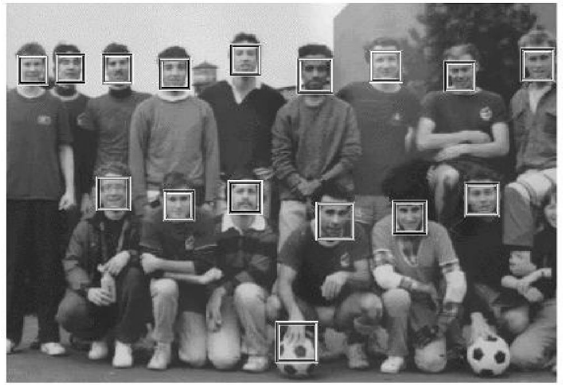
# Machine Learning for Image Recognition

## Outline

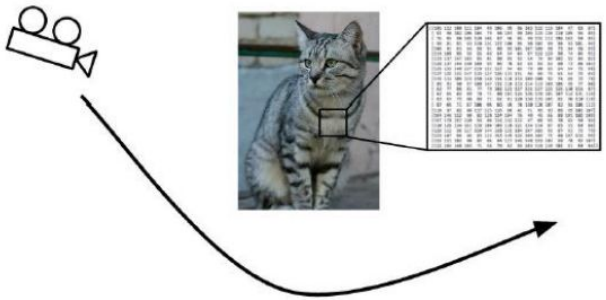
- BoW
- HOG + SVM
- Viola Jones
- Evaluation Metrics



Acknowledgement : Most of slide credits go to Kristen Gauman, Udacity and ECE627 - Computer Vision

# Challenges of Image Recognition

Viewpoint



Illumination

This image is CC0 1.0 public domain

Deformation

This image by Umberto Salvagnin is licensed under CC-BY 2.0

Occlusion

This image by jonsson is licensed under CC-BY 2.0

Clutter

This image is CC0 1.0 public domain

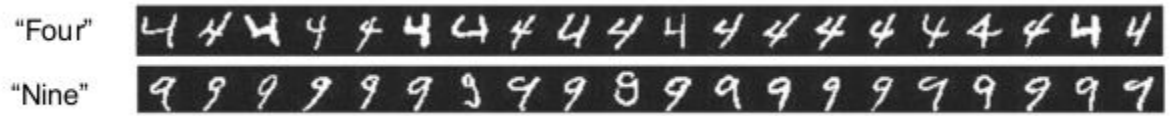
Intraclass Variation

This image is CC0 1.0 public domain

# What machine learning is really about

- Generative and Discriminative approaches
- Define a loss/objective function (optimization target) that quantifies our unhappiness (fitness) with the scores across the training set.
- Come up with a way of efficiently finding the parameters to minimize the loss function (optimization)
- Given a collection of labeled examples - Supervised learning
- Since we know the desired labels of training data, we want to minimize the expected misclassification ( Loss function )

Training Data



# Generative Models

The best decision boundary is at point x where:

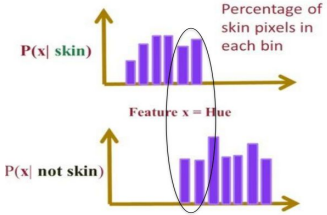
$P(class\ is\ 9|x)L(9 \rightarrow 4)$

$= P(class\ is\ 4|x)L\ (4 \rightarrow 9)$

Probability that is actually 9

Cost of labelling a 9 to 4

At best decision boundary, either choice of label yield same expected loss




posterior likelihood prior

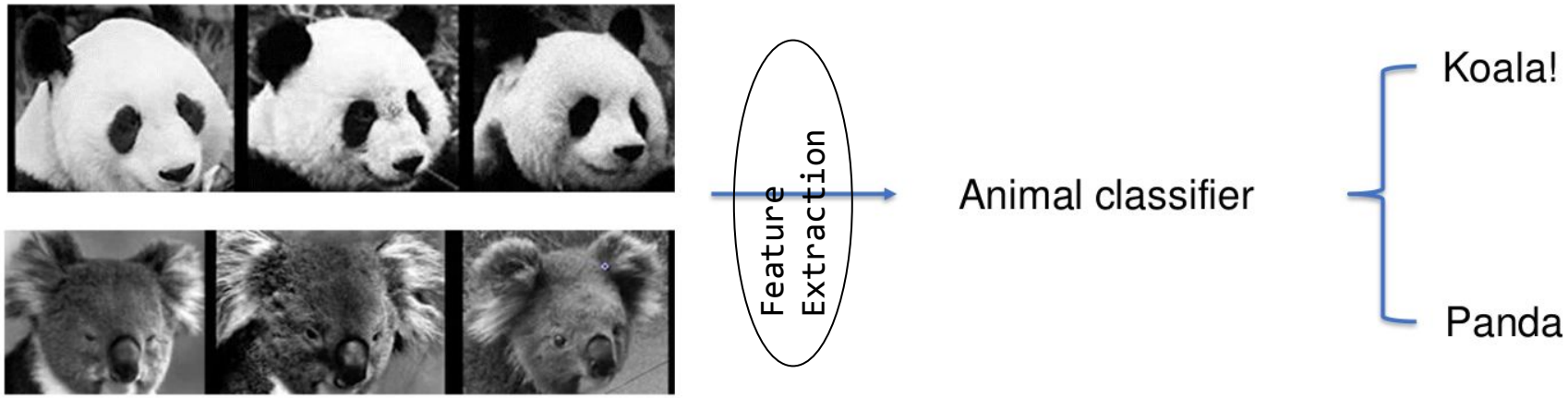
$$P(skin|x) = \frac{P(x|skin) \times P(skin)}{P(x)}$$

$$P(skin|x) \propto P(x|skin) \times P(skin)$$

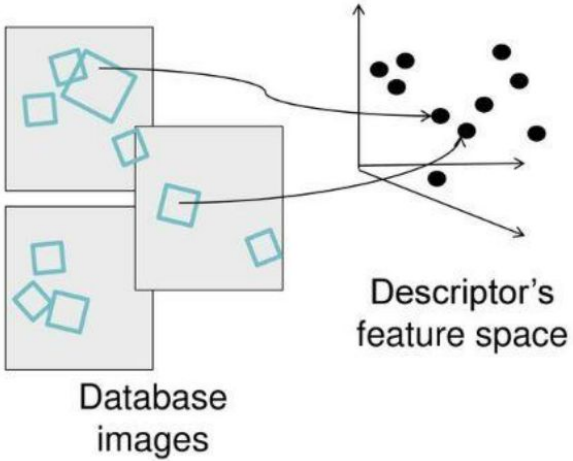
Where does the prior come from?

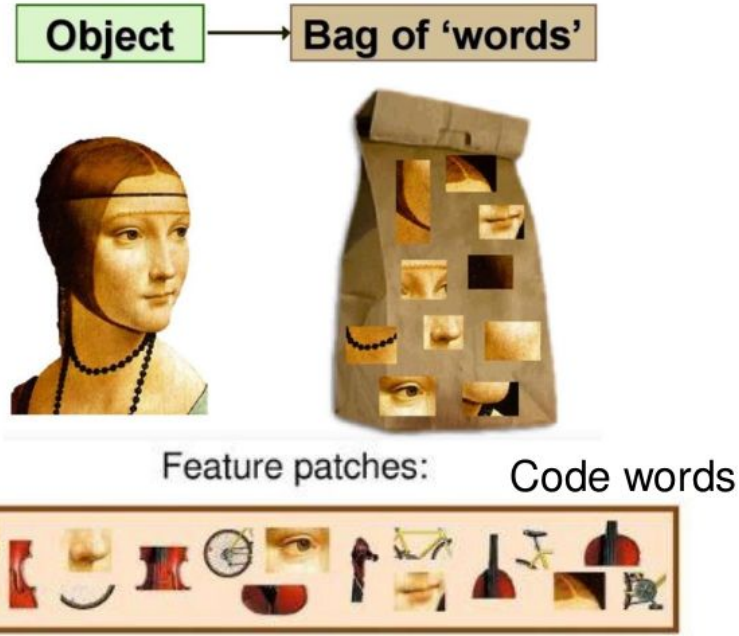
- Learn it from data
- Note: $P\ skin\ + P\ {\sim}skin\ = 1$

