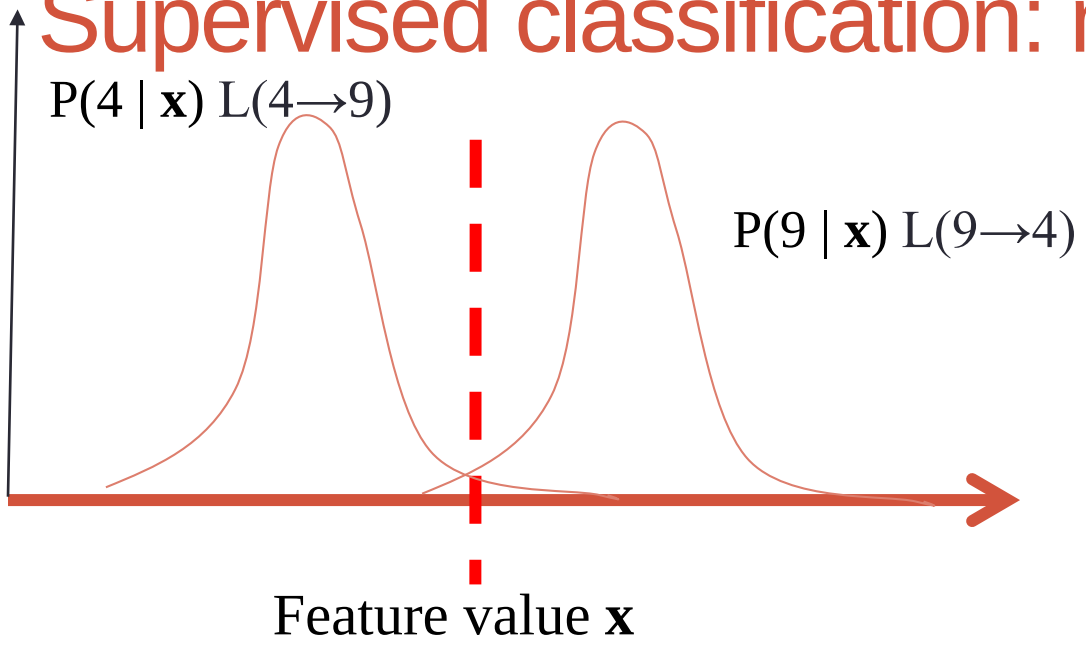
So, best decision boundary is at point **x** where

$$P(\text{class is } 9 \mid \mathbf{x})\, L(9 \rightarrow 4) = \mathrm{P}(\text{class is } 4 \mid \mathbf{x}) L(4 \rightarrow 9)$$

To classify a new point, choose class with lowest expected loss; i.e., choose "four" if

$$P(4 \mid \mathbf{x}) L(4 \rightarrow 9) > P(9 \mid \mathbf{x}) L(9 \rightarrow 4)$$

Supervised classification: minimal risk



Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

So, best decision boundary is at point **x** where

$$P(\text{class is } 9 \mid \mathbf{x})\, L(9 \to 4) = P(\text{class is } 4 \mid \mathbf{x}) L(4 \to 9)$$

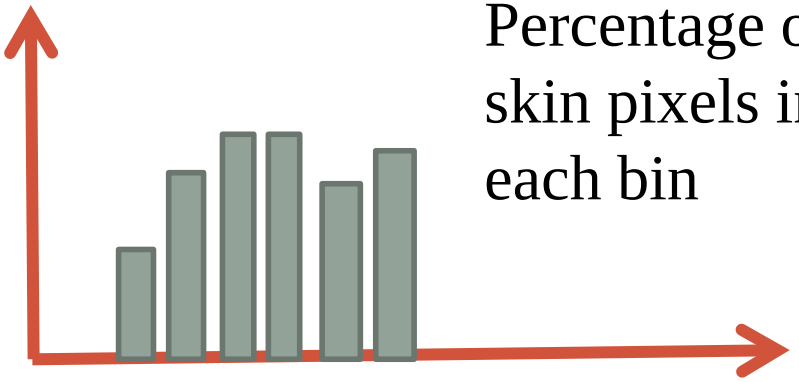
To classify a new point, choose class with lowest expected loss; i.e., choose "four" if

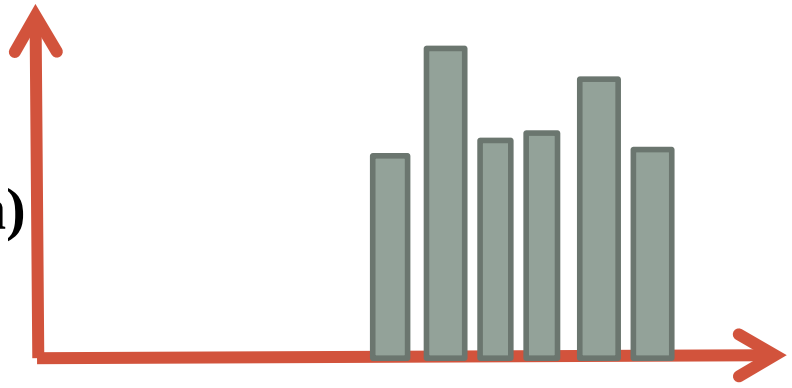
$$P(4 \mid \mathbf{x}) L(4 \to 9) > P(9 \mid \mathbf{x}) L(9 \to 4)$$

How to evaluate these probabilities?

Example: learning skin colors

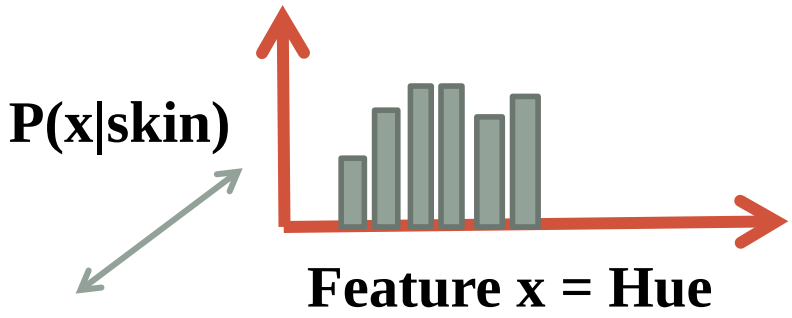
- We can represent a class-conditional likelihood density using a histogram (a "non-parametric" distribution)





Kristen Grauman

Bayes rule



$$P(skin \mid x) = \frac{P(x \mid skin)P(skin)}{P(x)}$$

posterior — likelihood — prior

$$P(skin \mid x) \propto P(x \mid skin)P(skin)$$

Where does the prior come from?

Why even use a prior?

Source: Steve Seitz

Example: classifying skin pixels

Now for every pixel in a new image, we can estimate probability that it is generated by skin.

Brighter pixels → higher probability of being skin

Classify pixels based on these probabilities

- if $p(\text{skin}|\boldsymbol{x}) > \theta$, classify as skin
- if $p(\text{skin}|\boldsymbol{x}) < \theta$, classify as not skin

Some challenges for generative models

Generative approaches were some of the first methods in *pattern recognition*.

- Easy to model analytically and could be done with modest amounts of moderate dimensional data.

Some challenges for generative models

But for the modern world there are some liabilities:

- Many signals are *high-dimensional* and *representing the complete density of class is data-hard*.
- In some sense, we don't care about modeling the classes, *we only care about making the right decisions*.
  - *Model the hard cases- the ones near the boundaries!!*
- We don't typically know which features of instances actually *discriminate* between classes.

So…

- If we have only a fixed number of label types…
- If what matters is getting the label right…
- If we're not sure which features/properties are even important in defining the classes

… then …

- We want to focus on ***discriminating*** between the class types.
- We want the machine to somehow ***learn*** the features that matter.

- This gets us to ***discriminative classification***.

Discriminative methods: assumptions

Going forward we're gong to make some assumptions

- There are a fixed number of known classes.
- Ample number of training examples of each class.
- Equal cost of making mistakes - what matters is getting the label right.
- Need to construct a representation of the instance but we don't know a priori what features are diagnostic of the class label.

Generic category recognition: basic framework

Train

- Build an object model – a ***representation***

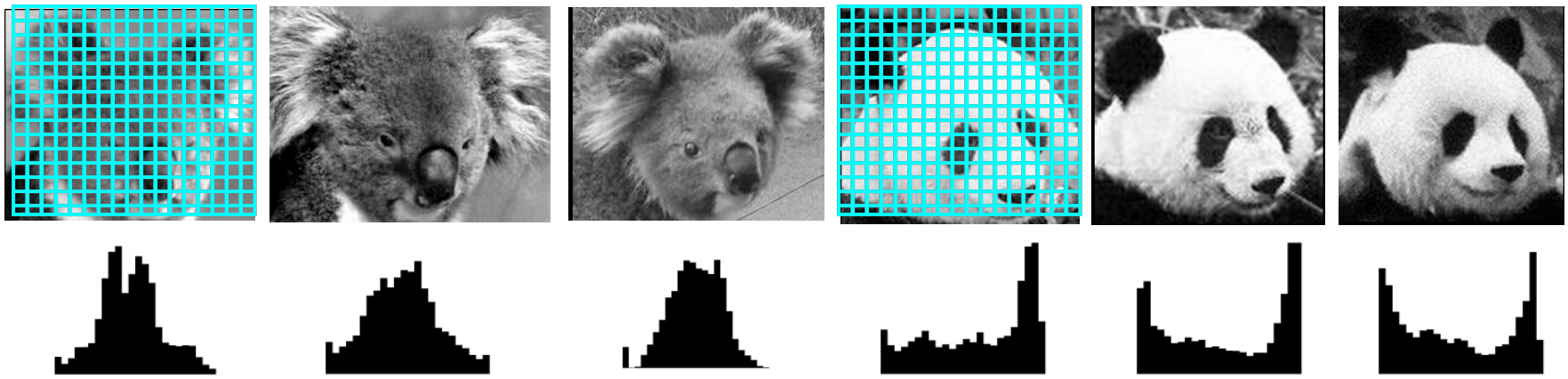
  Describe training instances (here images)

- Learn/train a ***classifier***

Test

- Generate candidates in new image
- *Score* the candidates

Window-based models

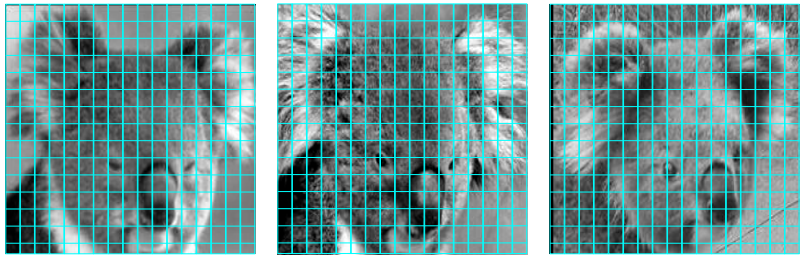


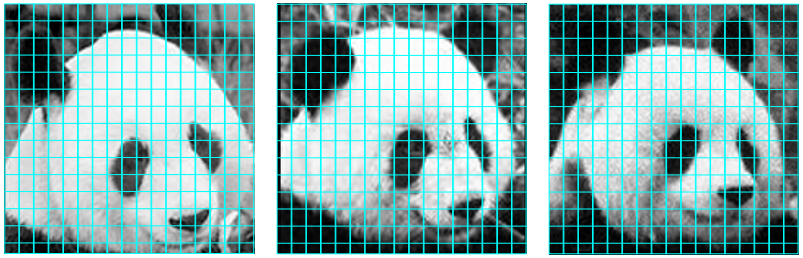
Simple holistic descriptions of image content

- **grayscale / color histogram**
- **vector of pixel intensities**

Window-based models

- **Pixel-based representations sensitive to small shifts**

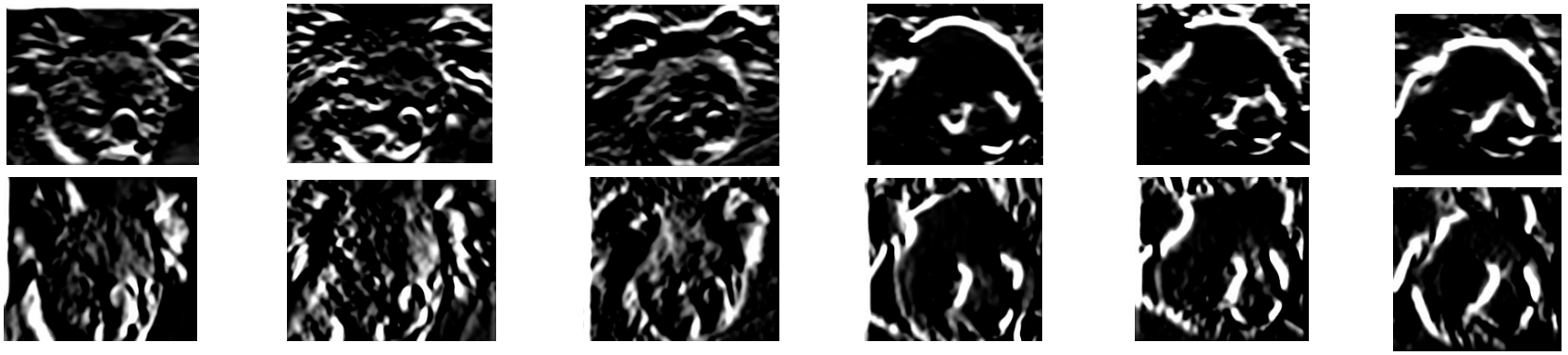




- **Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation**

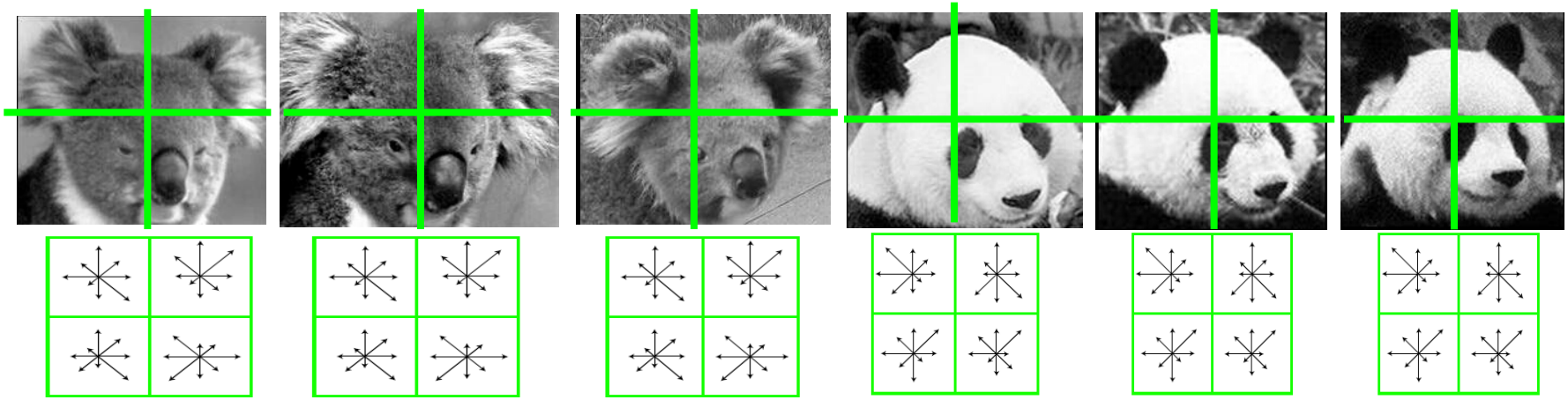
Window-based models

- **Consider edges, contours, and (oriented) intensity gradients**



Window-based models

- **Consider edges, contours, and (oriented) intensity gradients**



- **Summarize local distribution of gradients with histogram**
  - **Locally orderless: offers invariance to small shifts and rotations**
  - **Contrast-normalization: try to correct for variable illumination**

Generic category recognition: basic framework

Train

- Build an object model – a *representation*

  Describe training instances (here images)

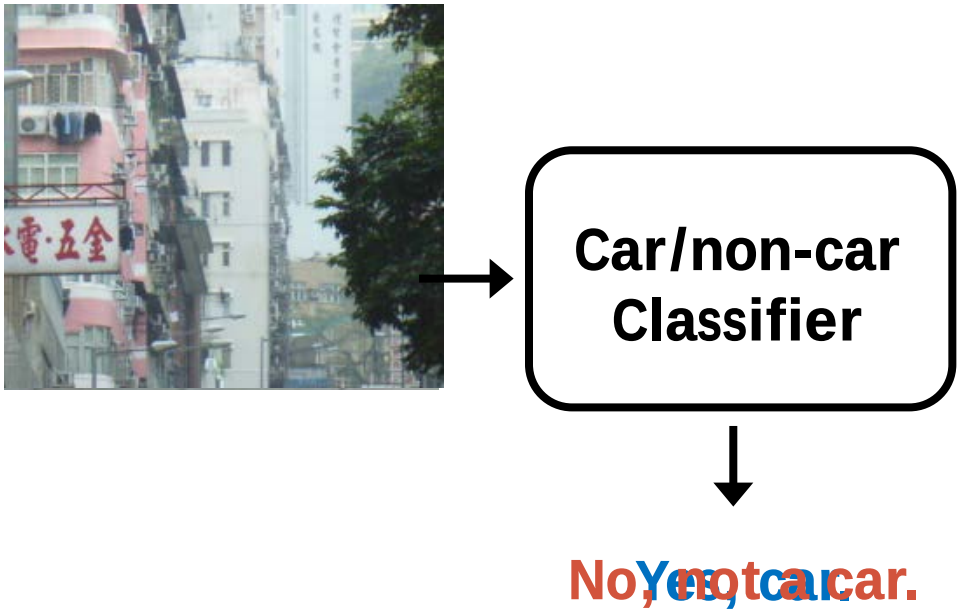
- Learn/train a ***classifier***

Test

- Generate candidates in new image
- *Score* the candidates

Window-based models

Given the representation, train a binary classifier



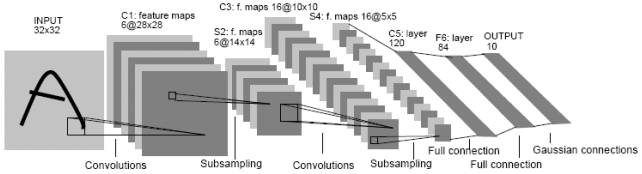
Discriminative classifier construction

Nearest neighbor

10^6 examples

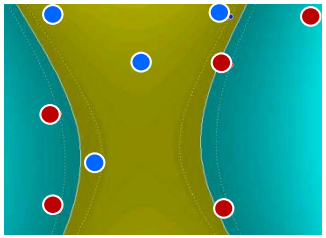
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



