Paul Viola
Microsoft Research
viola@microsoft.com

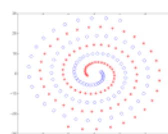
Overview

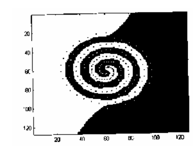
- Perceptrons
- Support Vector Machines
  - Face and pedestrian detection
- AdaBoost
  - Faces
- Building Fast Classifiers
  - Trading off speed for accuracy…
  - Face and object detection
- Memory Based Learning
  - Simard
  - Moghaddam

History Lesson

- 1950's Perceptrons are cool
  - Very simple learning rule, can learn "complex" concepts
  - Generalized perceptrons are better -- too many weights
- 1960's Perceptron's stink (M+P)
  - Some simple concepts require exponential # of features
    - Can't possibly learn that, right?
- 1980's MLP's are cool (R+M / PDP)
  - Sort of simple learning rule, can learn anything (?)
  - Create just the features you need
- 1990 MLP's stink
  - Hard to train : Slow / Local Minima
- 1996 Perceptron's are cool

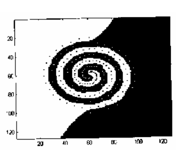
Why did we need multi-layer perceptrons?





- Problems like this seem to require very complex non-linearities.
- Minsky and Papert showed that an exponential number of features is necessary to solve generic problems.

Why an exponential number of features?



14th Order???
120 Features

$$\Phi(x) = \left\{ \begin{array}{c} 1 \\ x_1^1, x_2^1 \\ x_1^2, x_1^1x_2^1, x_2^2 \\ x_1^3, x_1^2x_2^1, x_1^1x_2^2, x_2^3 \\ x_1^4, x_1^3x_2, x_1^2x_2^2, x_1^2x_2^2, x_1x_2^3, x_2^4 \\ x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5 \ldots \\ \vdots \end{array} \right\}$$

n: variables
k: order poly

$\binom{n+k}{k}$ $(n+k)!$ … $k^n))$

N=21, k=5 --> 65,000 features

MLP's vs. Perceptron

- MLP's are hard to train…
  - Takes a long time (unpredictably long)
  - Can converge to poor minima
- MLP are hard to understand
  - What are they really doing?
- Perceptrons are easy to train…
  - Type of linear programming. Polynomial time.
  - One minimum which is global.
- Generalized perceptrons are easier to understand.
  - Polynomial functions.

$y : \{+1, -1\}$
$\mathbf{x} : R^N$
$\{(y_i, \mathbf{x}_i)\}$

Perceptron Training is Linear Programming

$$\forall i: \; y_i\left(w^T \mathbf{x}_i\right) > 0$$

Polynomial time in the number of variables and in the number of constraints.

What about linearly inseparable?

$$\forall i: \; y_i\left(w^T \mathbf{x}_i\right) + s_i > 0 \qquad \min \sum_i s_i$$

$$s_i > 0 \quad \forall i$$

Rebirth of Perceptrons

- How to train effectively
  - Linear Programming (… later quadratic programming)
  - Though on-line works great too.
- How to get so many features inexpensively?!?
  - Kernel Trick
- How to generalize with so many features?
  - VC dimension. (Or is it regularization?)

Support Vector Machines

Lemma 1: Weight vectors are simple

$$w_0 = 0 \qquad \Delta w_\tau = \eta_\tau \, \mathbf{x}_\tau$$

$$w_\tau = \sum_{t<\tau} \eta_t \mathbf{x}_t = \sum_l b_l \mathbf{x}_l \qquad w_\tau = \sum_l b_l \Phi(\mathbf{x}_l)$$

- The weight vector lives in a sub-space spanned by the examples…
  - Dimensionality is determined by the number of examples not the complexity of the space.

Lemma 2: Only need to compare examples

$$w_\tau = \sum_l b_l \Phi(\mathbf{x}_l)$$

$$\begin{aligned} y(\mathbf{x}) &= w^T \Phi(\mathbf{x}) \\ &= \left(\sum_l b_l \Phi(\mathbf{x}_l)\right)^T \Phi(\mathbf{x}) \\ &= \sum_l b_l \Phi(\mathbf{x}_l)^T \Phi(\mathbf{x}) \\ &= \sum_l b_l K(\mathbf{x}_l, \mathbf{x}) \end{aligned}$$

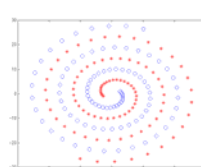
Simple Kernels yield Complex Features

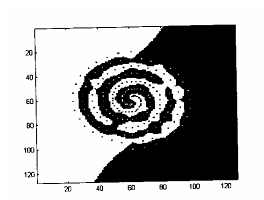
$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^T \mathbf{x}')^2 \\ &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2 x_1 x_1' x_2 x_2' \end{aligned}$$

$$\Phi(x) = \begin{Bmatrix} 1 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{Bmatrix}$$