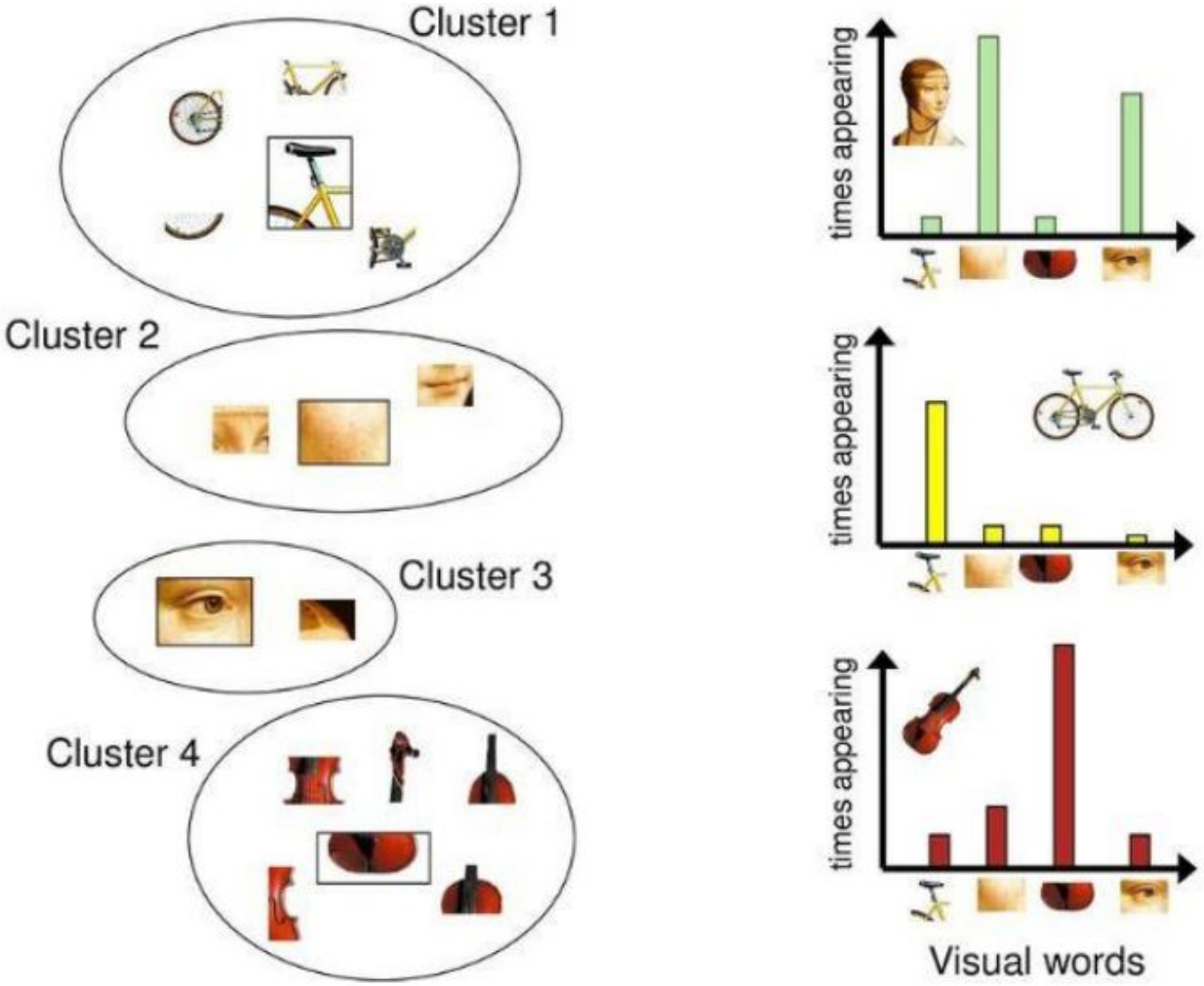
# Discriminative Models



# Bag of Visual Words

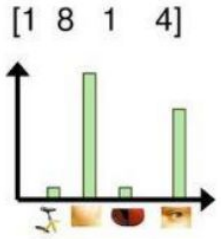


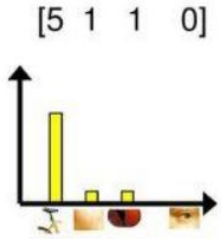




Cluster 1
Cluster 2
Cluster 3
Cluster 4
times appearing
times appearing
times appearing
Visual words