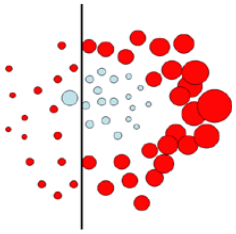
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



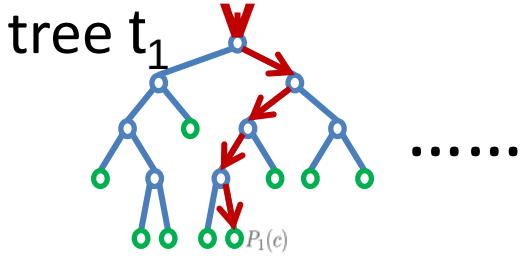
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

Random Forests



Breiman, 1984
Shotton, et al CVPR 2008

Slide adapted from Antonio Torralba

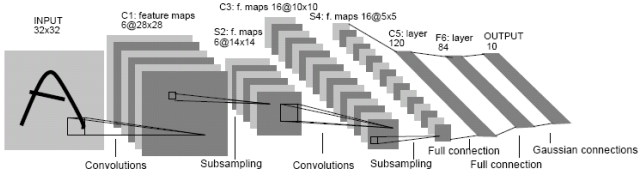
Discriminative classifier construction

Nearest neighbor

10^6 examples

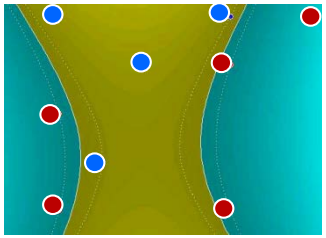
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



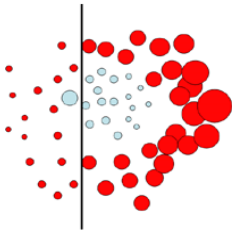
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



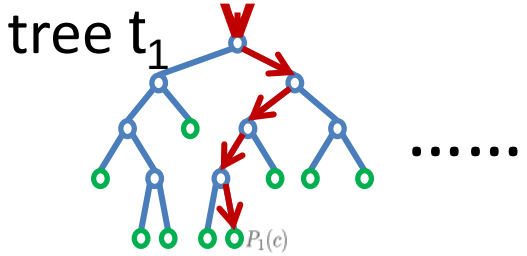
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

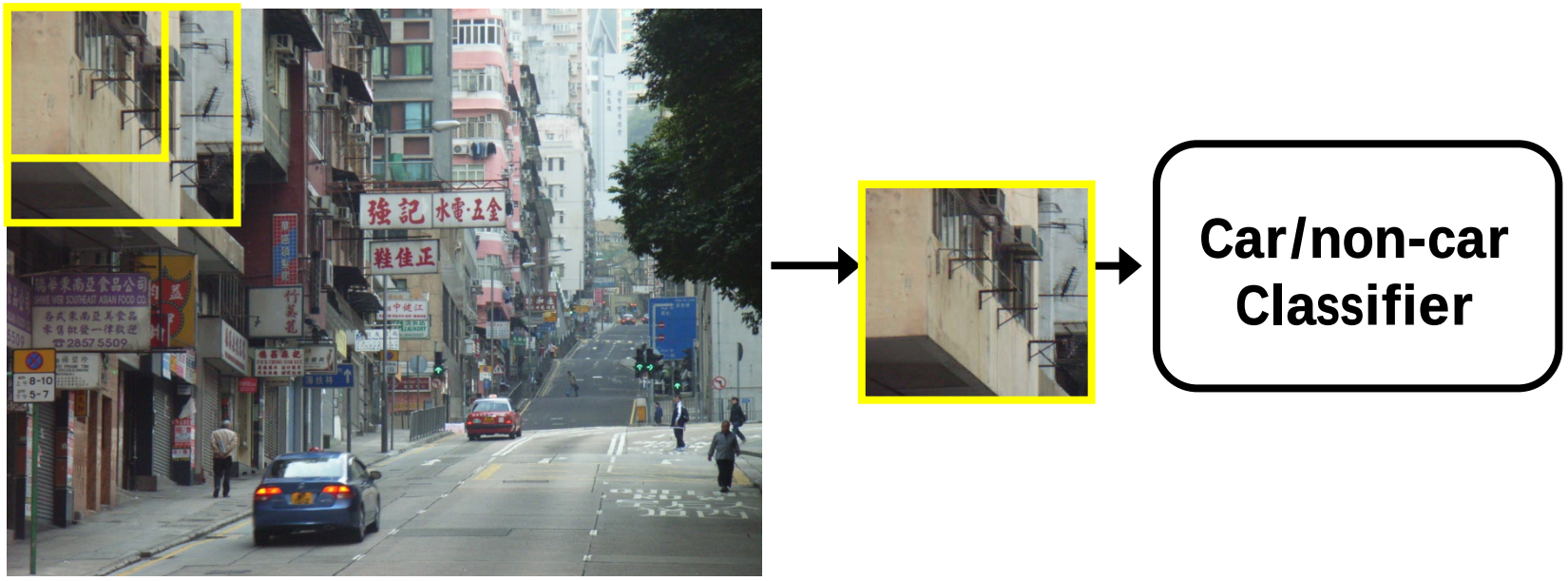
Random Forests



Breiman, 1984
Shotton, et al CVPR 2008

Slide adapted from Antonio Torralba

Window-based models
Generating and scoring candidates



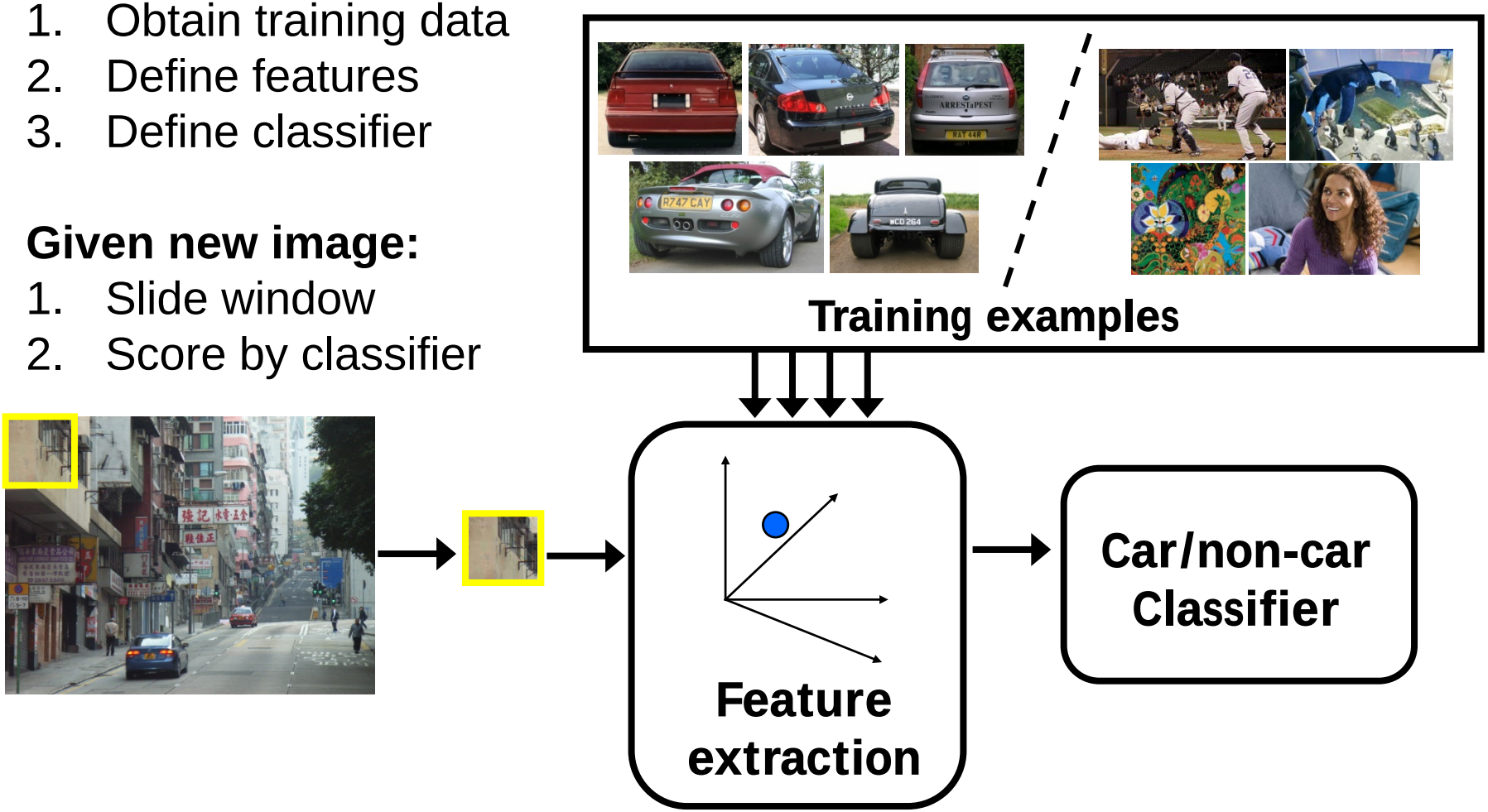
Window-based object detection:

Training:

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

Given new image:

1. Slide window
2. Score by classifier



Discriminative classification methods

Discriminative classifiers – find a division (surface) in feature space that separates the classes

Several methods

- Nearest neighbors
- Boosting
- Support Vector Machines

Discriminative classification methods

Discriminative classifiers – find a division (surface) in feature space that separates the classes

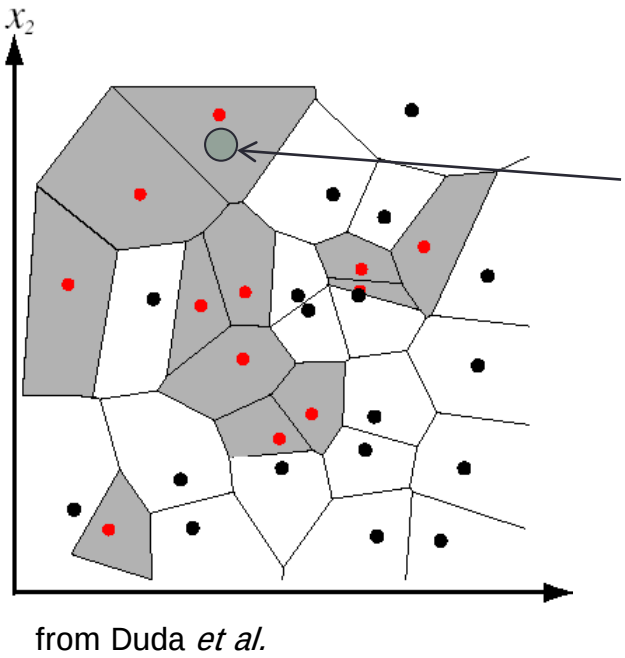
Several methods

- Nearest neighbors
- Boosting
- Support Vector Machines

Nearest Neighbor classification

- Assign label of nearest training data point to each test data point

Black = negative
Red = positive



from Duda *et al.*

Novel test example

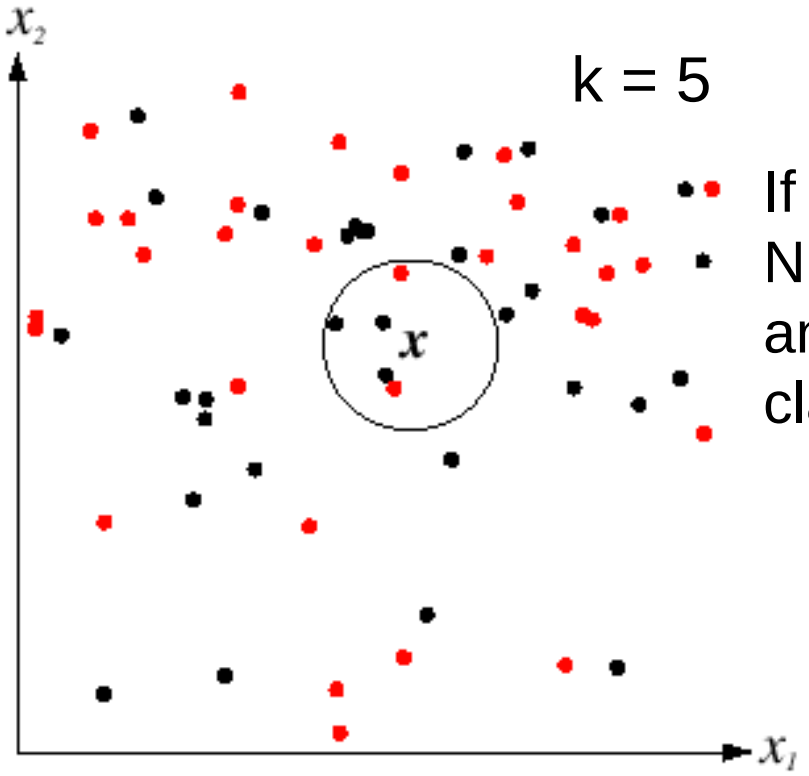
Closest to a positive example from the training set, so classify it as positive.

Voronoi partitioning of feature space
for 2-category 2D data

K-Nearest Neighbors classification

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify

Black = negative
Red = positive



If query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.

Source: D. Lowe

Discriminative classification methods

Discriminative classifiers – find a division (surface) in feature space that separates the classes

Several methods

- Nearest neighbors
- Boosting
- Support Vector Machines

Discriminative classification methods

Discriminative classifiers – find a division (surface) in feature space that separates the classes

Several methods

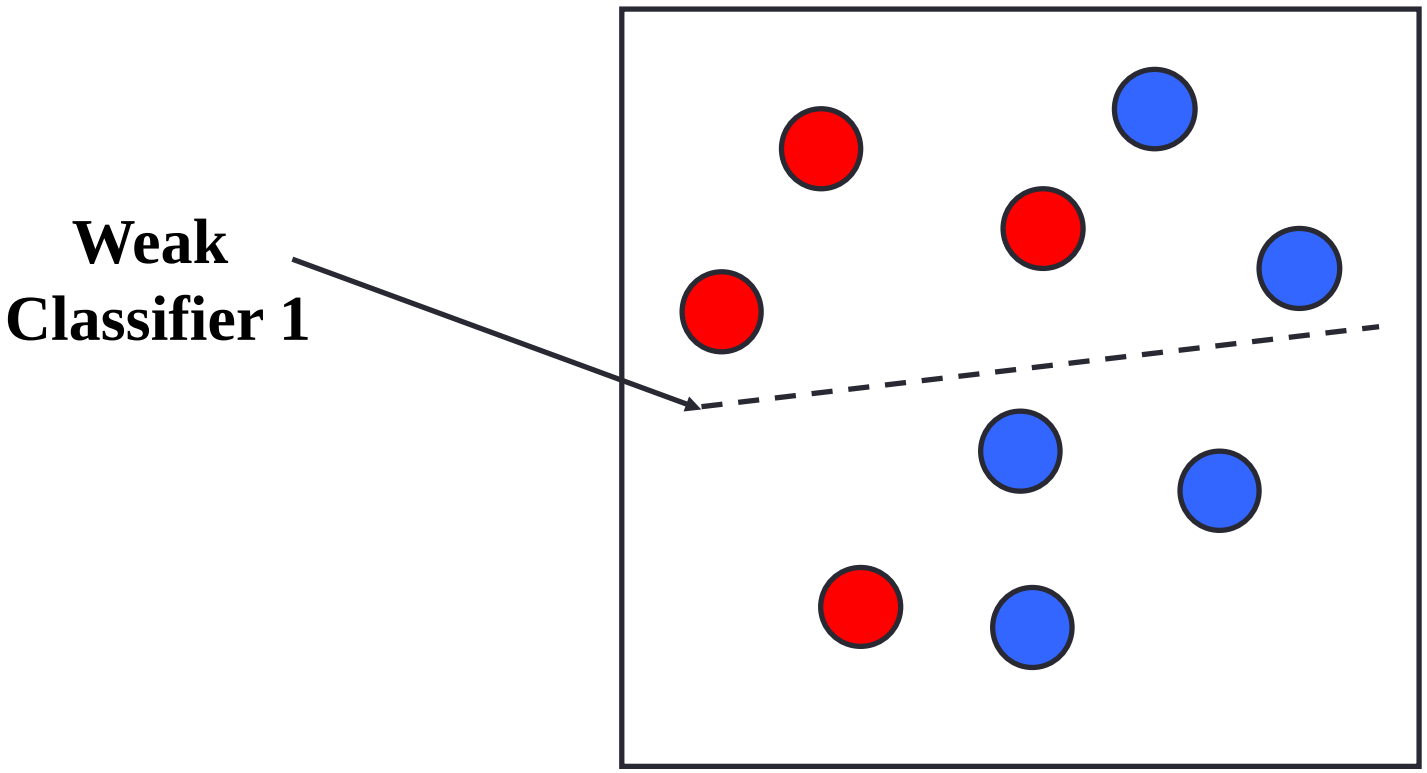
- Nearest neighbors
- Boosting
- Support Vector Machines

Boosting: Training method

- 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)