But Kernel Perceptrons Can Generalize Poorly

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^{14}$$





Perceptron Rebirth: Generalization

- Too many features … Occam is unhappy
  - Perhaps we should encourage smoothness?

$$\forall i: \quad y_i \sum_j b_j K(\mathbf{x}_j, \mathbf{x}_i) + s_i > 0 \qquad \min \sum_i s_i$$

$$\forall i: \quad s_i > 0$$

$$\boxed{\min \sum_j b_j^2}$$

Smoother

Linear Program is not unique

The linear program can return any multiple of the correct weight vector...

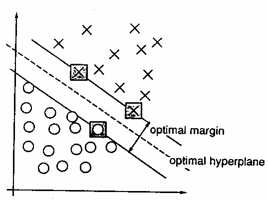
$$\forall i: \; y_i\left(w^T \mathbf{x}_i\right) > 0 \quad ➡ \quad \forall i: \; y_i(\lambda w)^T \mathbf{x}_i > 0$$

Slack variables & Weight prior
- Force the solution toward zero

$$y_i\left(w^T \mathbf{x}_i\right) + s_i > 0 \quad \forall i \qquad \min \sum_i s_i$$

$$s_i > 0 \quad \forall i \qquad \min w^T w$$

Definition of the Margin



- Geometric Margin: Gap between negatives and positives measured perpendicular to a hyperplane
- Classifier Margin $\quad \min_{i \in POS}\left(w^T \mathbf{x}_i\right) - \max_{i \in NEG}\left(w^T \mathbf{x}_i\right)$

Require non-zero margin

$$w^T \mathbf{x}_l + s_l > 0 \quad \forall l$$

Allows solutions with zero margin

$$w^T \mathbf{x}_l + s_l > 1 \quad \forall l$$

Enforces a non-zero margin between examples and the decision boundary.

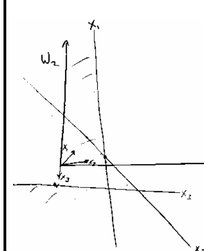
Constrained Optimization

$$y_l \sum_j b_j K(\mathbf{x}_j, \mathbf{x}_l) + s_l > 1 \qquad \min \sum_l s_l$$

$$s_l > 0 \qquad \min \sum_j b_j^2$$

- Find the smoothest function that separates data
  - Quadratic Programming (similar to Linear Programming)
    - Single Minima
    - Polynomial Time algorithm

Constrained Optimization 2



$$\min W^T W$$

$$W^T X^1 \geq 1$$

$$W^T X^2 \geq 1$$

$$W^T X^3 \leq -1$$

$$W^T X^1 = 1$$

$$W = \alpha_1 X^1 + \beta \overset{\perp}{X^1}$$

$$W^T X^2 = 1$$

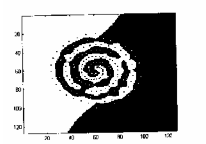
$$W = \alpha_2 X^2 + \beta_2 \overset{\perp}{X^2}$$

$$\begin{bmatrix} - X_1 - \\ - X_2 - \end{bmatrix} \begin{bmatrix} W \end{bmatrix} = \overset{One}{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}$$

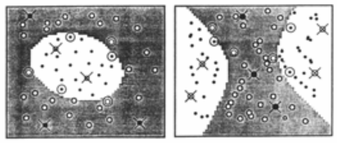
$$W = One \setminus X$$

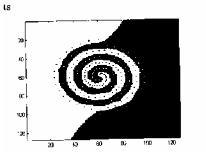
x^3 *is inactive*

SVM: examples



e 1: Kernel-Perceptron (small margin) clear
ts

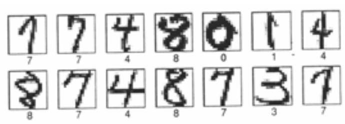




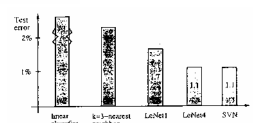
SVM: Key Ideas

- Augment inputs with a very large feature set
  - Polynomials, etc.
- Use Kernel Trick(TM) to do this efficiently
- Enforce/Encourage Smoothness with weight penalty
- Introduce Margin
- Find best solution using Quadratic Programming