$\vec{d}_j$

$\vec{q}$



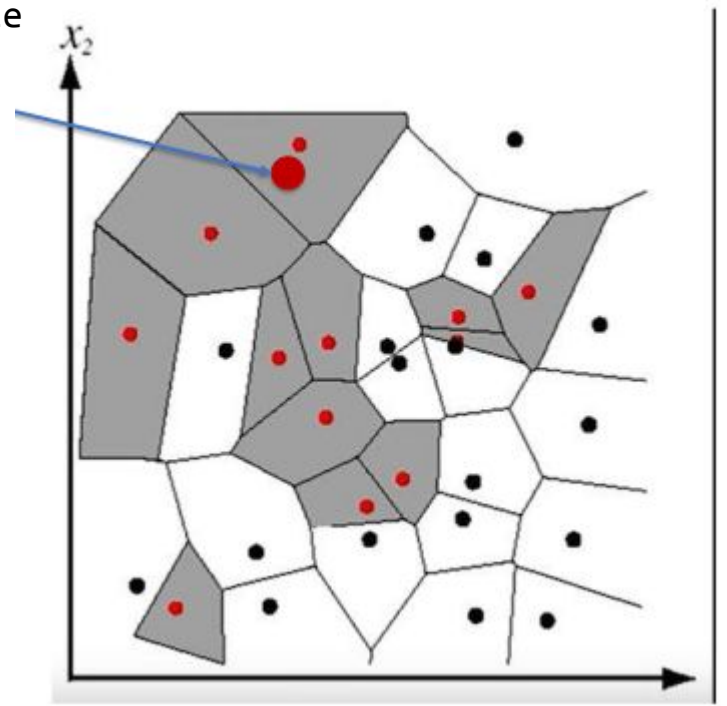

$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

$$= \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}}$$

for vocabulary of *V* words

# Nearest Neighbor for Classification

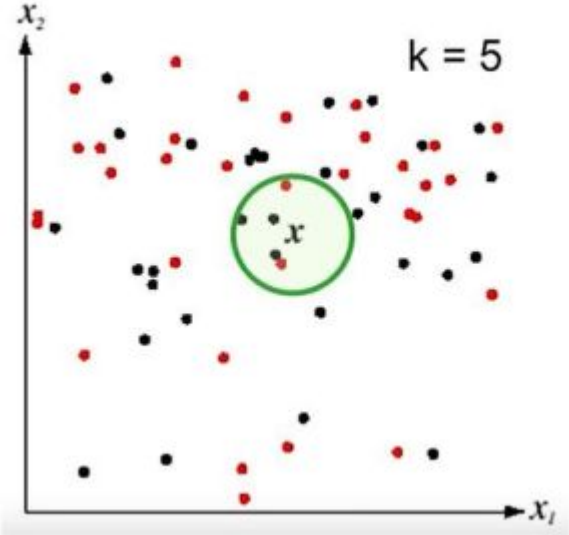
Novel Test Example




Black = Class A
Red = Class B

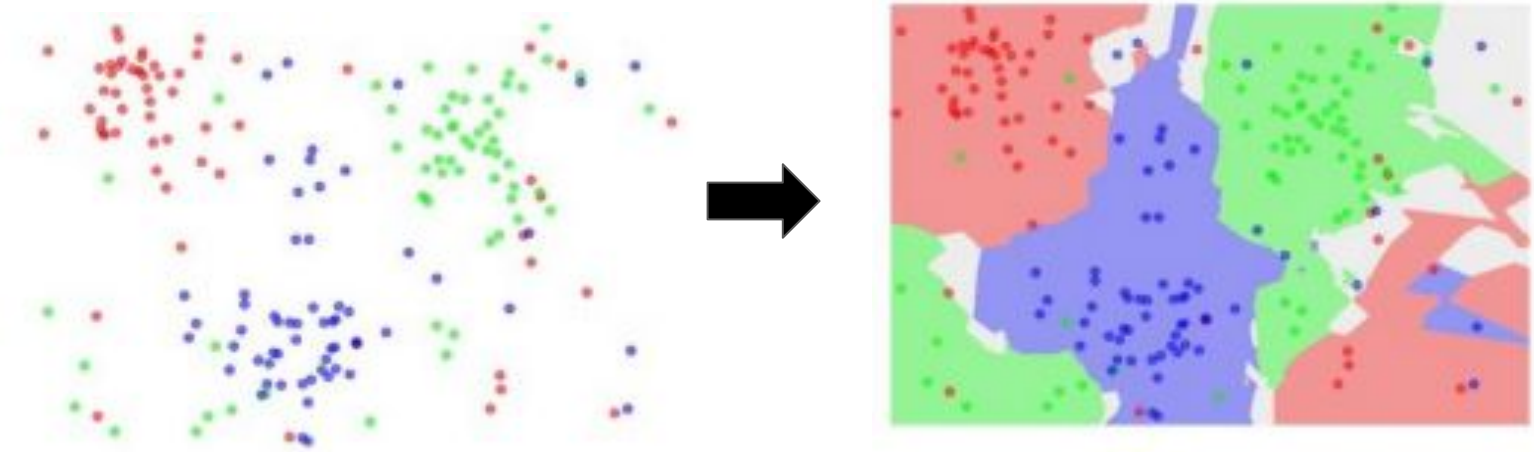
2-D Categorical Data

# Nearest Neighbor for Classification

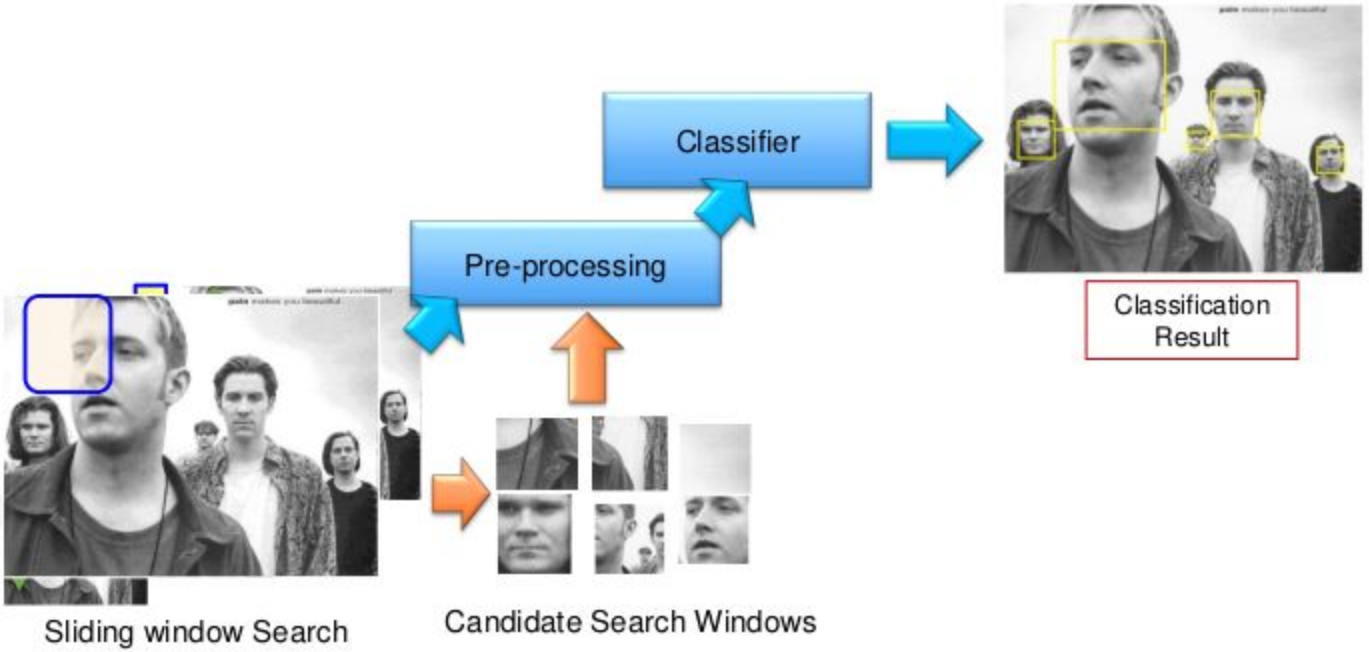



K = 5
5 Nearest Neighbors
3 Class A and 2 Class B,
so x is Class A

# Nearest Neighbor for Classification

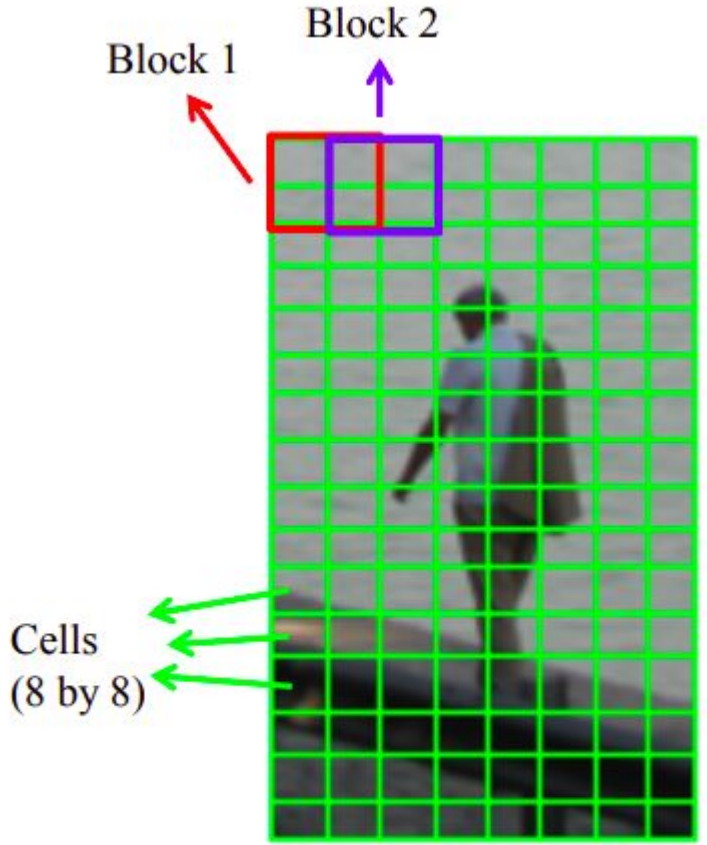


# Sliding window approach

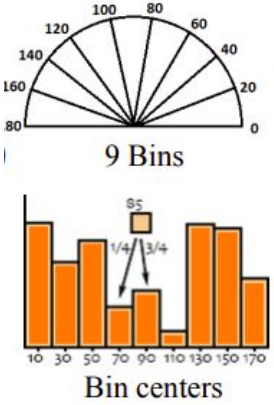


# Blocks, Cells

- 16x16 blocks of 50% overlap.
    - 7x15=105 blocks in total
- Each block should consist of 2x2 cells with size 8x8.