Slide credit: Lana Lazebnik

Boosting intuition



Slide credit: Paul Viola

Boosting: Training method

- 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

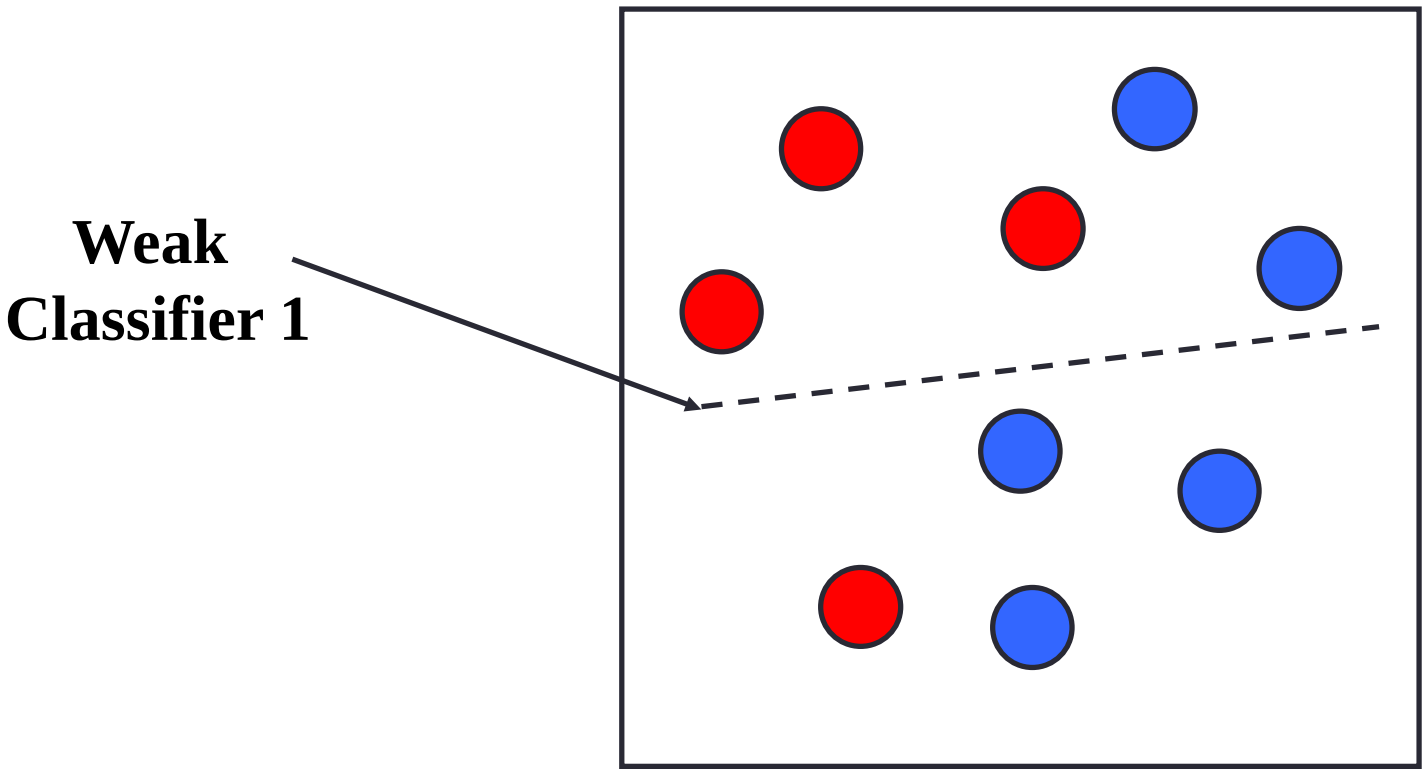
Slide credit: Lana Lazebnik

Boosting: Training method

- 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*

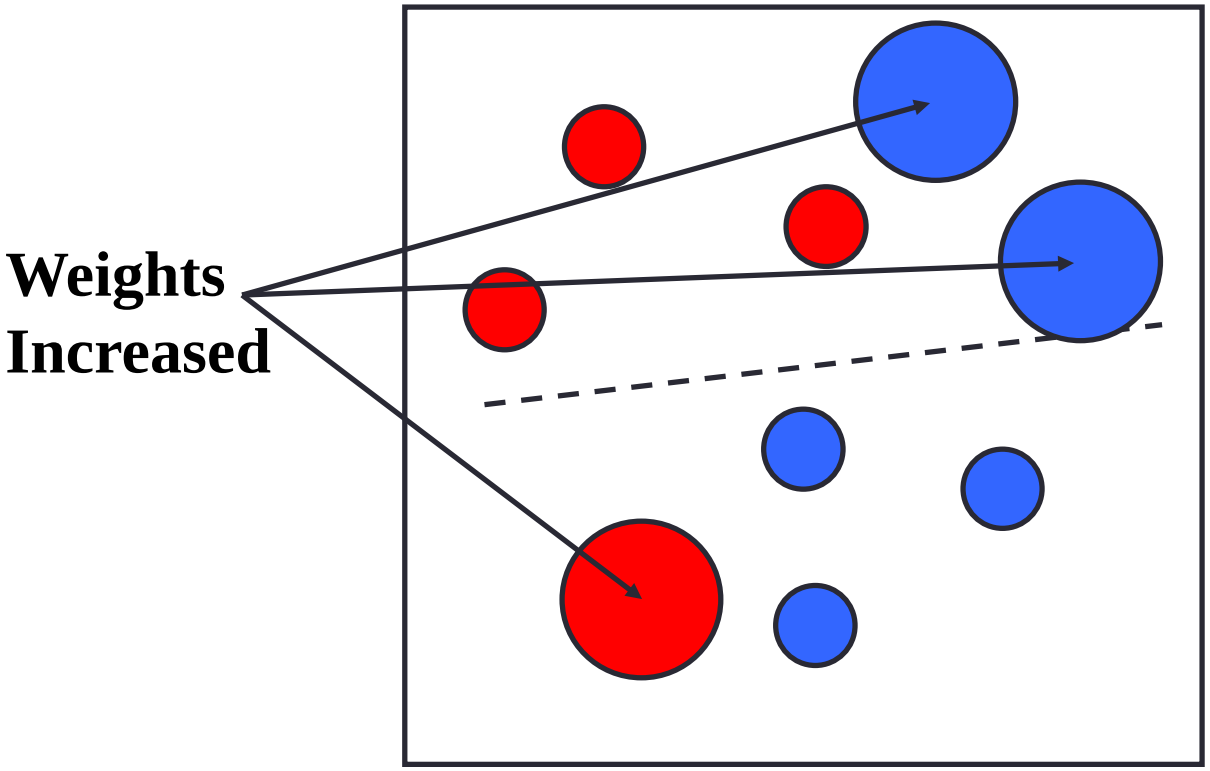
Slide credit: Lana Lazebnik

Boosting intuition

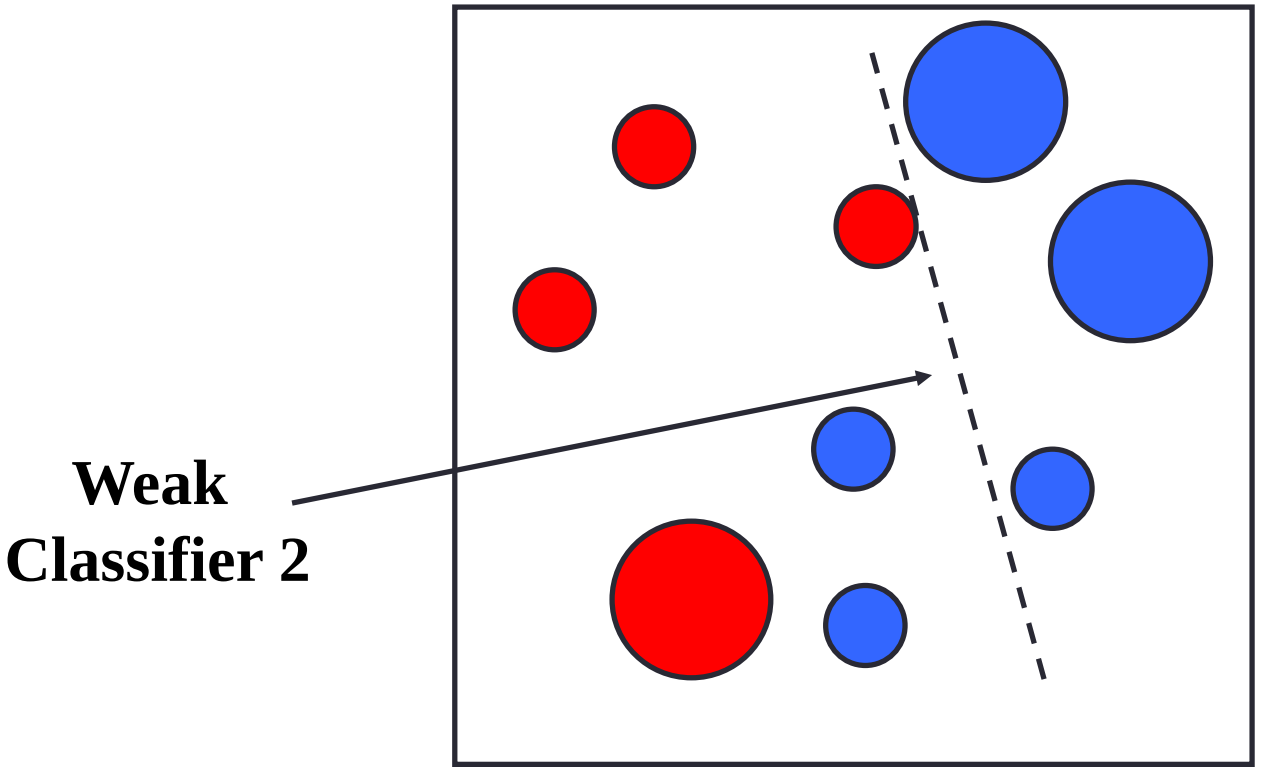


Slide credit: Paul Viola

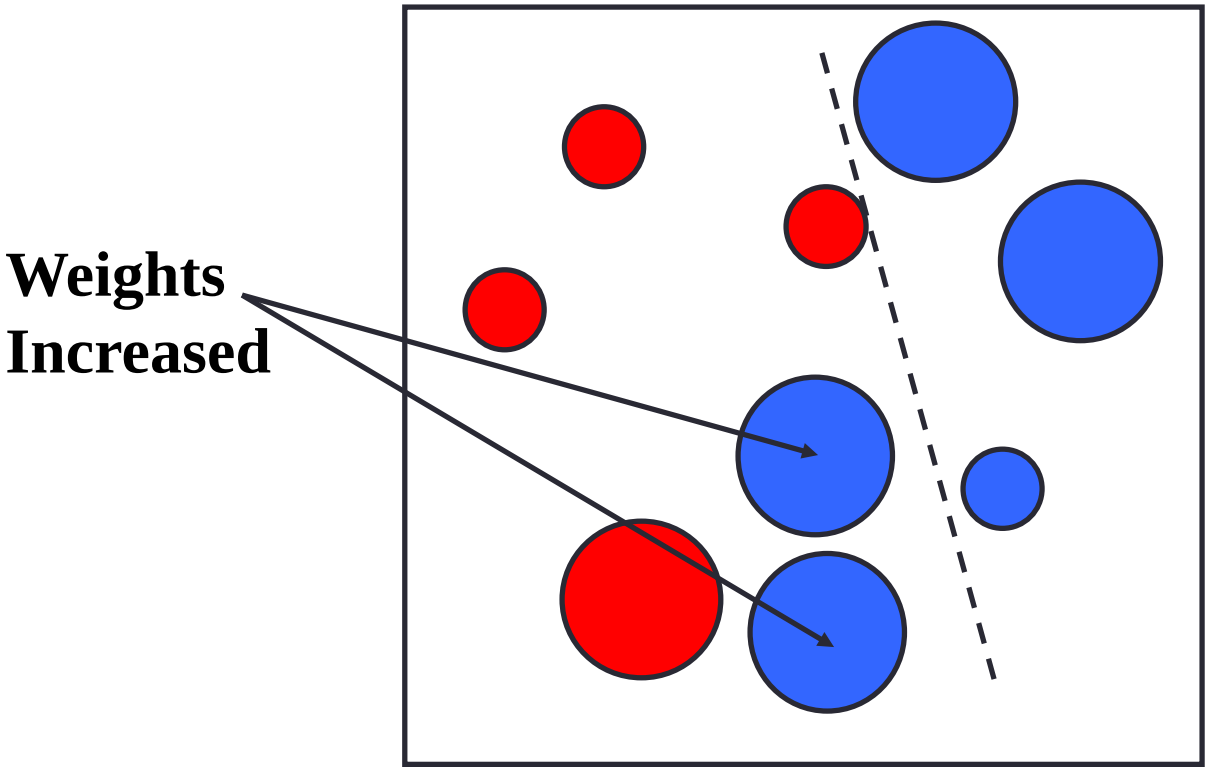
Boosting illustration



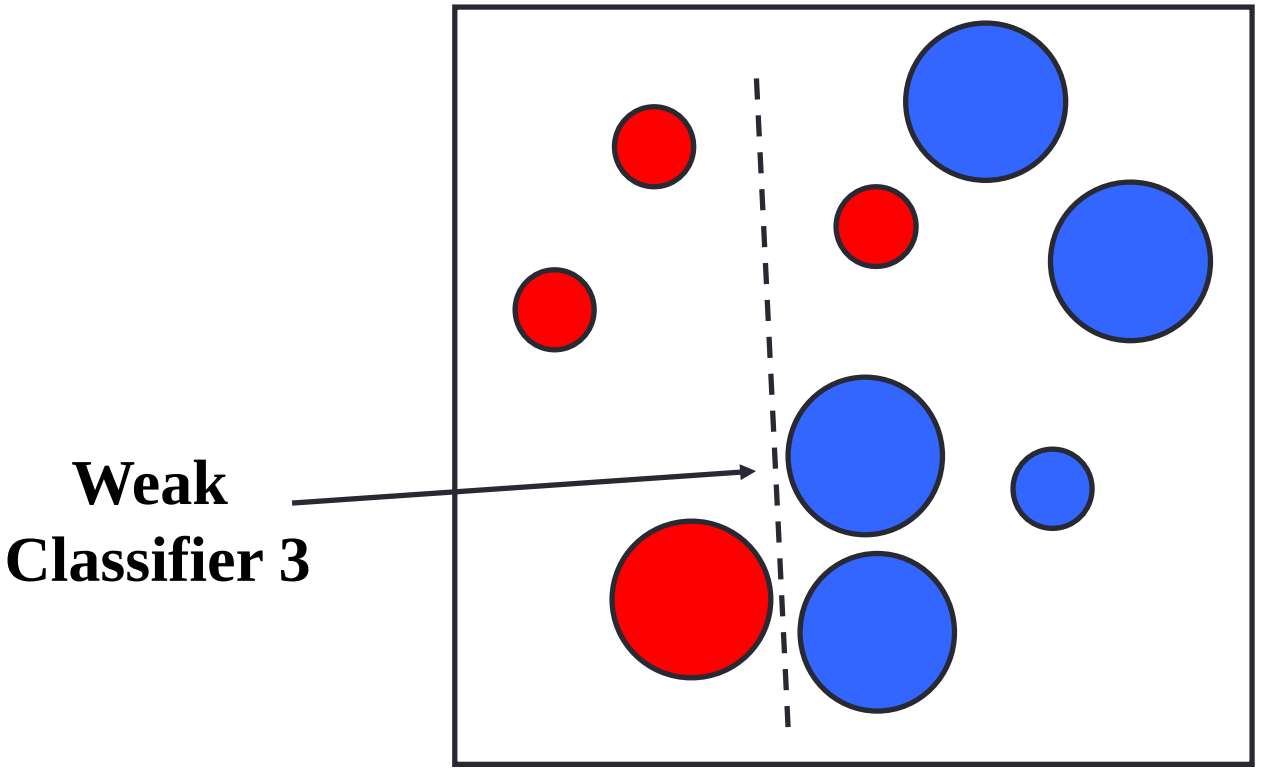
Boosting illustration



Boosting illustration

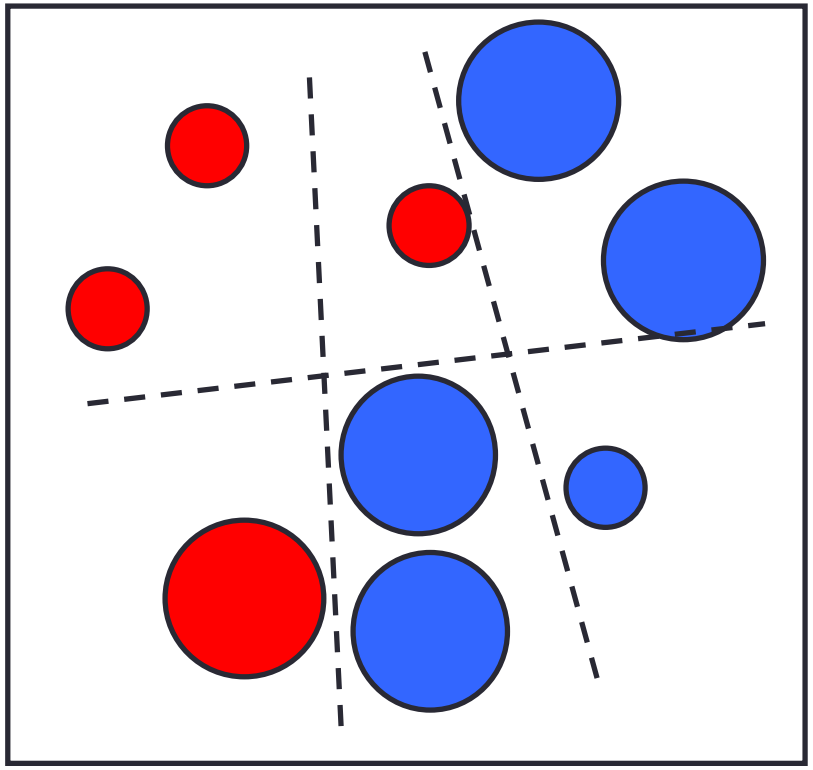


Boosting illustration



Boosting illustration

Final classifier is a combination of weak classifiers



Boosting: Training method

- 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)

Slide credit: Lana Lazebnik

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 vi-

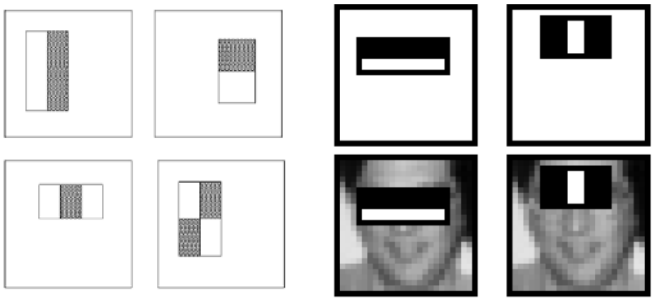
tected at 15 frames per second on a conventional 700 MHz Intel Pentium III. In other face detection systems, auxiliary information, such as image differences in video sequences,

Viola-Jones face detector

Main ideas:

- Represent local texture with efficiently computable "rectangular" features within window of interest
- 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

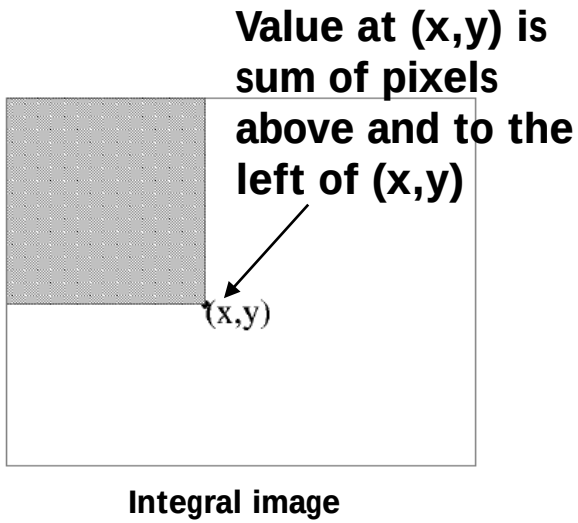
Viola-Jones detector: features



"Rectangular" filters

Feature output is difference between adjacent regions

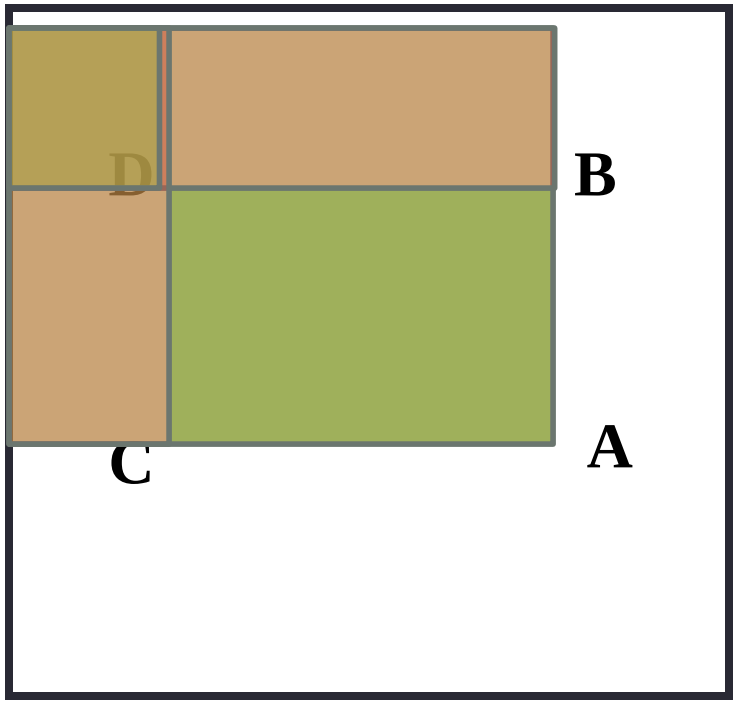
Efficiently computable with integral image: any sum can be computed in constant time.



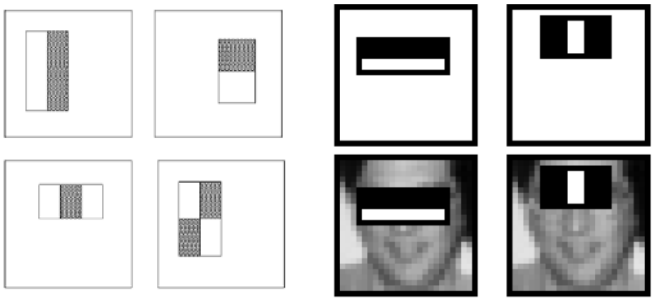
Computing sum within a rectangle

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!



Viola-Jones detector: features

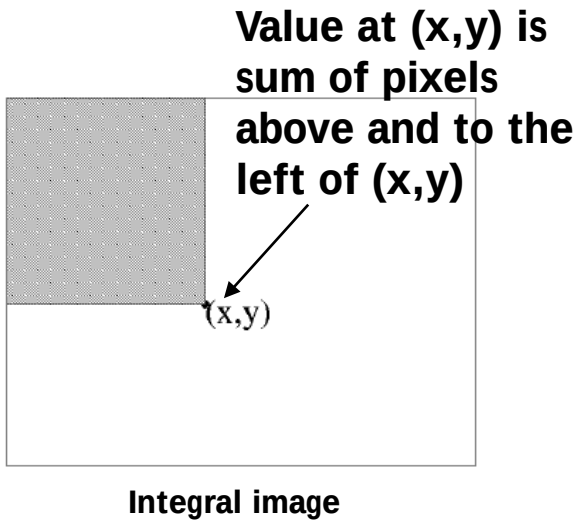


"Rectangular" filters

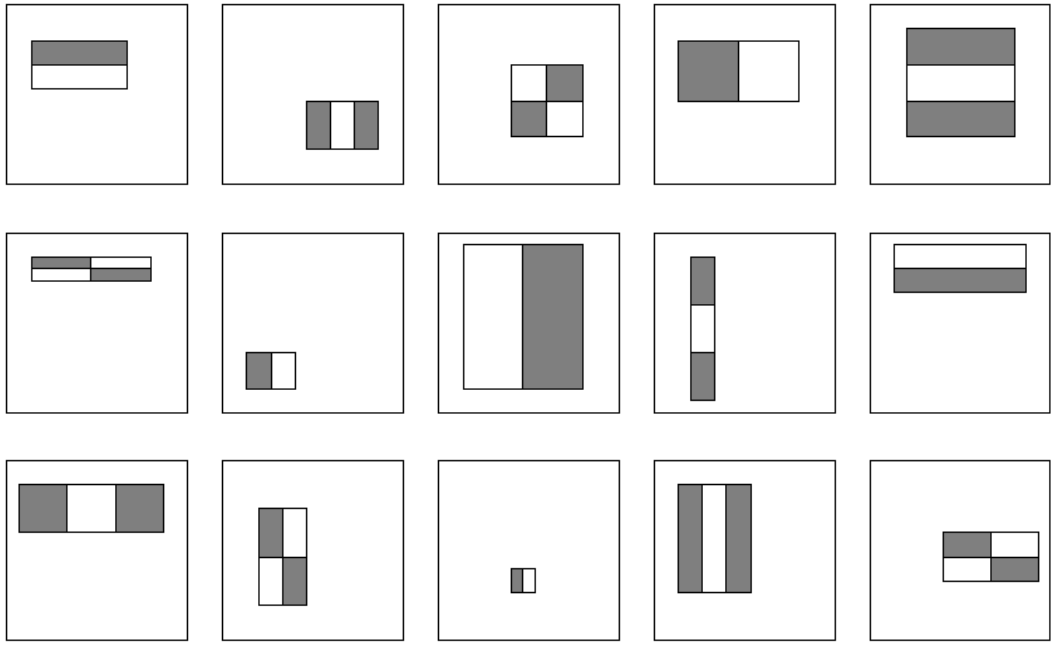
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost



Viola-Jones detector: features



Considering all possible filter parameters: position, scale, and type:

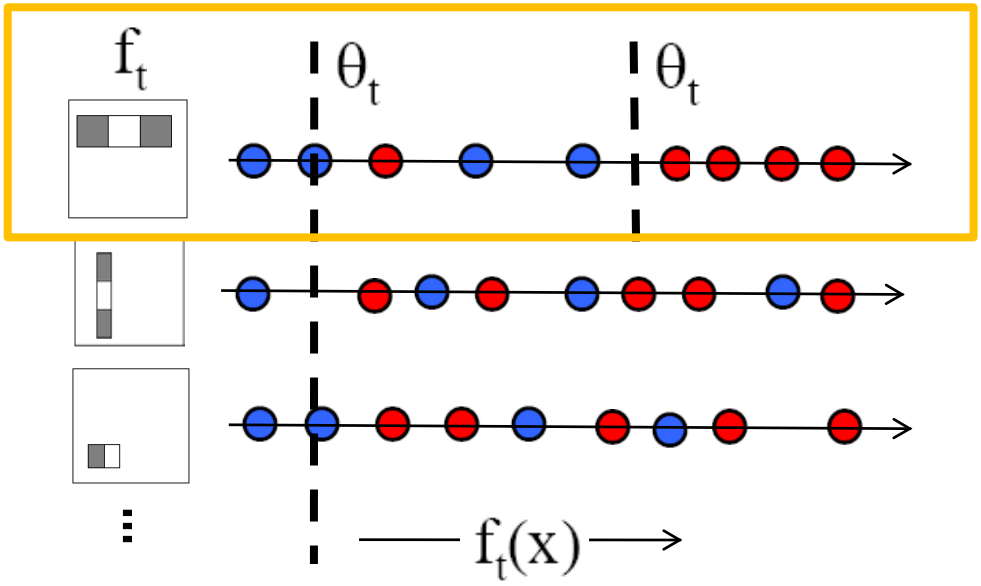
180,000+ possible features associated with each 24 x 24 window

Which subset of these features should we use to determine if a window has a face?

Use AdaBoost both to select the informative features and to form the classifier

Viola-Jones detector: AdaBoost

- **Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.**



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

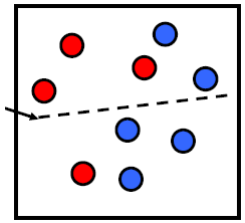
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

AdaBoost Algorithm

- 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 $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

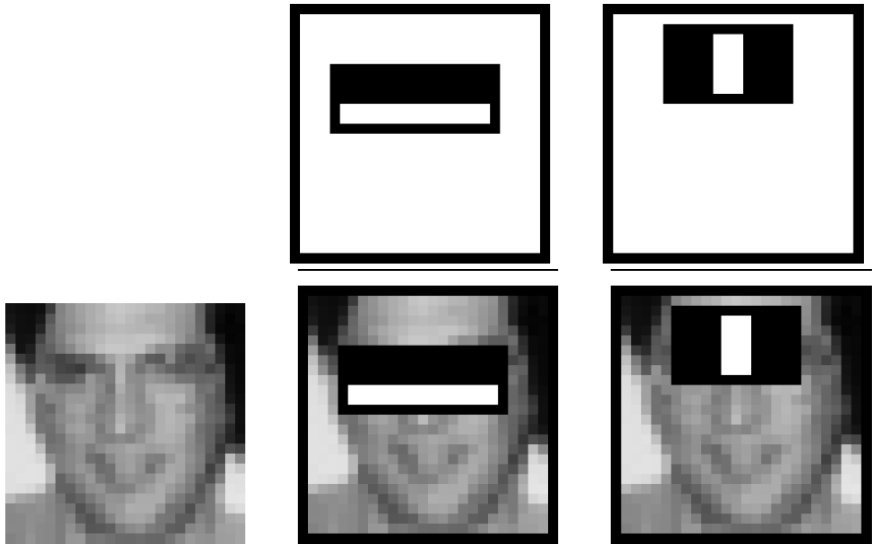
Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

Viola-Jones Face Detector: Results



First two features selected

Viola-Jones face detector

Main ideas:

- Represent local texture with efficiently computable "rectangular" features within window of interest
- 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

Viola-Jones face detector

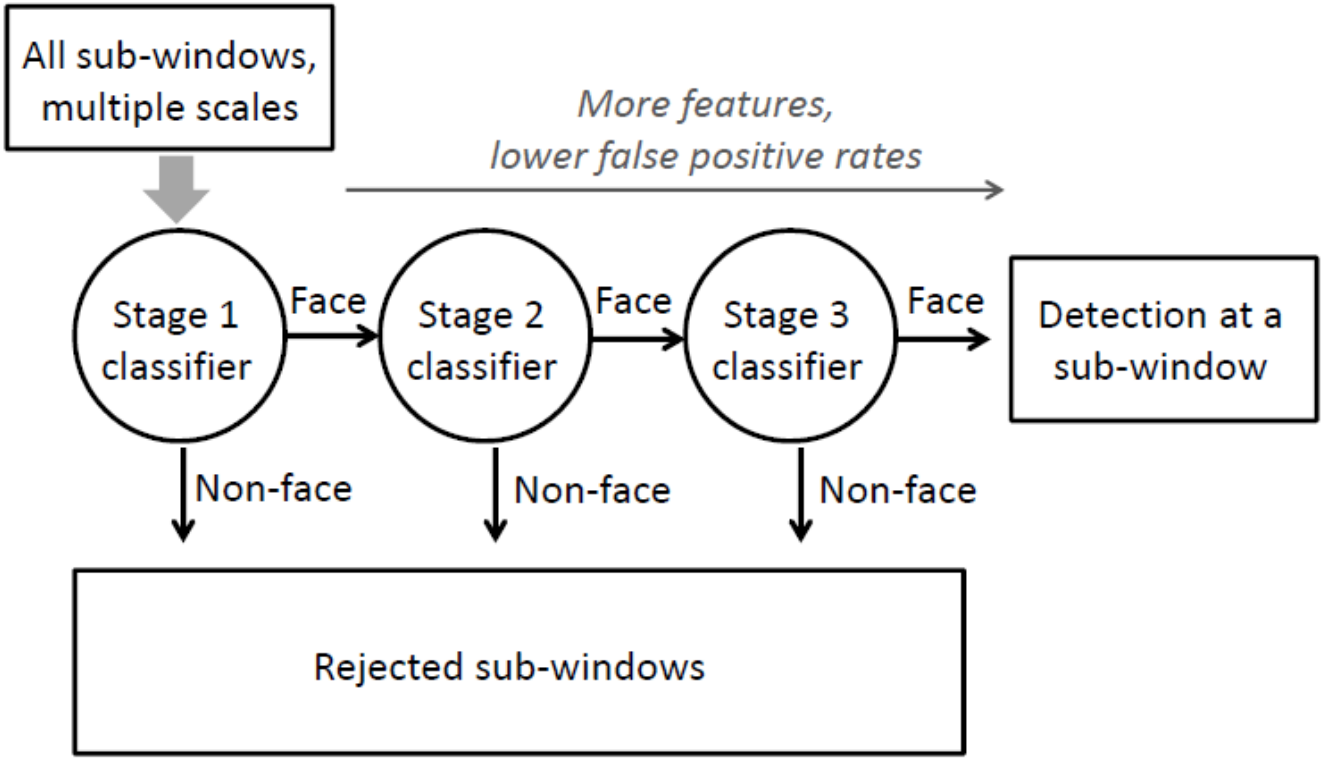
Main ideas:

- Represent local texture with efficiently computable "rectangular" features within window of interest
- 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**

2nd idea: Cascade…

- Key insight: *almost every where is a non-face.*
  - So… detect non-faces more quickly than faces.
  - And if you say it's not a face, be sure and move on.

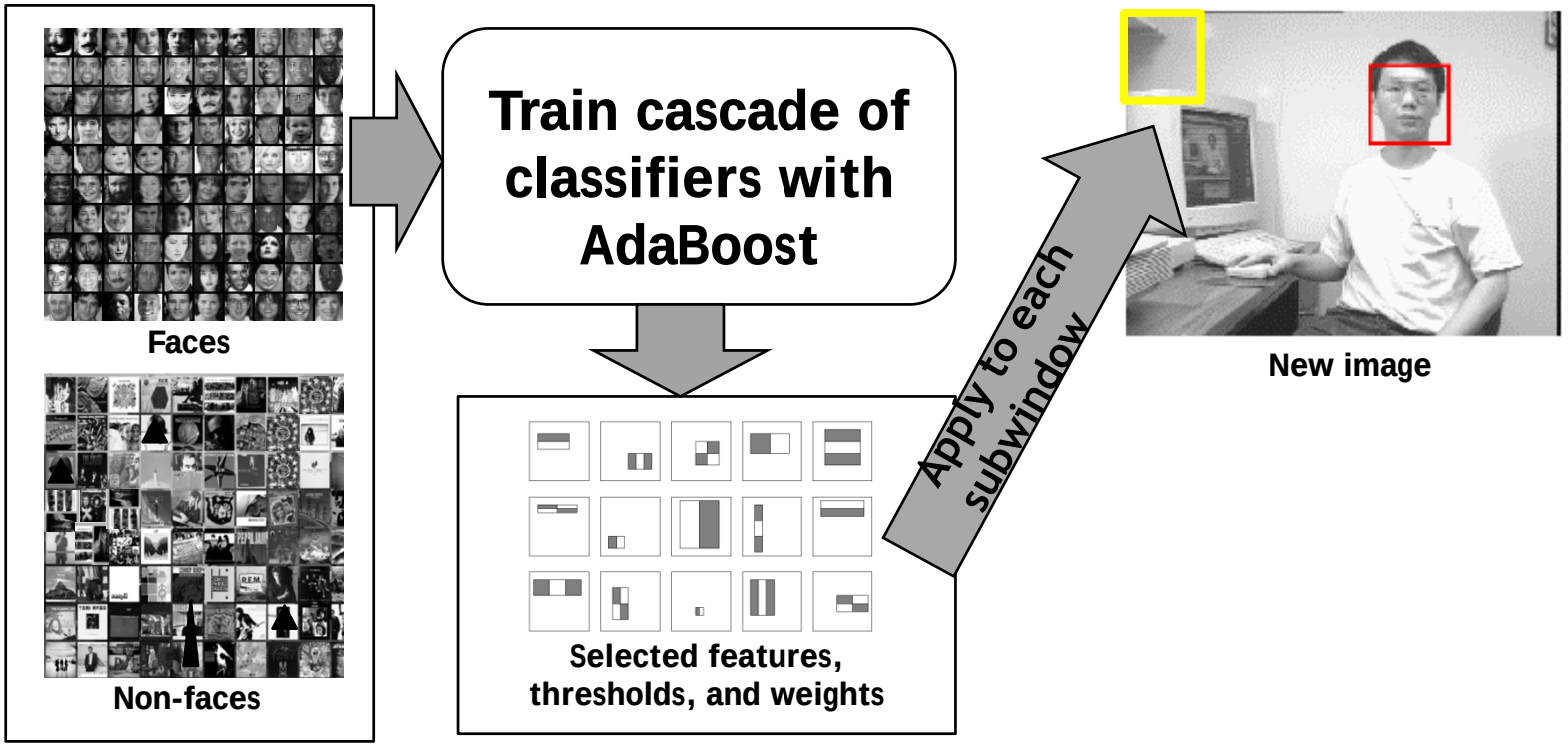
Cascading classifiers for detection



- 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

Kristen Grauman

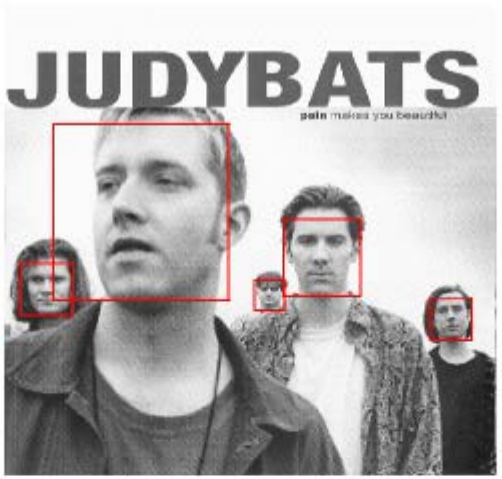
Viola-Jones detector: summary

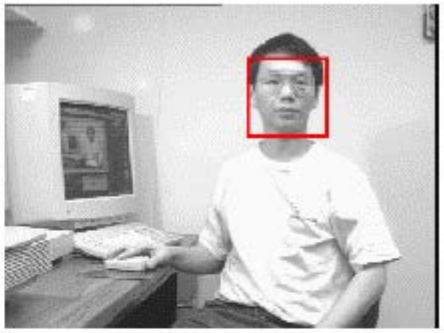


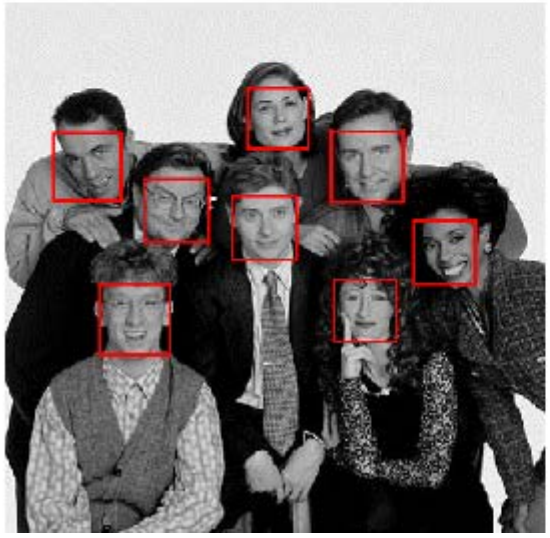
Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