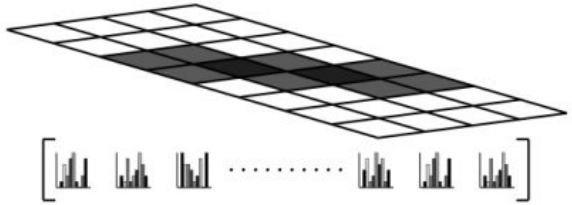


- Each block consists of 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins (0-180)
  - The vote is the gradient magnitude Interpolate votes linearly between neighboring bin centers.
  - Example: if θ=85 degrees.  Distance to the bin center Bin 70 and Bin 90 are 15 and 5 degrees, respectively.
  - Hence, ratios are 5/20=1/4, 15/20=3/4.
- The vote can also be weighted with Gaussian to down weight the pixels near the edges of the block

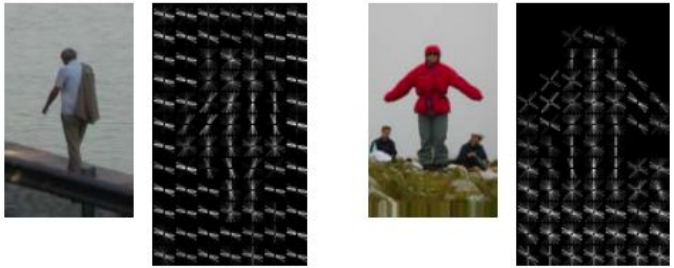


# Final Feature Vector

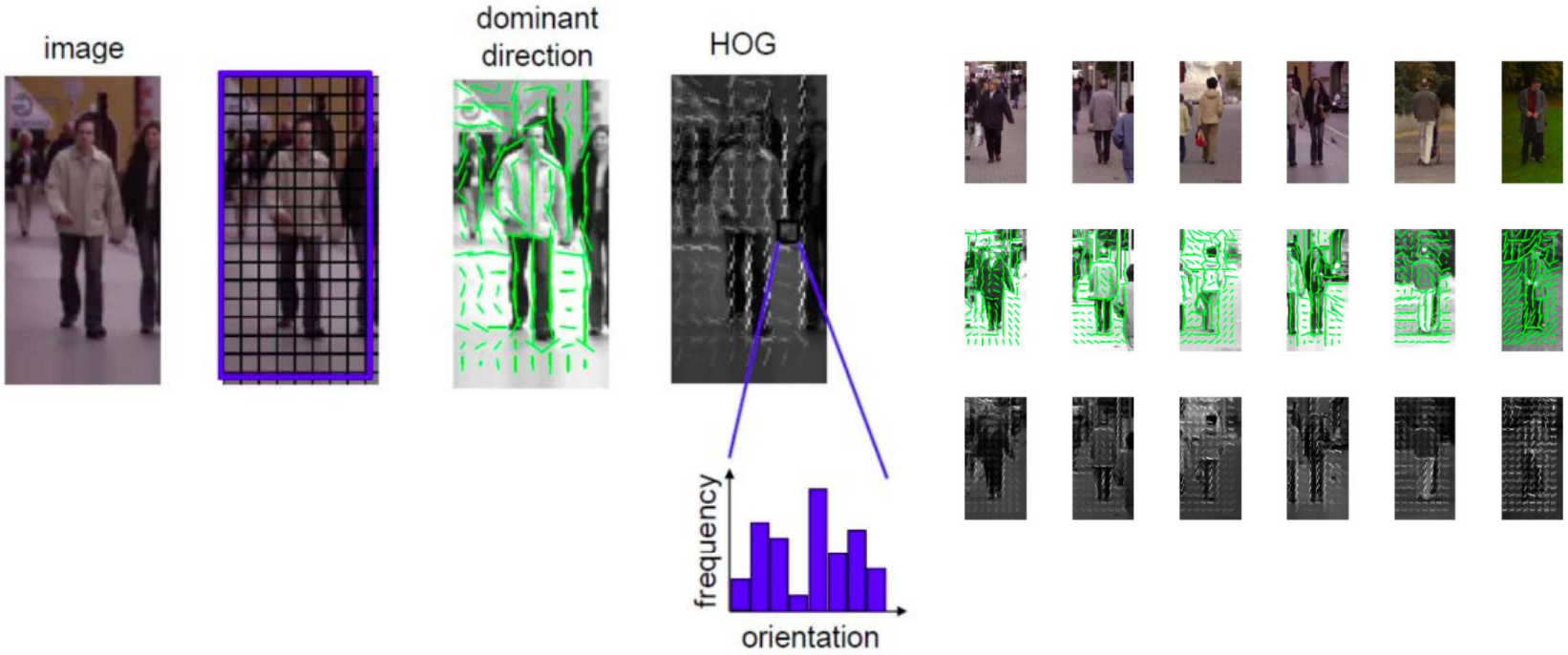
- Concatenate histograms
  - Make it a 1D vector of length 3780.



- Visualization



# HOG Features



# HOG Features

## Training (Learning)

- Represent each example window by a HOG feature vector

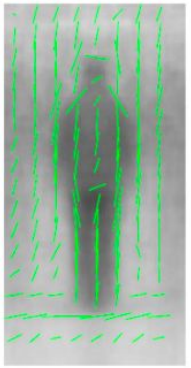
$$\mathbf{x}_i \in \mathbb{R}^d,$$

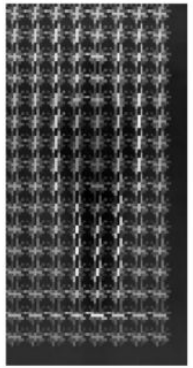
- Train a SVM classifier

## Testing (Detection)

- Sliding window classifier

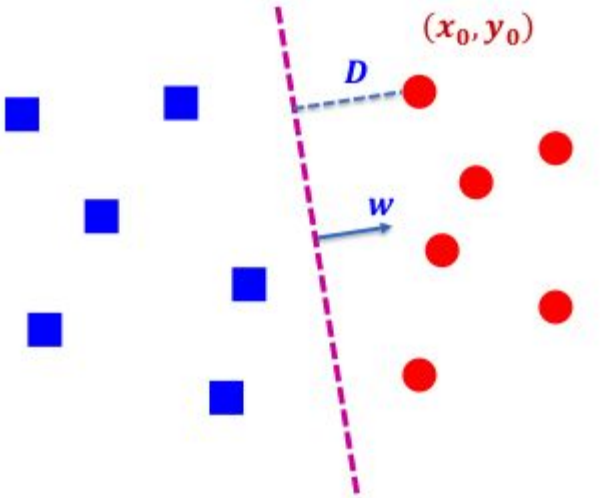
$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$



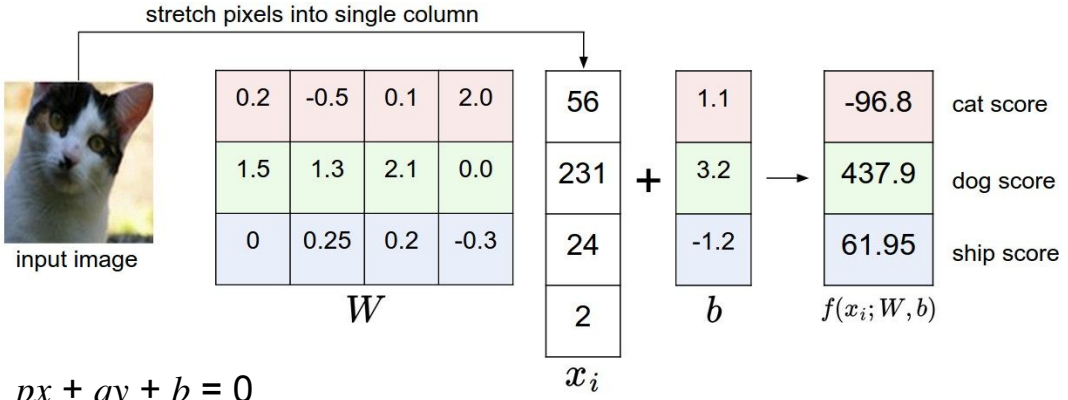


Slides Credits
Andrew Zisserman

# Linear Classifier




Y pred = sign(W.X + b)