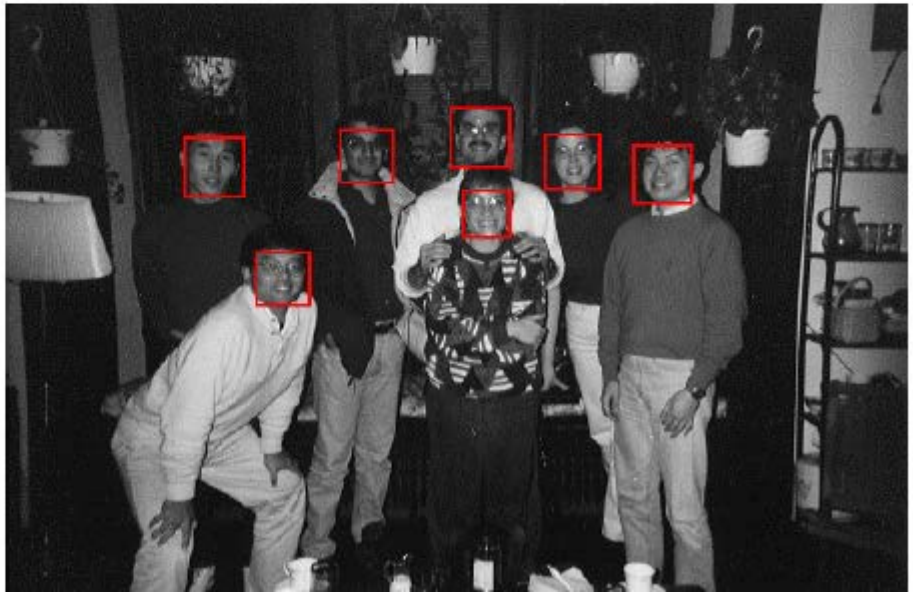
[Implementation available in OpenCV:
http://www.intel.com/technology/computing/opencv/]

Viola-Jones Face Detector: Results

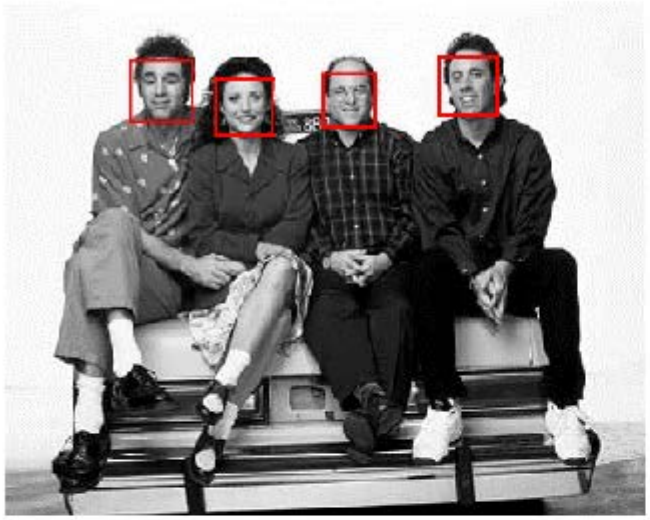


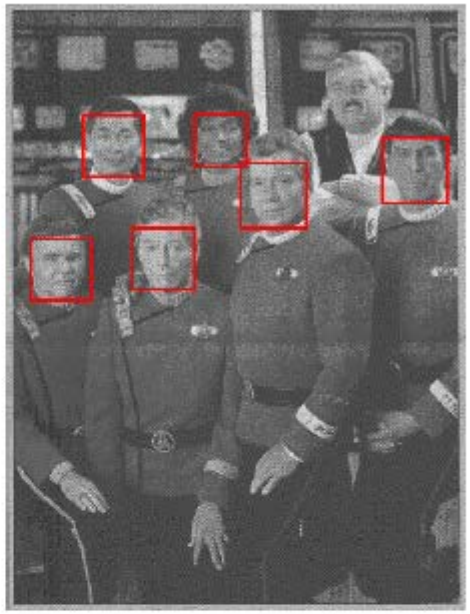


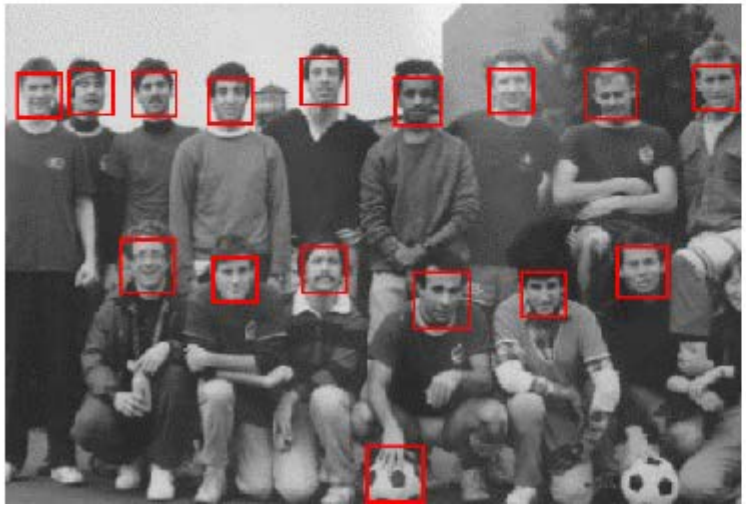


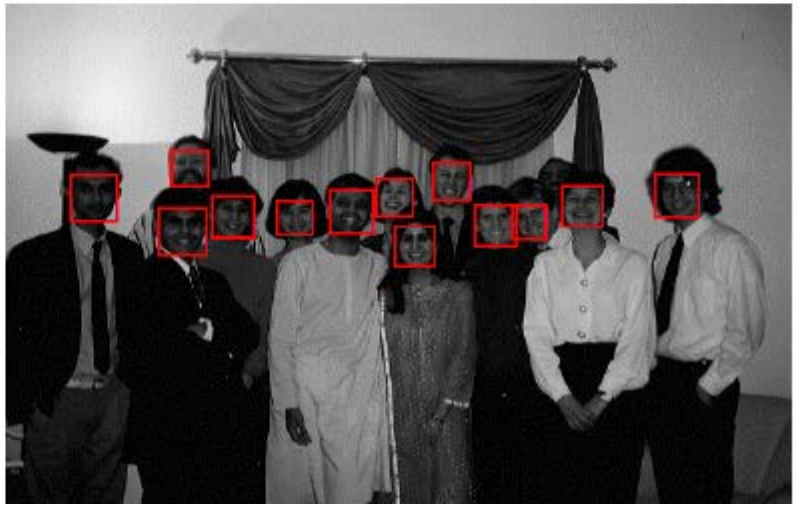


Viola-Jones Face Detector: Results





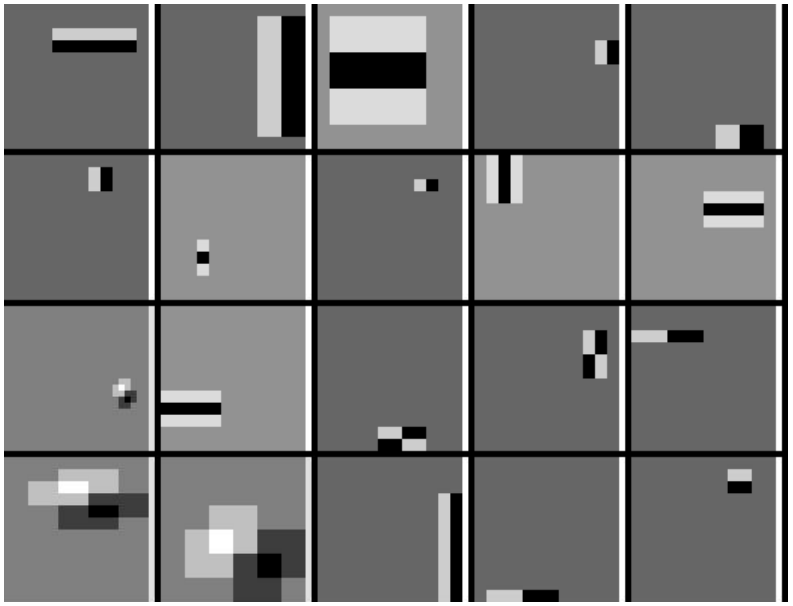




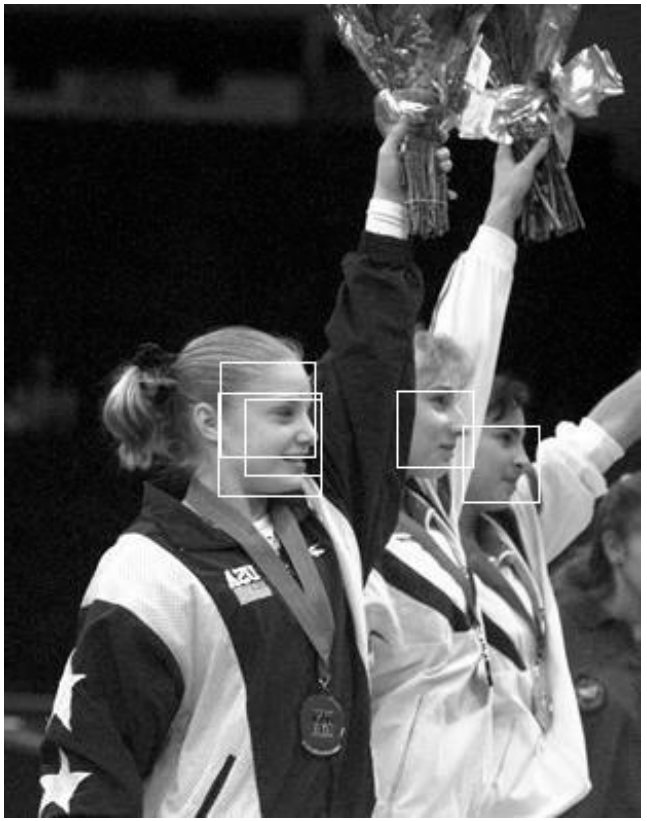
Viola-Jones Face Detector: Results

Detecting profile faces?

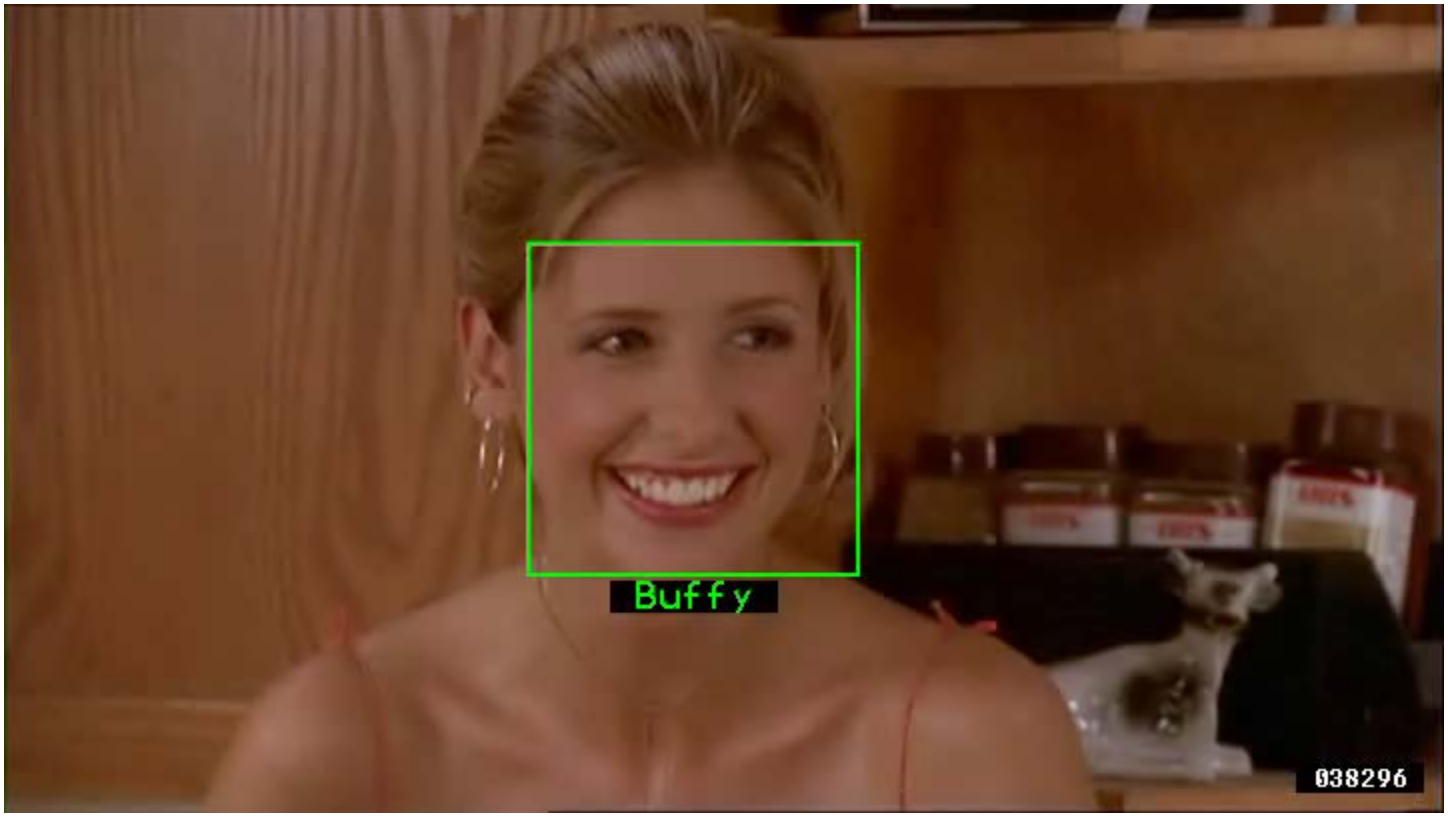
Can we use the same detector?



Viola-Jones Face Detector: Results



Example using Viola-Jones detector



Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles.

Everingham, M., Sivic, J. and Zisserman, A.
"Hello! My name is... Buffy" - Automatic naming of characters in TV video, BMVC 2006. http://www.robots.ox.ac.uk/~vgg/research/nface/index.html

Google now erases faces, license plates on Map Street View

By Elinor Mills, CNET News.com
Friday, August 24, 2007 01:37 PM

Google has gotten a lot of flack from privacy advocates for photographing faces and license plate numbers and displaying them on the Street View in Google Maps. Originally, the company said only people who identified themselves could ask the company to remove their image.

But Google has quietly changed that policy, partly in response to criticism, and now anyone can alert the company and have an image of a license plate or a recognizable face removed, not just the owner of the face or car, says Marissa Mayer, vice president of search products and user experience at Google.

"It's a good policy for users and also clarifies the intent of the product," she said in an interview following her keynote at the Search Engine Strategies conference in San Jose, Calif., Wednesday.

The policy change was made about 10 days after the launch of the product in late May, but was not publicly announced, according to Mayer. The company is removing images only when someone notifies them and not proactively, she said. "It was definitely a big policy change inside."

Consumer application: iPhoto 2009

http://www.apple.com/ilife/iphoto/

Slide credit: Lana Lazebnik

Consumer application: iPhoto 2009

- Things iPhoto thinks are faces

Slide credit: Lana Lazebnik

Viola-Jones detector: summary

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

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Boosting: pros and cons

- Advantages of boosting
  - Integrates classification with feature selection
  - 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 an alternative discriminative classifier, support vector machine (SVM)
    - especially for many-class problems

Slide credit: Lana Lazebnik

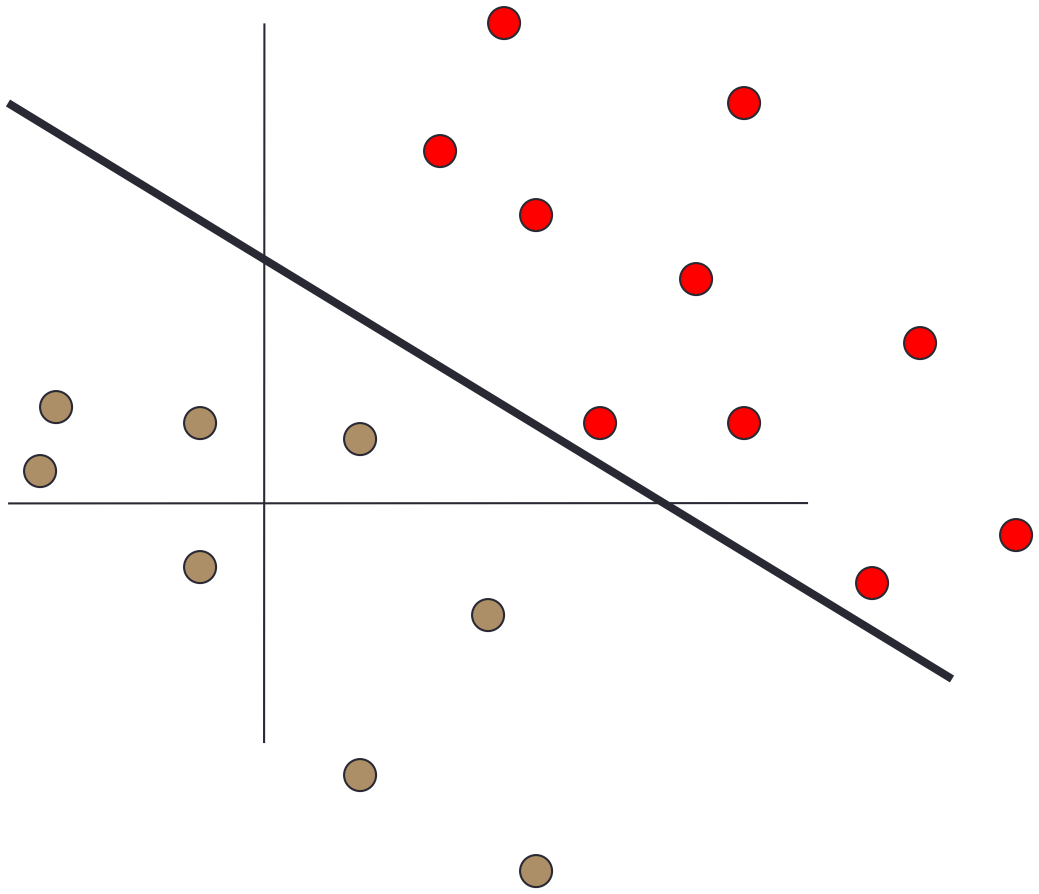
Discriminative classification methods

Discriminative classifiers – find a division (surface) in feature space that separates the classes