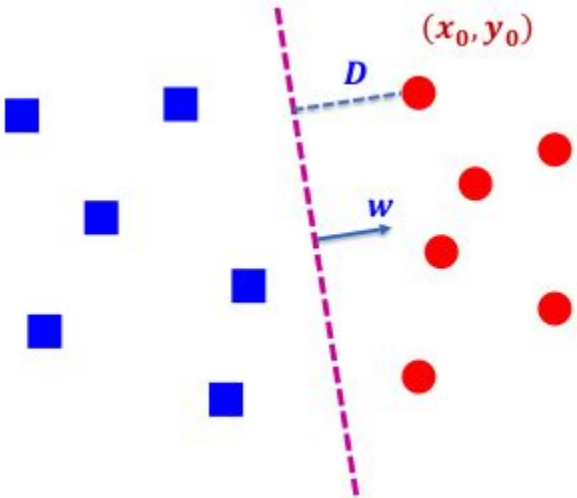

$px + qy + b = 0$

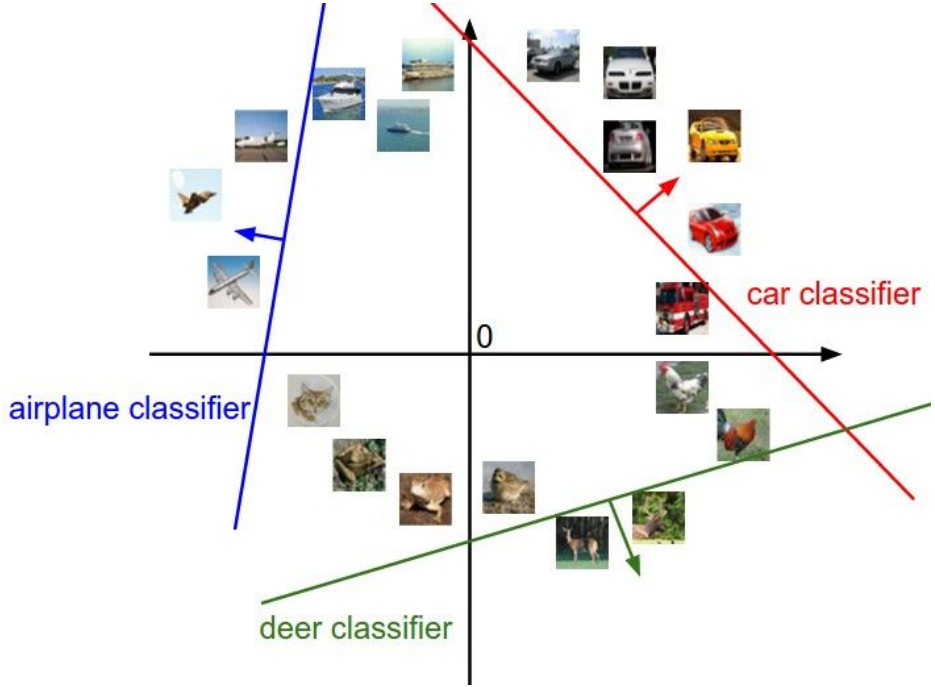
$Let: \boldsymbol{w} = \begin{bmatrix} p \\ q \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$\boldsymbol{wx} + b = 0$

Distance from point to line: $D = \frac{|px_0 + qy_0 + b|}{\sqrt{p^2 + q^2}} = \frac{\boldsymbol{w}^T \boldsymbol{x} + b}{|\boldsymbol{w}|}$

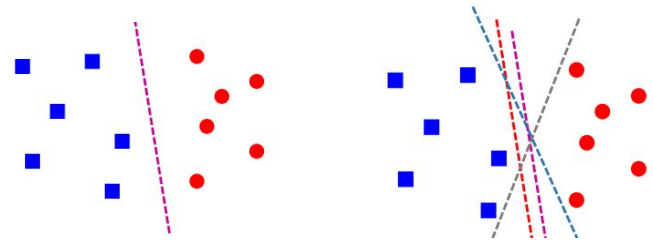
(Derived through vector projection)


$(x_0, y_0)$
D
w

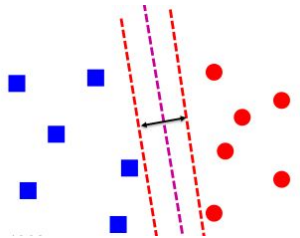

car classifier
0
airplane classifier
deer classifier

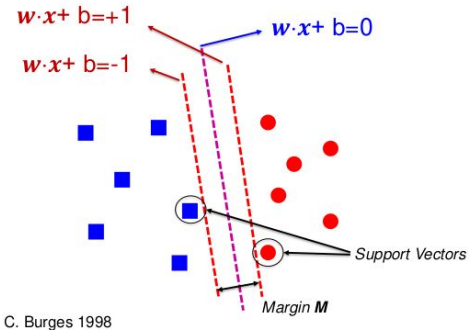

plane
car
bird
cat
deer
dog
frog
horse
ship
truck

# Support Vectors










Discriminative classifier based on optimal separating line (2D case)

- Plane and hyperplanes in higher dimensional spaces.
- Margin: The distance between the two classes

Maximize the margin between two classes

$x[i]$ Class A $(y[i] = 1)$ : w x + b ≥ 1

$x[i]$ Class B $(y[i] = -1)$: w x + b ≤ −1

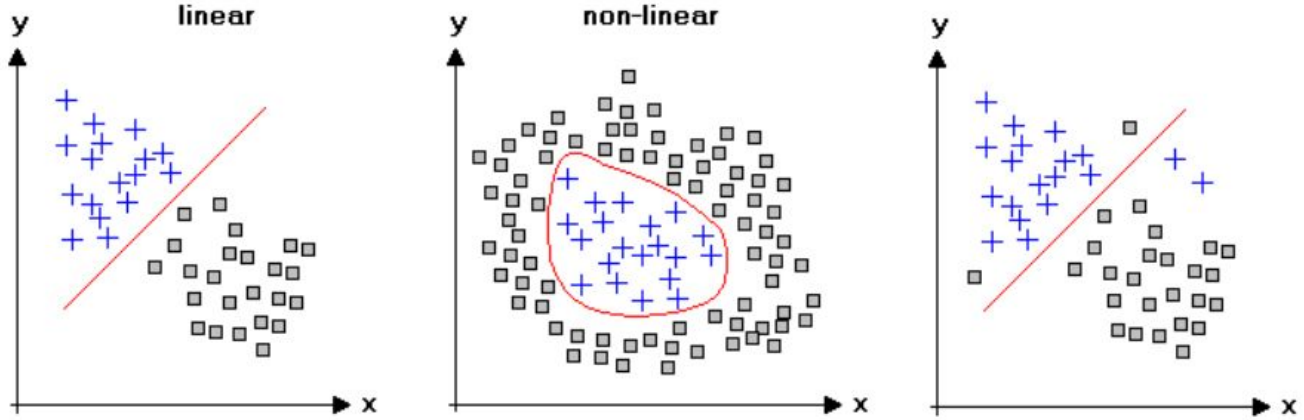
Dist. Between point and line: $\frac{|w^T x+b|}{\|w\|}$