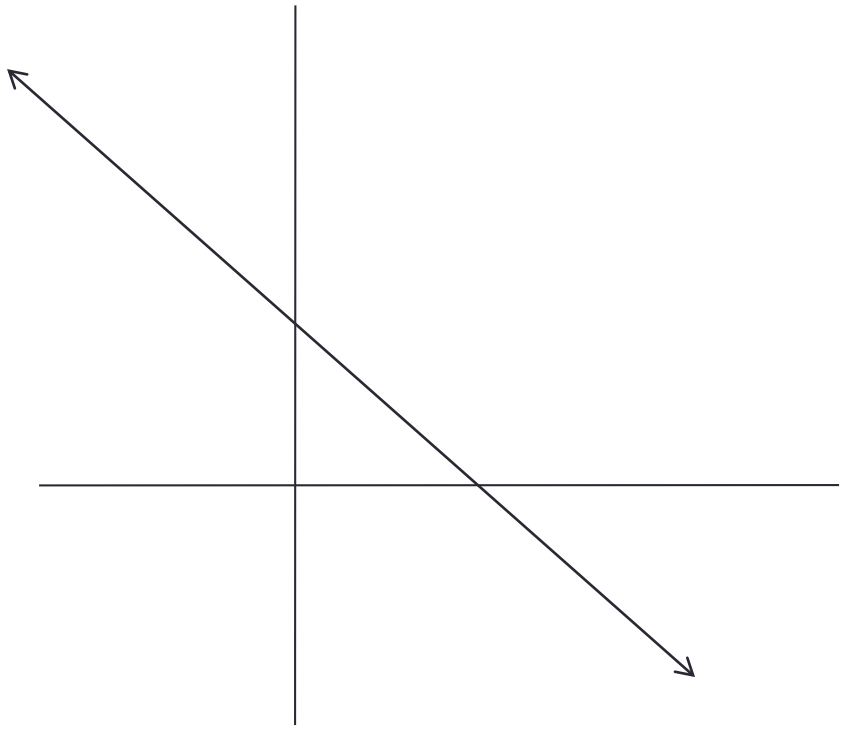
Several methods

- Nearest neighbors
- Boosting
- Support Vector Machines

Linear classifiers



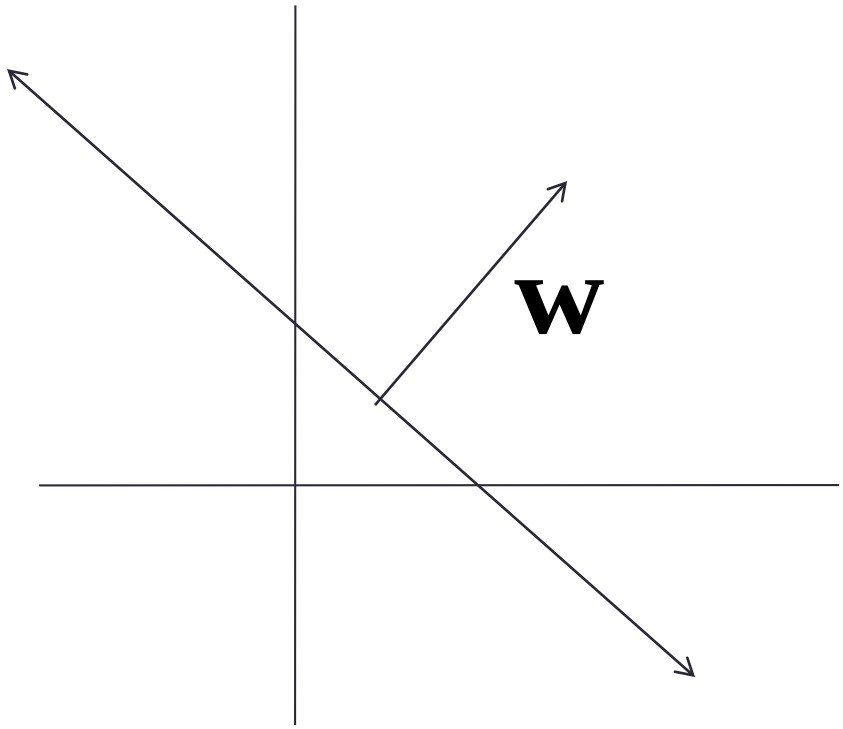
Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Lines in R^2



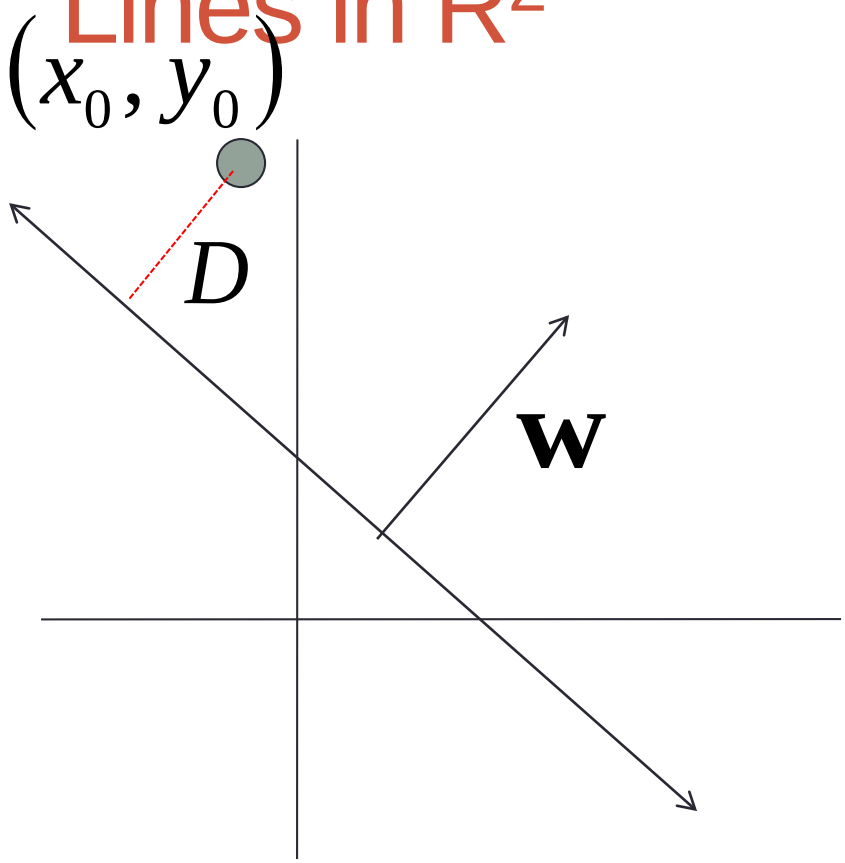
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Lines in R^2



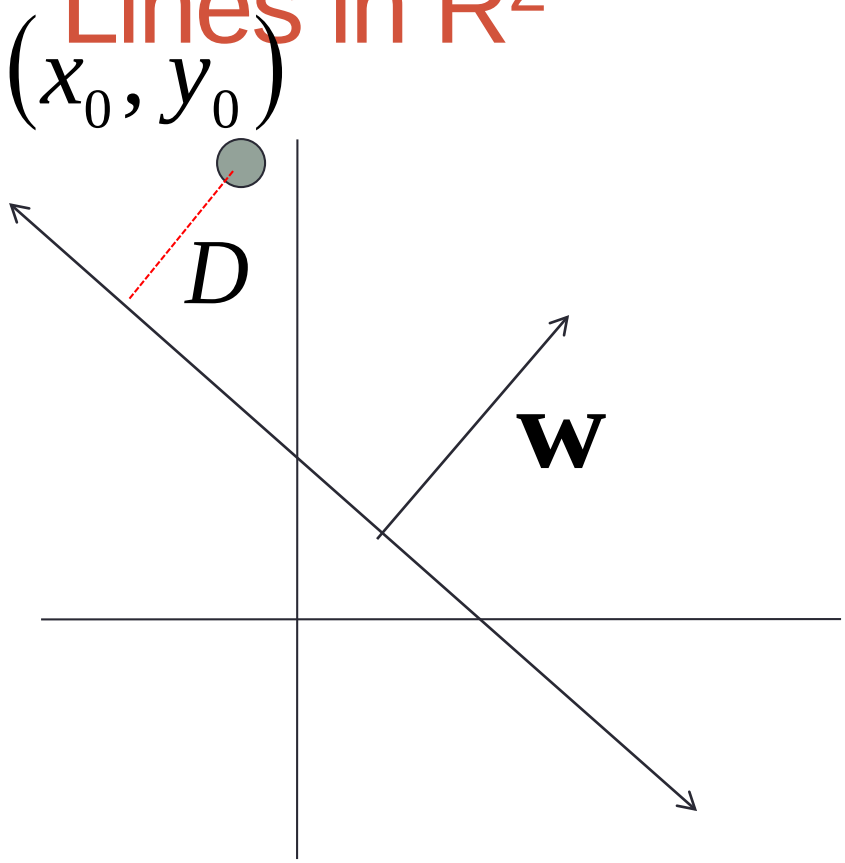
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

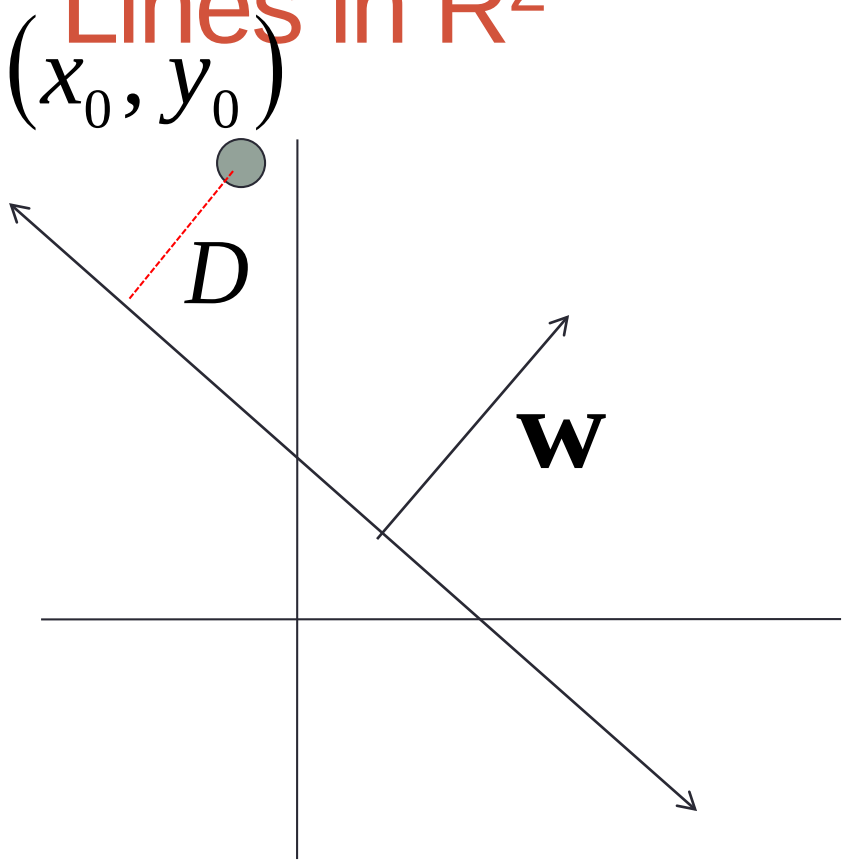
$$ax + cy + b = 0$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}}$$

distance from point to line

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

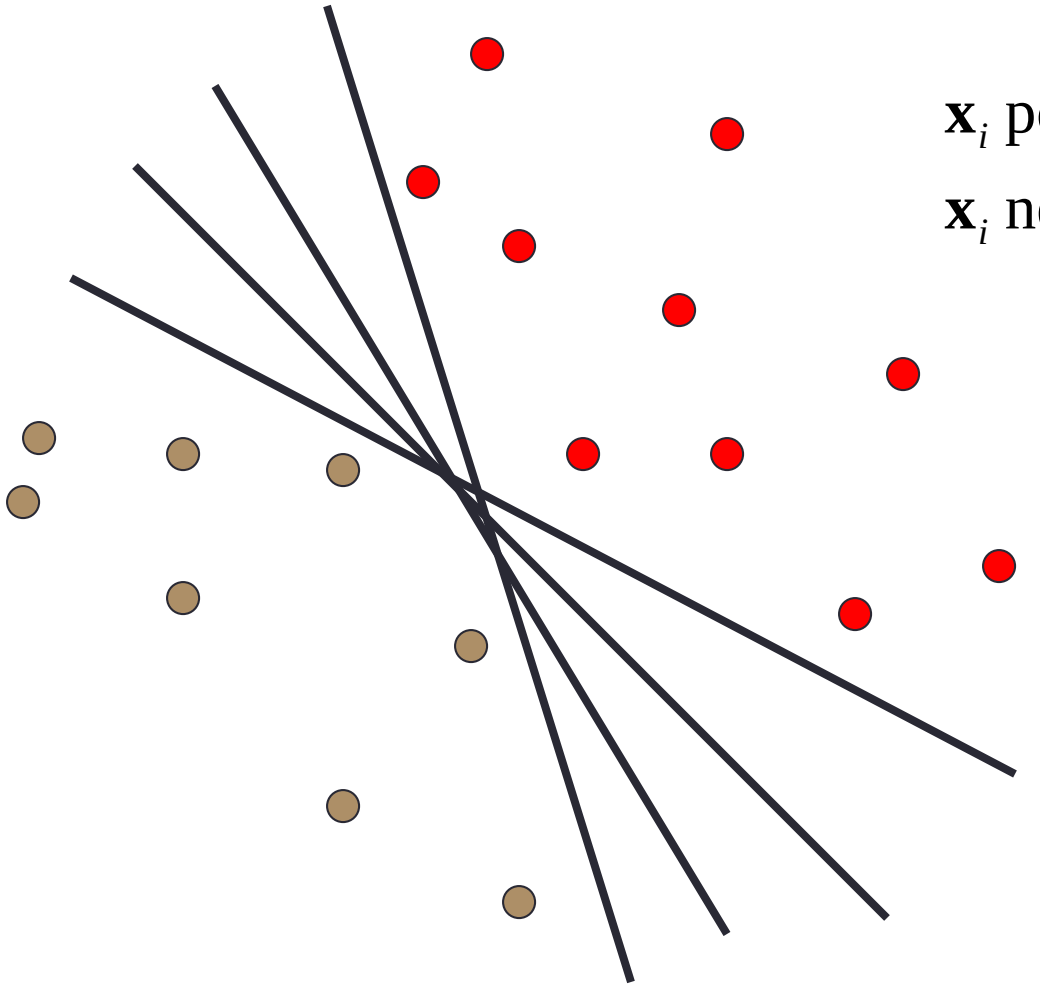
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

distance from point to line

Linear classifiers

- Find linear function to separate positive and negative examples

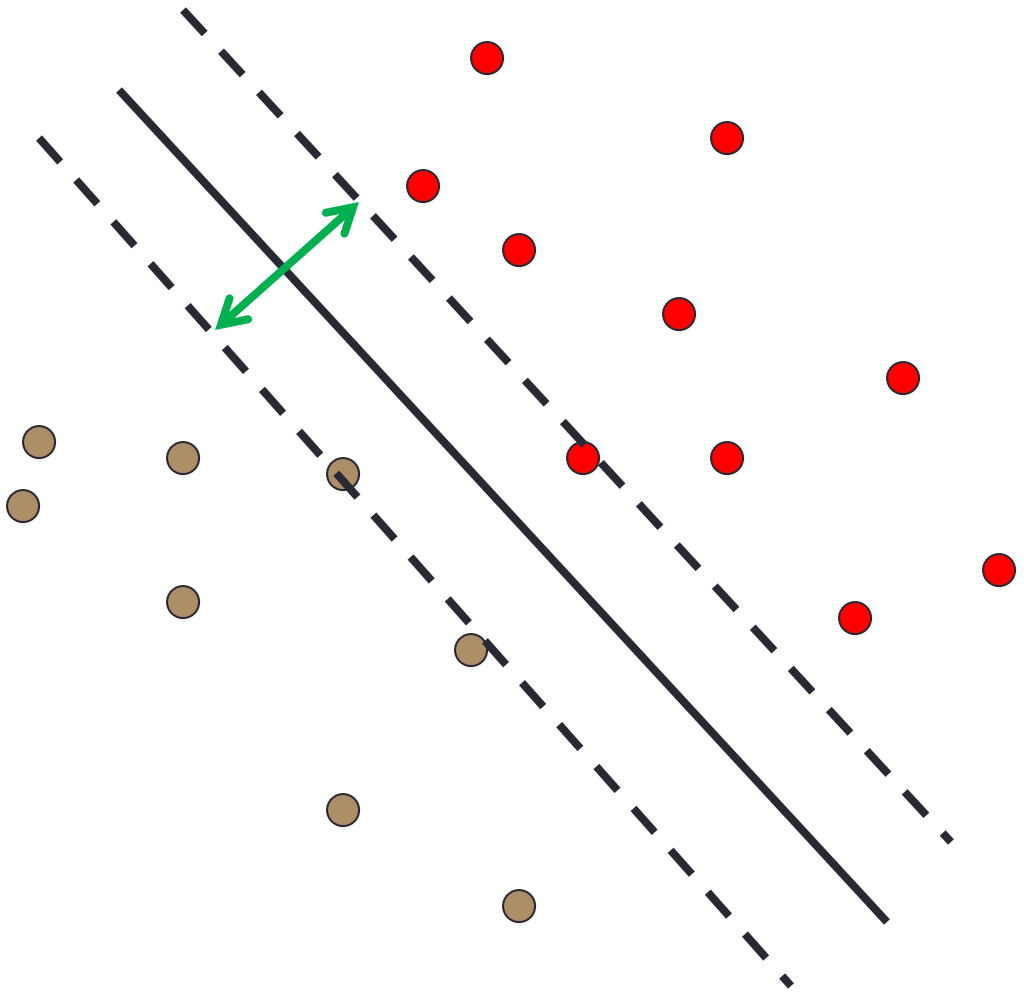


$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which line is best?