For Support Vectors: $\frac{w^T x+b}{\|w\|} = \frac{\pm 1}{\|w\|}$

$$M = \left| \frac{1}{\|w\|} - \frac{-1}{\|w\|} \right| = \frac{2}{\|w\|}$$

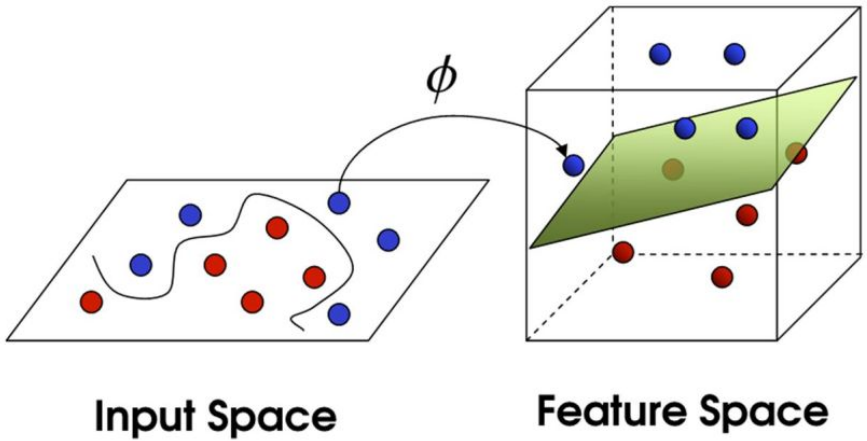
# Non-linear data

Datasets that are linearly separable with some noise work out great using slack variables or the hinge loss i.e., soft margin



# Non-linear data

- General idea: Original input space can be mapped to some higher-dimensional feature space…
- By projecting data to a higher dimensional space we have more tunable nobs by which to separate the classes.




$\Phi: x \rightarrow \varphi(x)$

…where the training set is separable

# Multi-class SVM

What if we have more than two categories?

Combine a number of binary classifiers

One vs. all

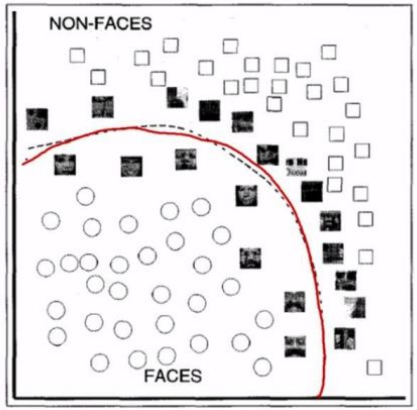
- Training: Learn SVM for each class vs. the rest
- Testing: Apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value.

One vs. one

- Training: Learn an SVM for each classes
- Testing: Apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value.

# SVM for recognition: Training

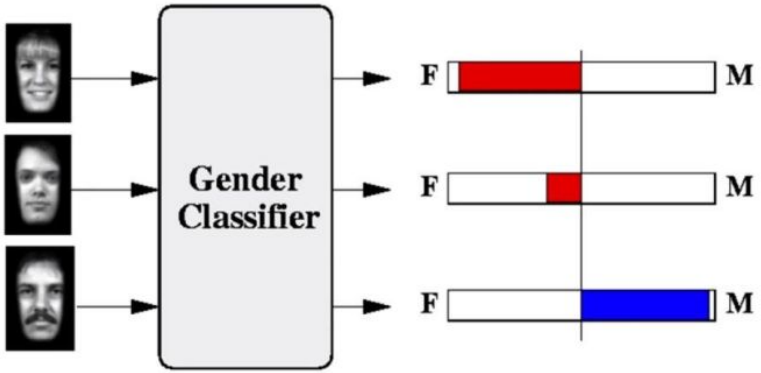
1. Define your representation
2. Select a kernel function
3. Compute the pairwise kernel values between labeled examples
4. Use the “kernel matrix” to solve for SVM support vectors and weights.



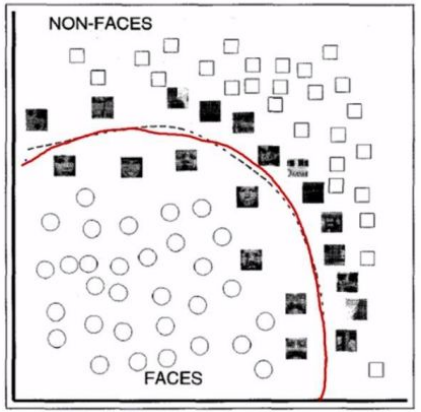

Not a line in the original space, but linear in a higher dimensional space

# SVM for recognition: Prediction

1. To classify a new example:
2. Compute the kernel values between new input and support vectors.
3. Apply weights.
4. Check the sign of output.




Learning Gender with Support Faces [Moghaddam and Yang, TPAMI 2002]

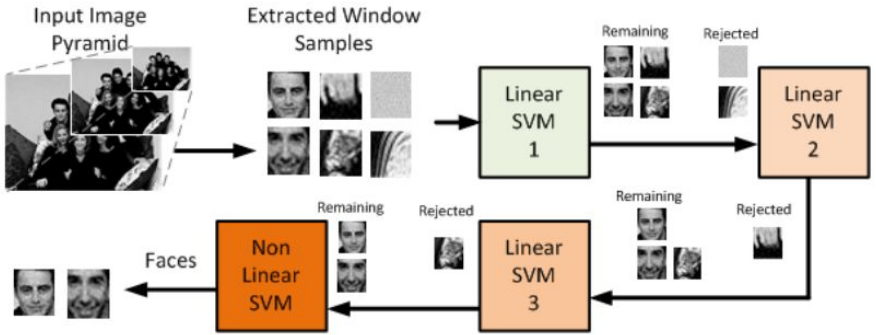



Training Support Vector Machines: an Application to Face Detection [Osuna et al, 1998]

# Making SVM faster

A hierarchy of SVM classifiers of increased complexity

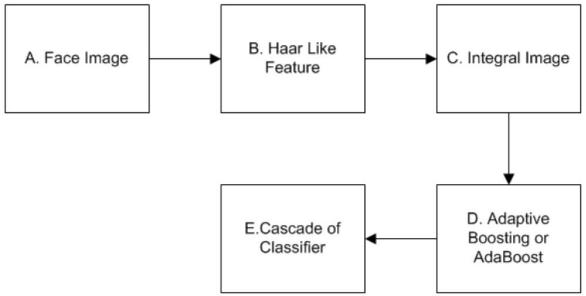
- Increasing number of SVs
- First stages are linear SVMs
  - Need to process only one vector
- One final non-linear Stage
  - 2nd degree polynomial

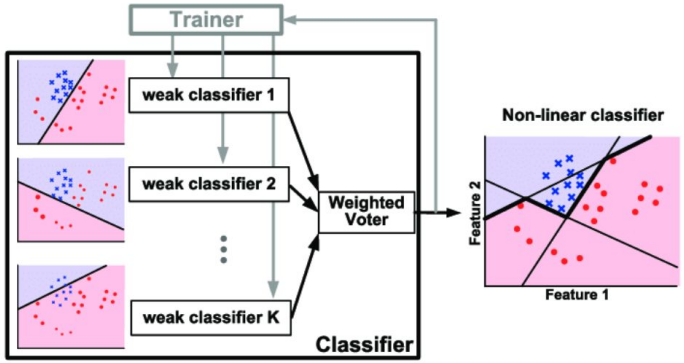



Christos Kyrkou, Christos-Savvas Bouganis, Theocharis Theocharides, Marios Polycarpou, "Embedded hardware-Efficient Real-Time Classification with Cascade Support Vector Machines", IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 1, pp. 99-112, January 2016.