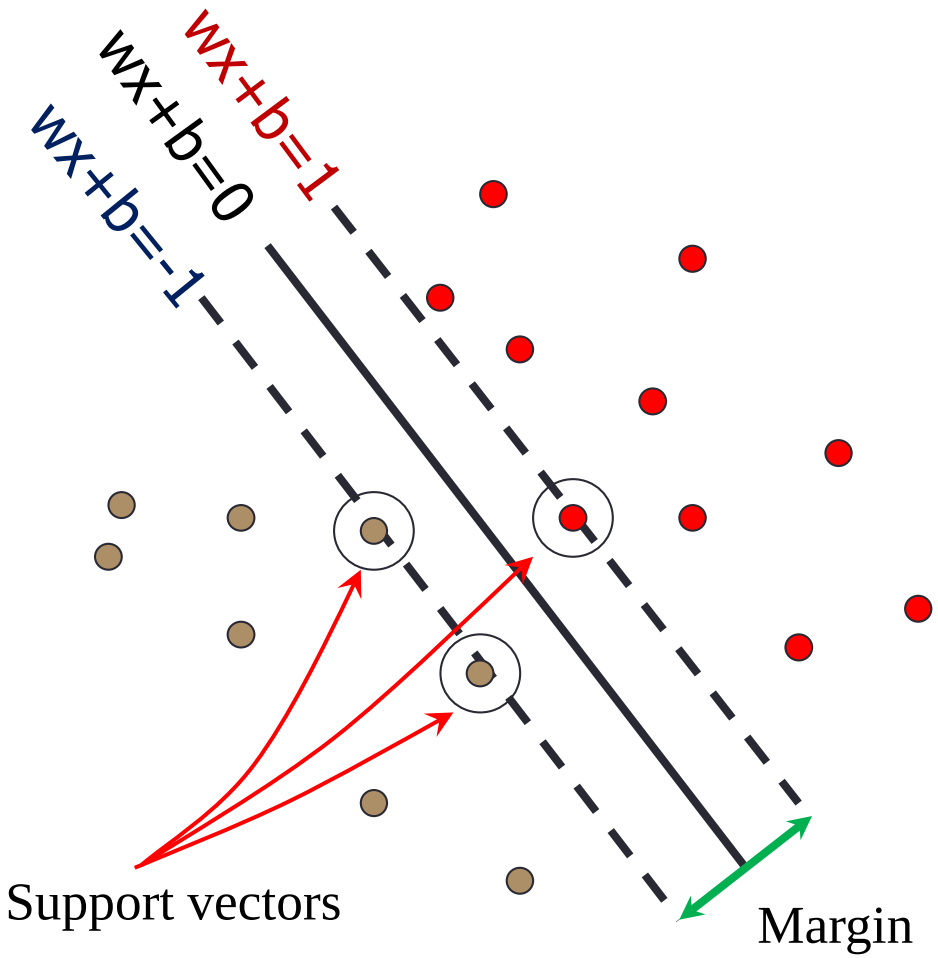
Support Vector Machines (SVMs)



- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

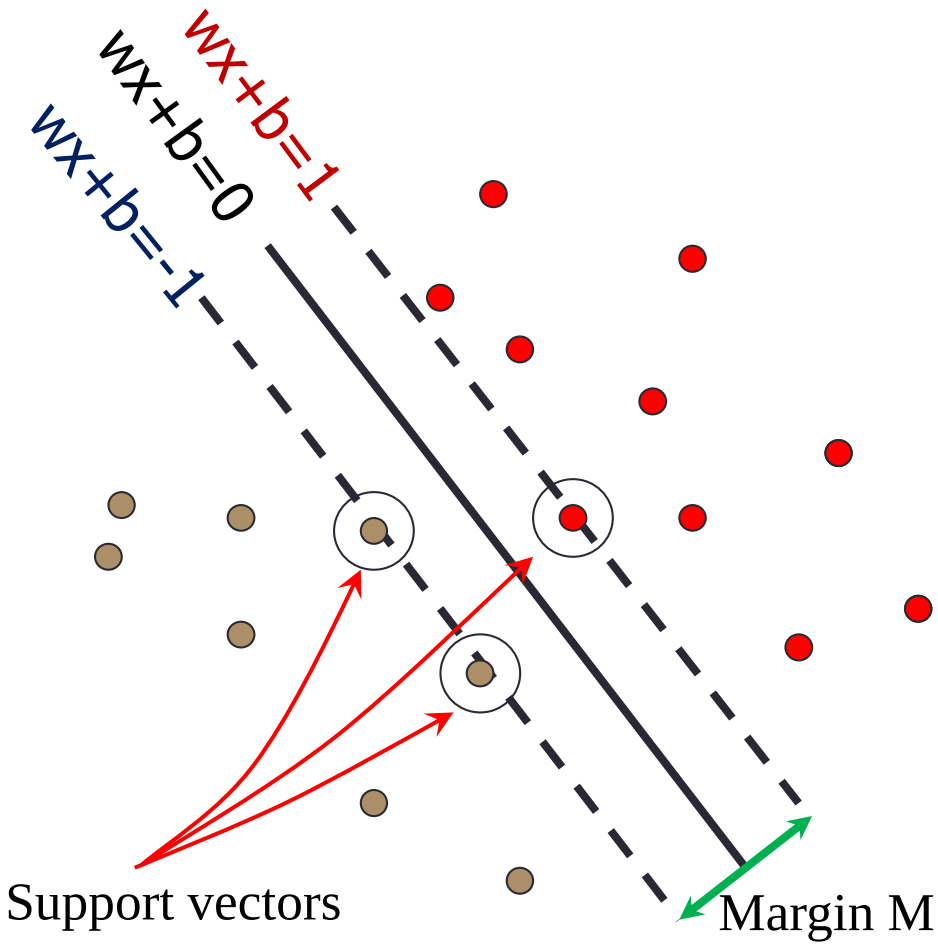
$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors:

$$\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

Finding the maximum margin line

1. Maximize margin 2/||w||
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

3. Quadratic optimization problem:

4. $$\text{Minimize } \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{Subject to } \mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

learned weight

Support vector

- The weights α_i are non-zero only at support vectors.

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Classification function:

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

$$= \text{sign}\left(\sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative,

if f(x) > 0, classify as positive

Dot product only!

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

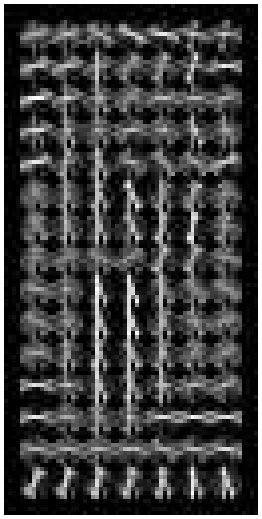
Questions

- **What if the features are not 2d?**
- What if the data is not linearly separable?
- What if we have more than just two categories?

Questions

- What if the features are not 2d?
  - Generalizes to d-dimensions – replace line with “hyperplane”
- What if the data is not linearly separable?
- What if we have more than just two categories?

Person detection with HoG's & linear SVM's



- Map each grid cell in the input window to a histogram counting the gradients per orientation.
- Train a linear SVM using training set of pedestrian vs. non-pedestrian windows.

Dalal & Triggs, CVPR 2005

Code available:
http://pascal.inrialpes.fr/soft/olt/

Person detection with HoG's & linear SVM's

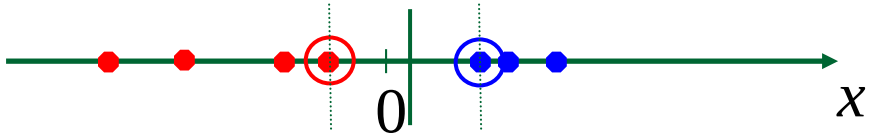
- Histograms of Oriented Gradients for Human Detection, Navneet Dalal, Bill Triggs, International Conference on Computer Vision & Pattern Recognition - June 2005
- http://lear.inrialpes.fr/pubs/2005/DT05/

Questions

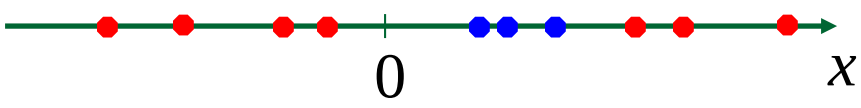
- What if the features are not 2d?
- **What if the data is not linearly separable?**
- What if we have more than just two categories?

Non-linear SVMs

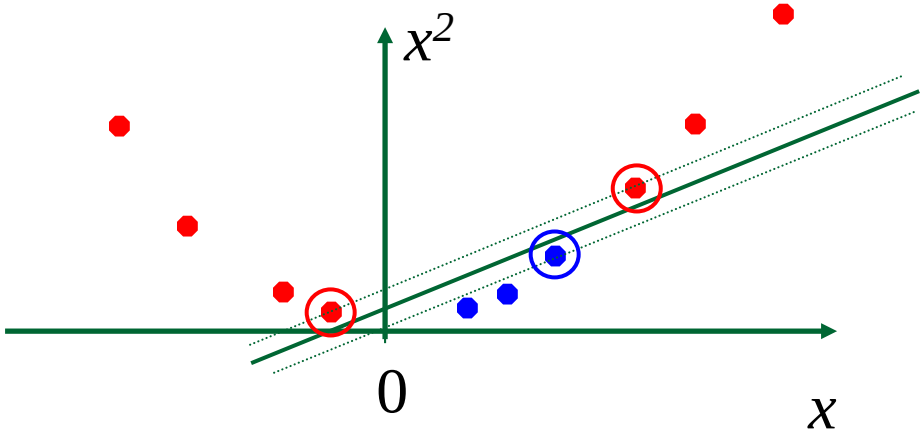
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

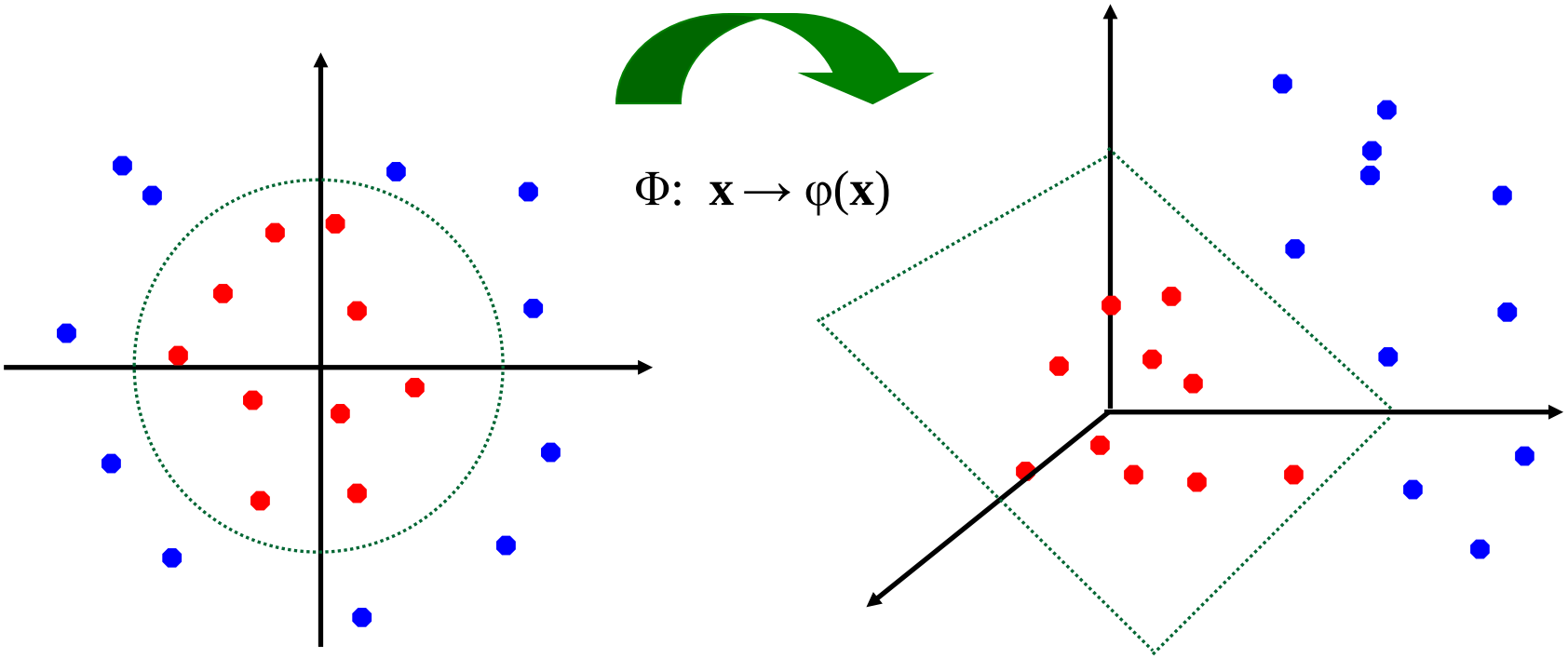


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: feature spaces

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



Slide from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

The "Kernel" Trick

- The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:

$$K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$$

- A *kernel function* is similarity function that corresponds to an inner product in some expanded feature space.

Example

2-dimensional vectors $x=[x_1 \;\; x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

$K(x_i,x_j)=(1 + x_i^T x_j)^2$,

$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T$

$[1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

$= \varphi(x_i)^T \varphi(x_j)$,

where $\varphi(x) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_{j^i}) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b \rightarrow \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Examples of kernel functions

- Linear: $K(x_i, x_j) = x_i^T x_j$

- Gaussian RBF: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

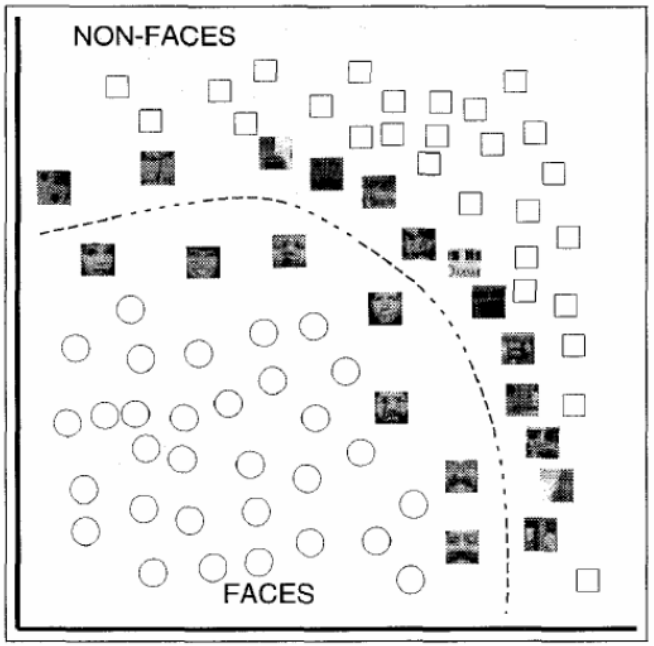
$$\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|_2^2\right) = \sum_{j=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|_2^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|_2^2\right)$$

- Histogram intersection:

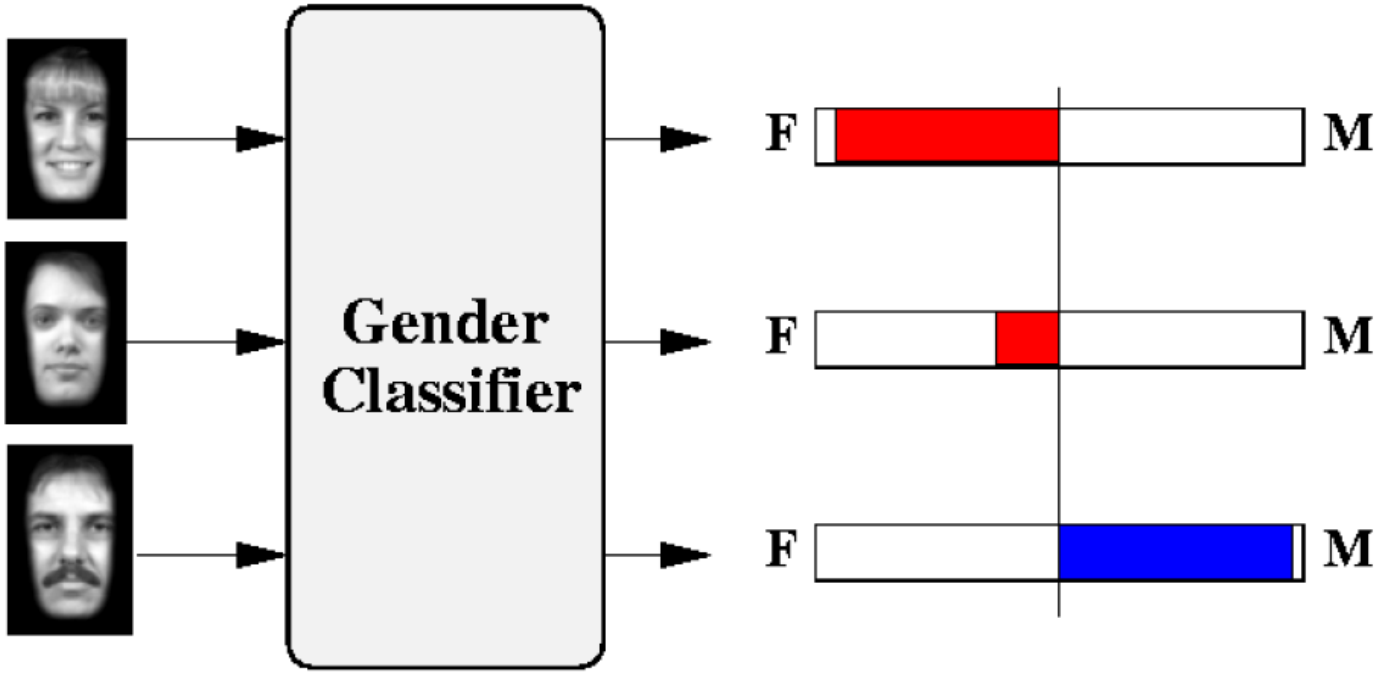
$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

SVMs for 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.