# Viola-Jones

## Techniques




- Three major contributions/phases of the algorithm:
  - Feature Extraction
  - Classification using boosting
  - Multi-scale detection algorithm
- Feature extraction and feature evaluation
  - Rectangular features are used, with a new image representation which makes their calculation very fast
- Classifier training and feature selection using a single variation of a method called AdaBoost.
- A combination of simple classifiers is very effective

# Viola-Jones Face detection

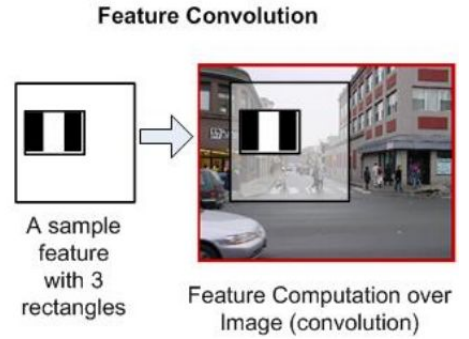
Want it to be very fast and accurate

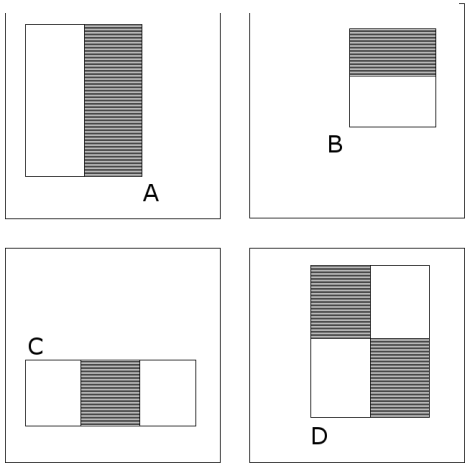
Run on a camera or cell phone, low cost

Use simple features and simple classifiers

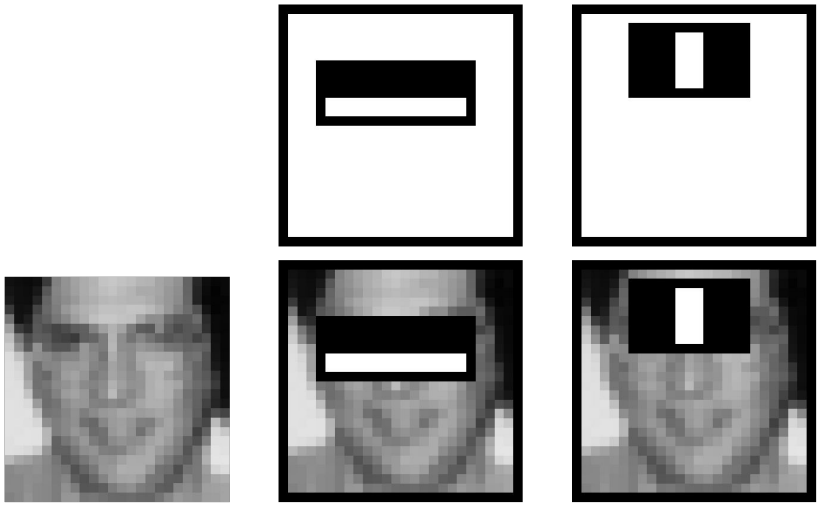
*Haar features*:

Response = Σ pix in black region - Σ pix in white region



# Feature Extraction



$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

| | | | |
|---|---|---|---|
| 10 | 20 | 10 | 20 |
| 20 | 10 | 10 | 10 |
| 30 | 10 | 10 | 20 |
| 10 | 20 | 30 | 20 |

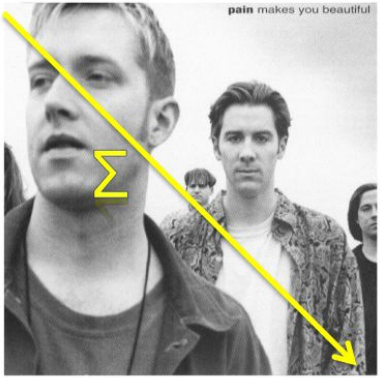
Original Image

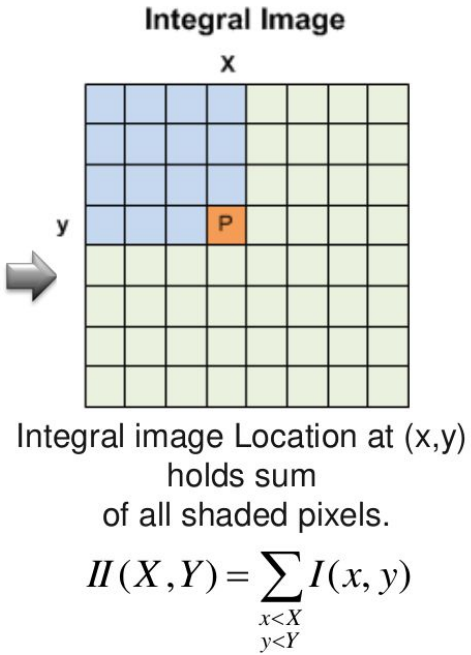
| | | | |
|---|---|---|---|
| 10 | 30 | 40 | 60 |
| 30 | 60 | 80 | 110 |
| 60 | 100 | 130 | 180 |
| 70 | 130 | 190 | 260 |