Example: learning gender with SVMs

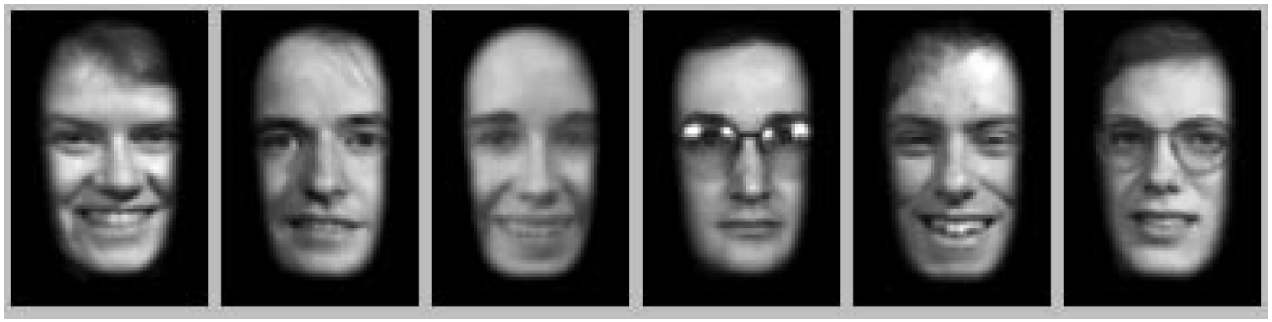


Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002.

Moghaddam and Yang, Face & Gesture 2000.

Face alignment processing

Processed faces



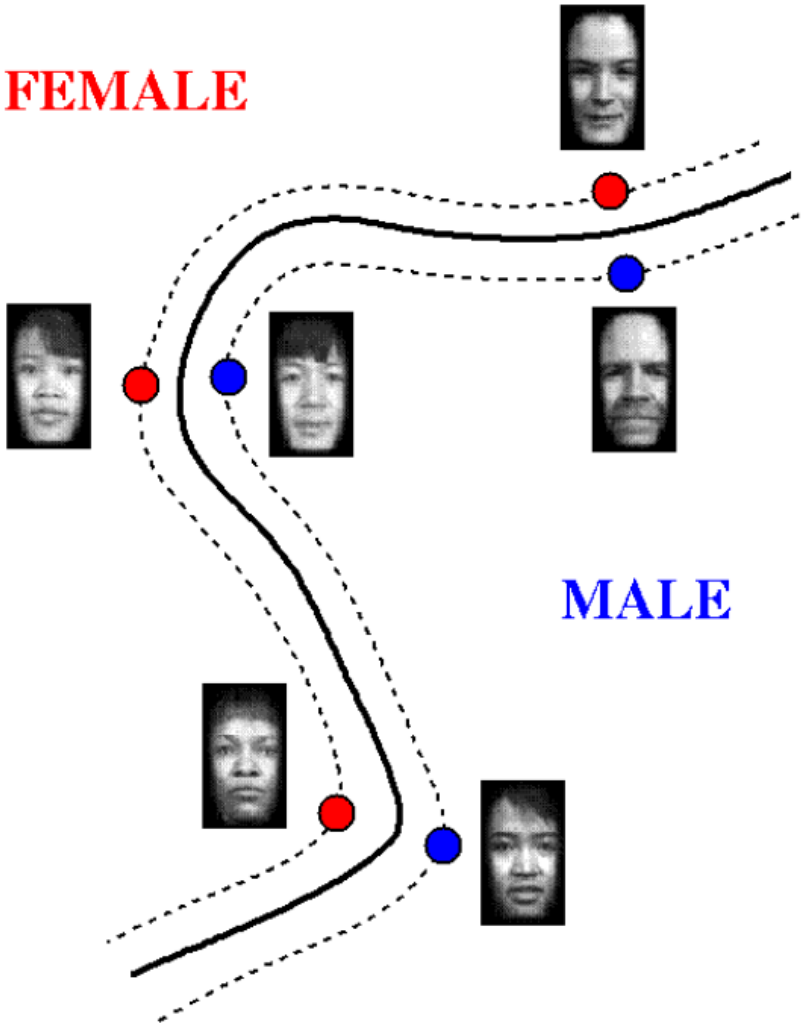
Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002.

Learning gender with SVMs

- Training examples:
  - 1044 males
  - 713 females
- Experiment with various kernels, select Gaussian RBF

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\frac{\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2}{2\sigma^2})$$

Support Faces



Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002.

Classifier Performance

Classifier	Error Rate		
	Overall	Male	Female
SVM with RBF kernel	**3.38%**	**2.05%**	**4.79%**
SVM with cubic polynomial kernel	**4.88%**	**4.21%**	**5.59%**
Large Ensemble of RBF	5.54%	4.59%	6.55%
Classical RBF	7.79%	6.89%	8.75%
Quadratic classifier	10.63%	9.44%	11.88%
Fisher linear discriminant	13.03%	12.31%	13.78%
Nearest neighbor	27.16%	26.53%	28.04%
Linear classifier	58.95%	58.47%	59.45%

Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002.

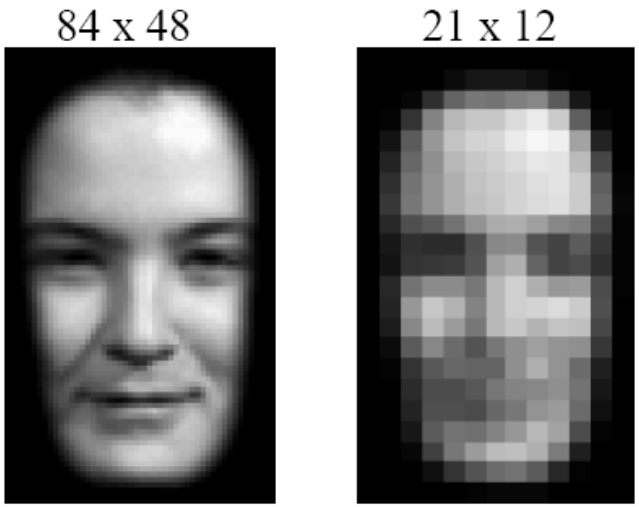
Gender perception experiment: How well can humans do?

- Subjects:
  - 30 people (22 male, 8 female)
  - Ages mid-20's to mid-40's
- Test data:
  - 254 face images (60% males, 40% females)
  - Low res and high res versions
- Task:
  - Classify as male or female, forced choice
  - No time limit

Moghaddam and Yang, Face & Gesture 2000.

Human Performance

Stimuli →



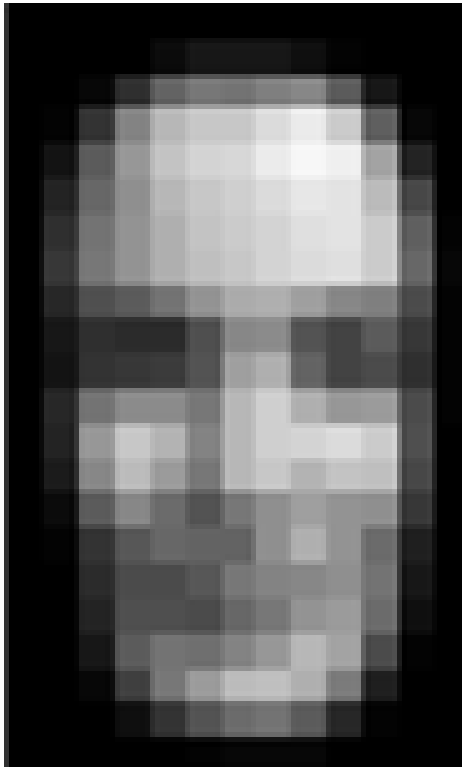
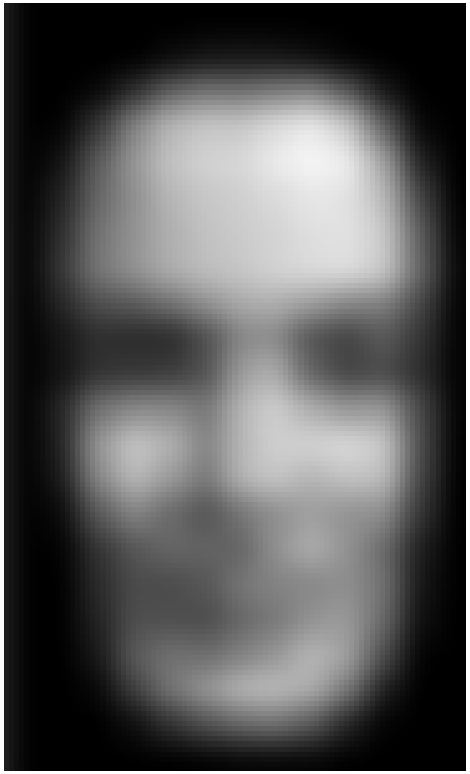
Results →

High-Res	Low-Res
6.54%	30.7%
Error	Error

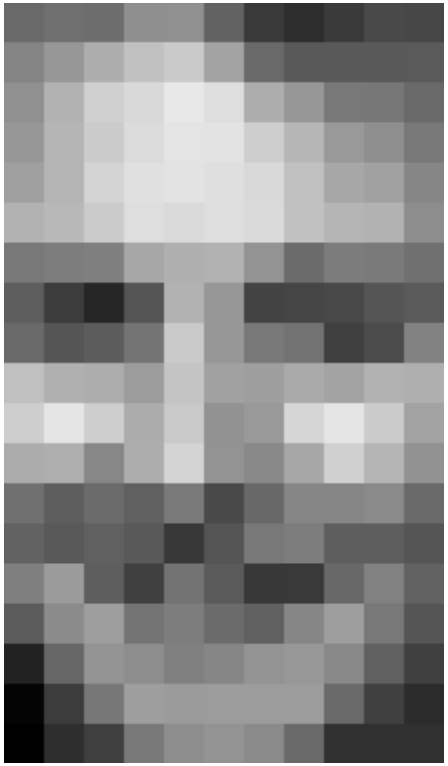
$\sigma = 3.7\%$

Moghaddam and Yang, Face & Gesture 2000.

Careful how you do things?

Careful how you do things?

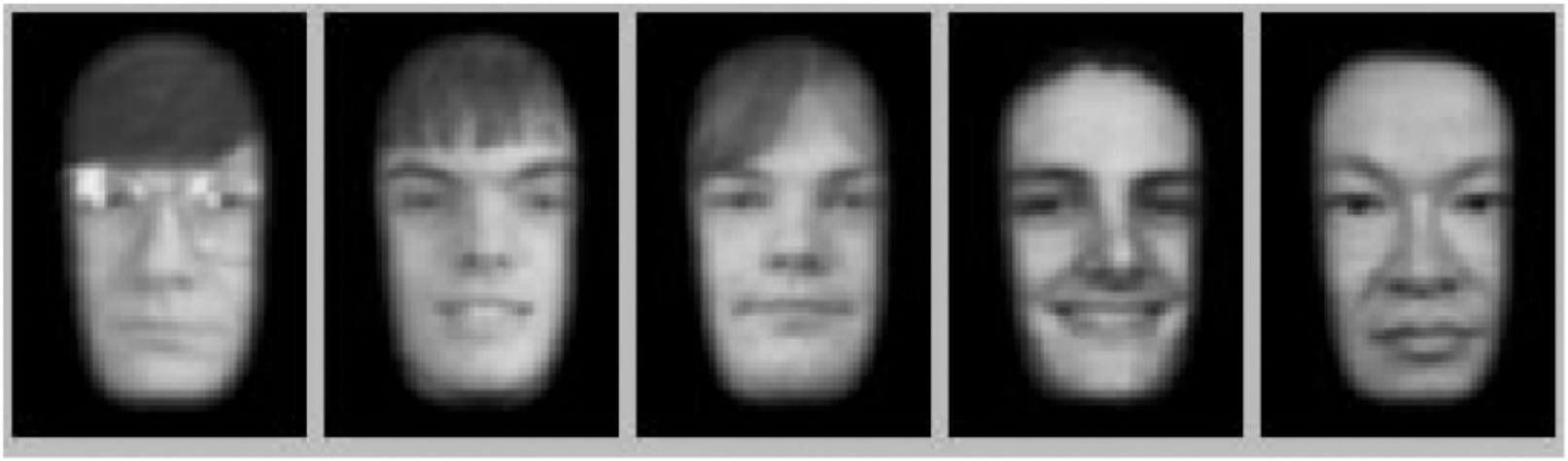


Human vs. Machine

Figure 6. SVM vs. Human performance

- SVMs performed better than any single human test subject, at either resolution

Hardest examples for humans



Top five human misclassifications

Moghaddam and Yang, Face & Gesture 2000.

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- **What if we have more than just two categories?**

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

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Kernel-based framework is very powerful, flexible
  - Often a sparse set of support vectors – compact at test time
  - Work very well in practice, even with very small training sample sizes
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Can be tricky to select best kernel function for a problem
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Adapted from Lana Lazebnik

Window-based detection: strengths

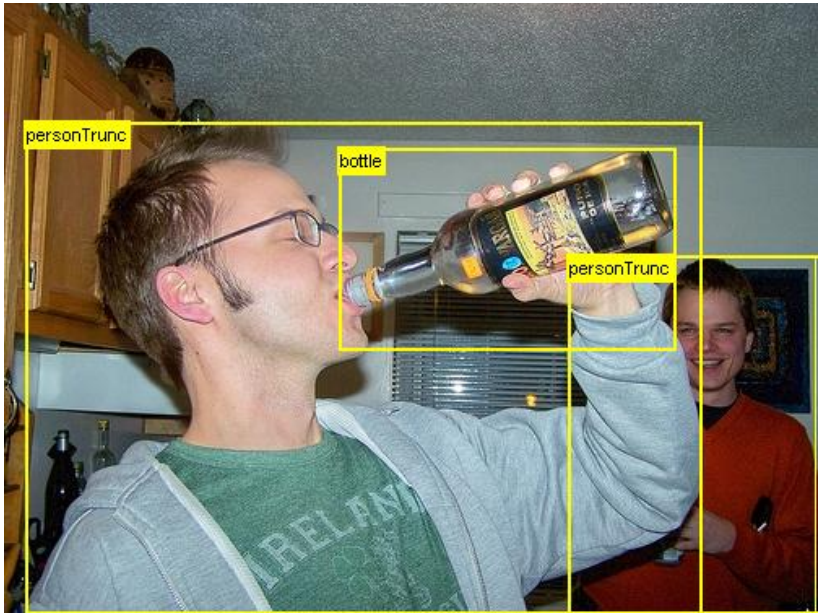
- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

Window-based detection: 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

Limitations (continued)

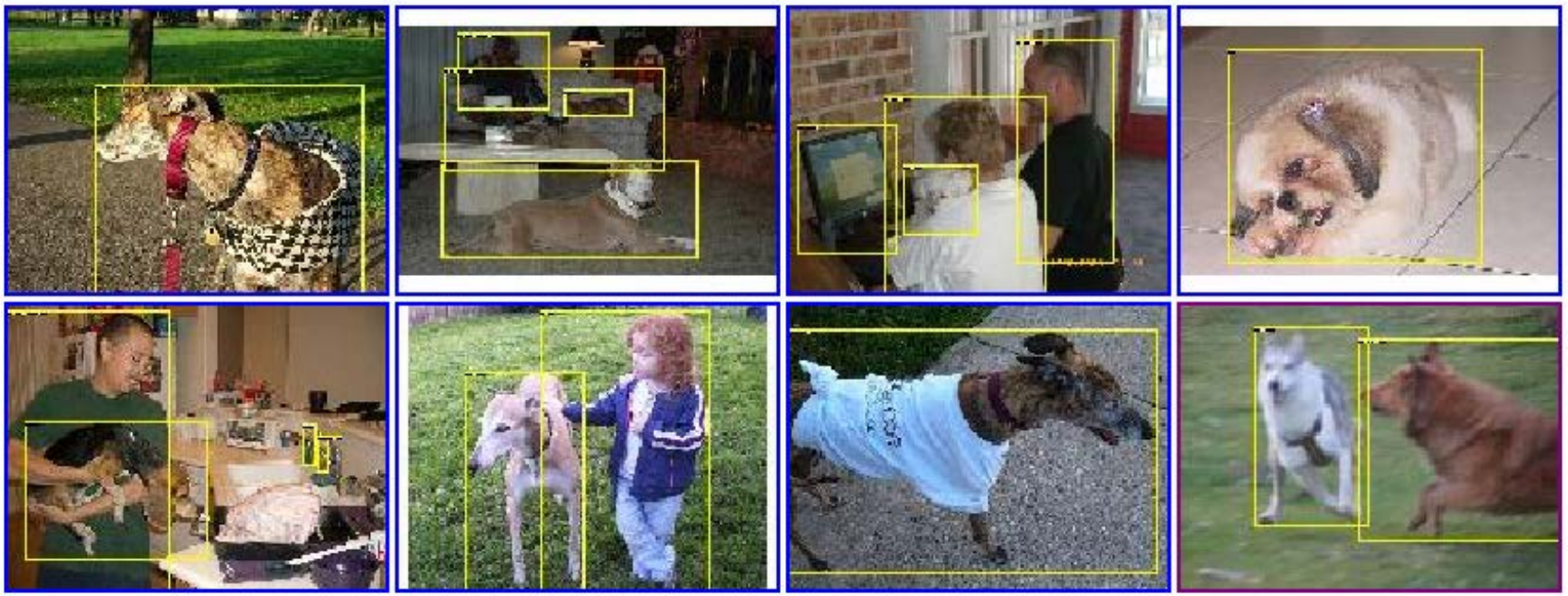
- Not all objects are “box” shaped



Kristen Grauman

Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions

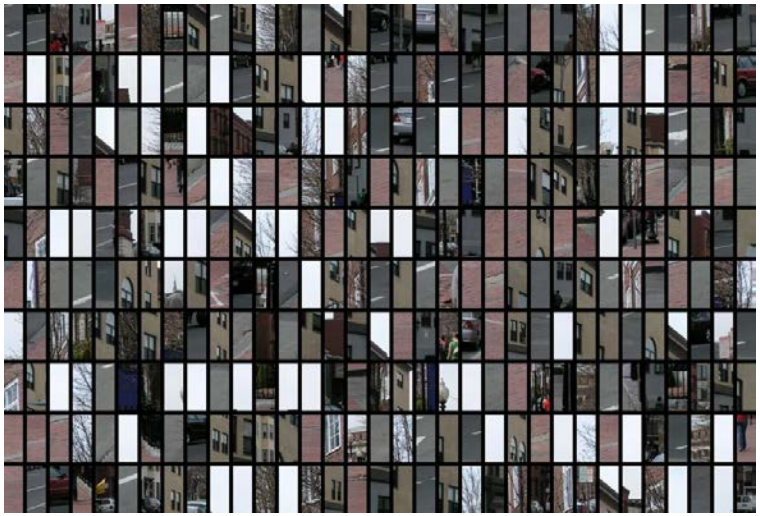


Kristen Grauman

Limitations (continued)

- If considering windows in isolation, context is lost

Sliding window



Detector's view

Figure credit: Derek Hoiem

Limitations (continued)

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

Image credit: Adam, Rivlin, & Shimshoni

Summary

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Boosting classifiers: general idea
- Viola-Jones face detector
  - Exemplar of basic paradigm
  - Plus key ideas: rectangular features, Adaboost for feature selection, cascade
- Pros and cons of window-based detection