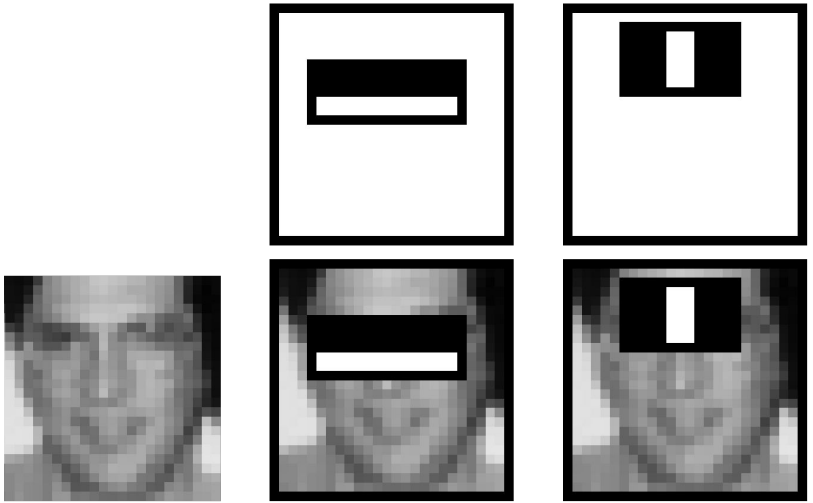
Integral Image

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

# Feature Extraction



https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

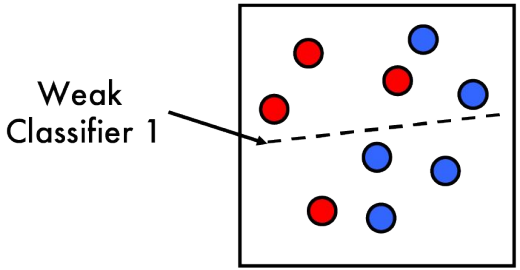
# Viola-Jones Face detection

Classifier: *boosted* partitions

*Boosting*

Way to make weak classifiers better

Train a weak classifier




https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

# Viola-Jones Face detection

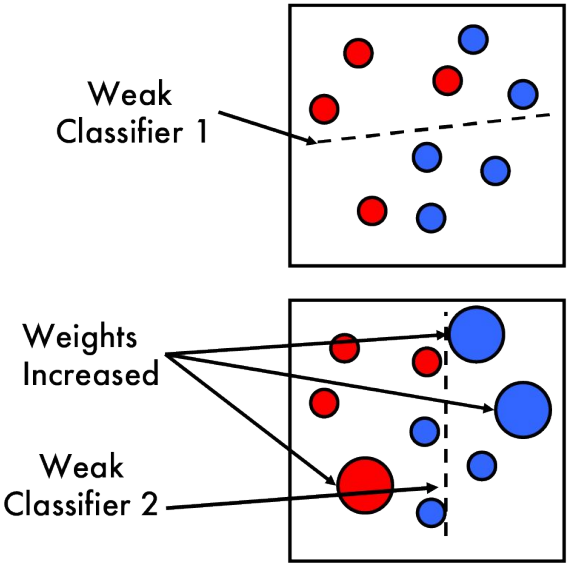
Classifier: *boosted* partitions

*Boosting*

Way to make weak classifiers better

Train a weak classifier

Reweight data we got wrong, train again



# Classifier: *boosted* partitions

## *Boosting*

Way to make weak classifiers better

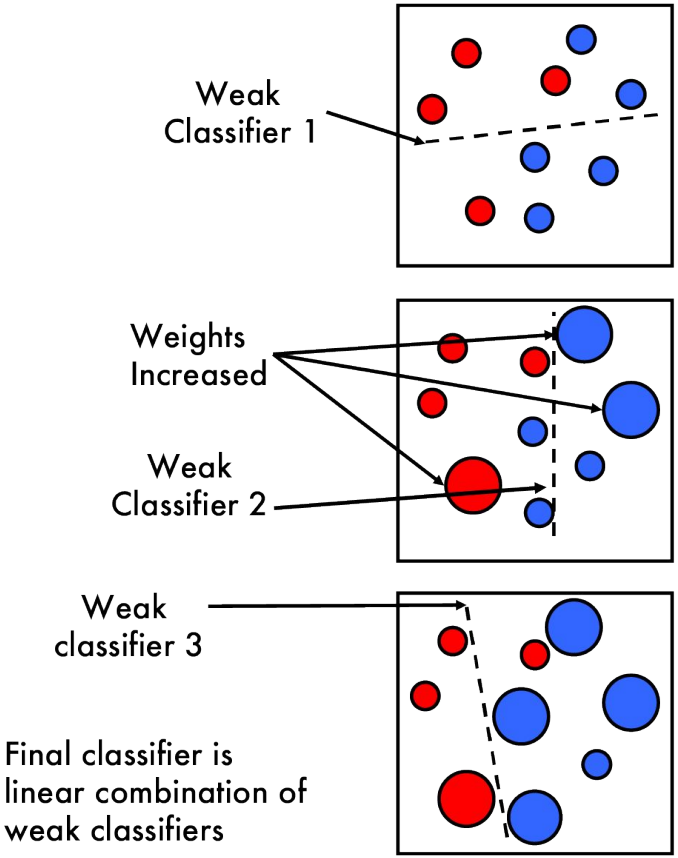
Train a weak classifier

Reweight data we got wrong, train again

...and again

Until you feel like stopping

Final classifier is combination of all



# Viola-Jones Face detection

Finally, use a **cascade of classifiers**

1st classifier

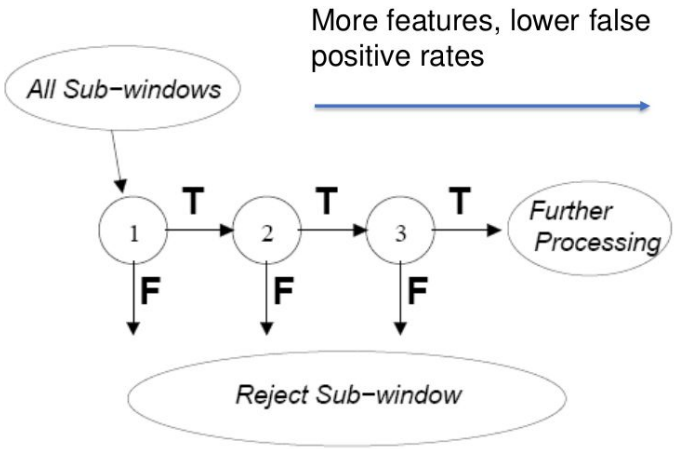
Very fast, throws out easy negatives

2nd classifier

Fast, throws out harder negatives

3rd classifier

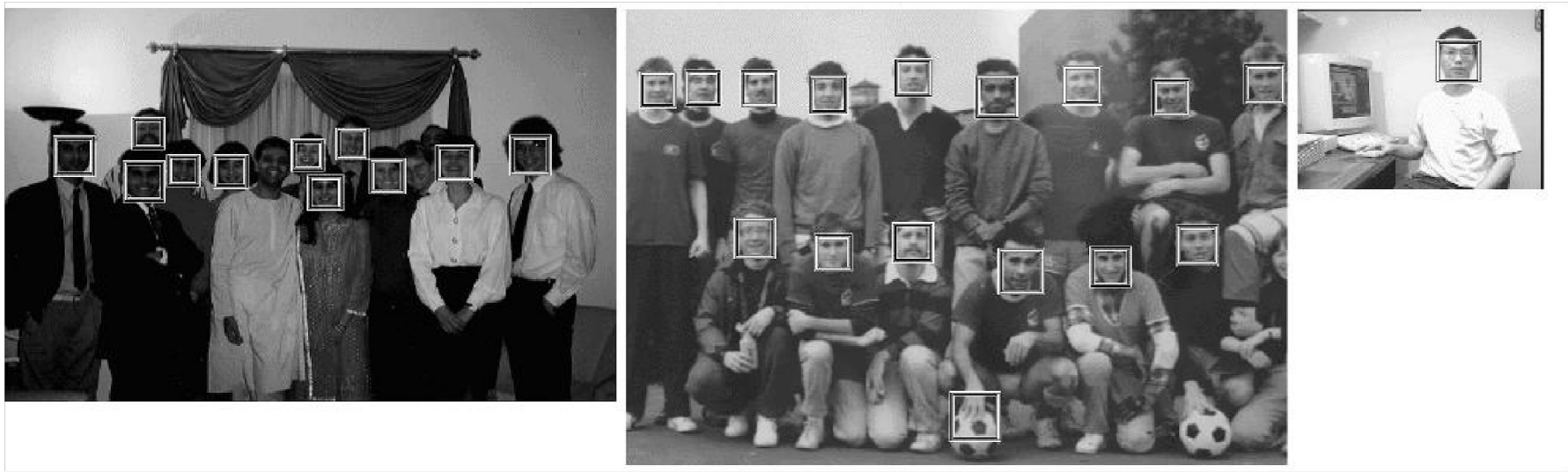
Slower, throws out hard negatives



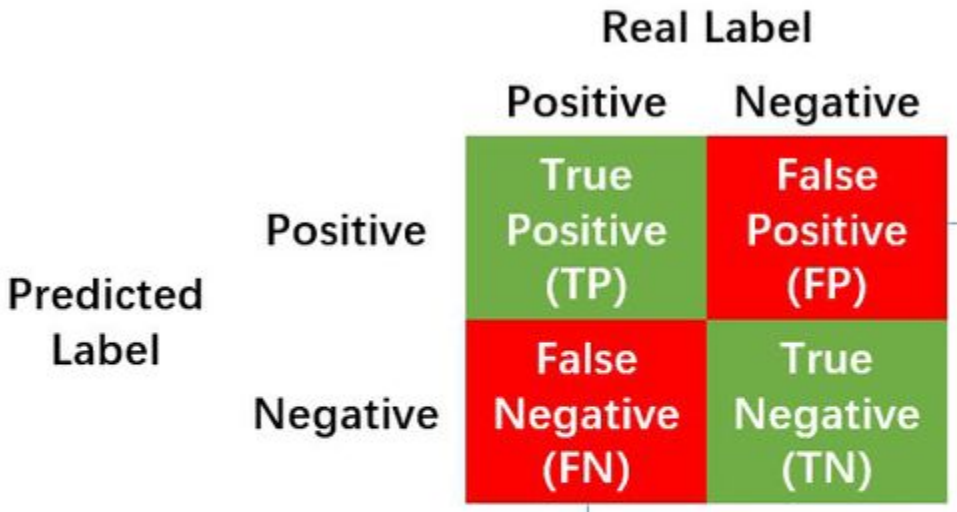

Only run slow, good classifiers on hard examples

Fast classifier that is still very accurate

# Viola-Jones Face detection



# Evaluation Metrics




$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

# Q & A

Thank You !