SVM: Zip Code recognition

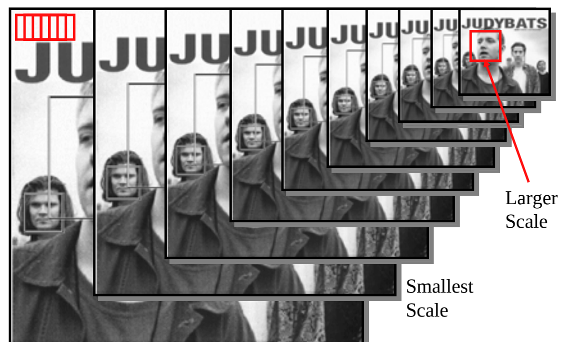


- Data dimension: 256
- Feature Space: 4 th order
  - roughly 100,000,000 dims



	Cl. 0	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Cl. 9
Supp. patt.	1379	989	1958	1900	1224	2024	1527	2064	2332	2765
Error train	7	16	8	11	2	4	8	16	4	1
Error test	19	14	35	35	36	49	32	43	48	63

The Classical Face Detection Process



50,000 Locations/Scales

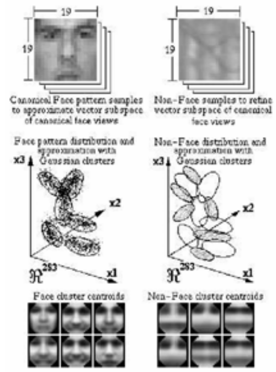
Classifier is Learned from Labeled Data

- Training Data
  - 5000 faces
    - All frontal
  - 10^8 non faces
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose (rotation both in plane and out)

Key Properties of Face Detection

- Each image contains 10 - 50 thousand locs/scales
- Faces are rare 0 - 50 per image
  - 1000 times as many non-faces as faces
- Extremely small # of false positives: 10^{-6}

Sung and Poggio



Rowley, Baluja & Kanade

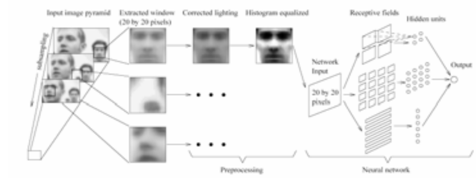
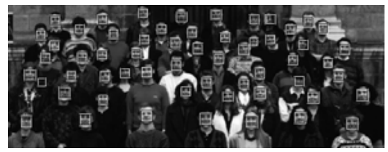


Figure 1: The basic algorithm used for face detection.

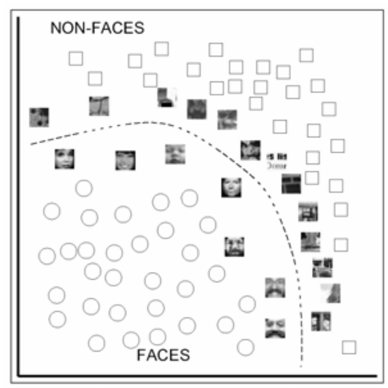


First Fast System
- Low Res to Hi

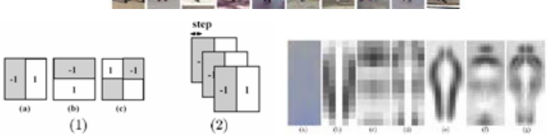
Osuna, Freund, and Girosi

	Test Set A		Test Set B	
	Detect Rate	False Alarms	Detect Rate	False Alarms
SVM	97.1 %	4	74.2%	20
Sung *et al.*	94.6 %	2	74.2%	11

Support Vectors



P, O, & G: First Pedestrian Work



On to AdaBoost

- Given a set of weak classifiers

  originally: $h_j(\mathbf{x}) \in \{+1, -1\}$

  also $h_j(\mathbf{x}) \in \{\alpha, \beta\}$ "confidence rated"

  - None much better than random
- Iteratively combine classifiers
  - Form a linear combination

  $$C(x) = \theta\left(\sum_t h_t(x) + b\right)$$

  - Training error converges to 0 quickly
  - Test error is related to training margin

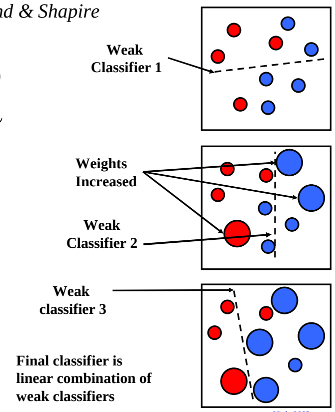
AdaBoost

Freund & Shapire

$$h_t = \min_h \underbrace{\sum_i D_t(i)\, e^{-y_i h(x_i)}}_{Z_t}$$

$$\{\alpha_t, \beta_t\} = \frac{1}{2}\log\left(\frac{W_+}{W_-}\right)$$

$$D_{t+1}(i) = \frac{D_t(i) e^{-y_i h_t(x_i)}}{Z_t}$$



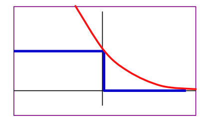
Viola 2003

AdaBoost Properties

$$D_{t+1}(i) = \frac{D_t(i) e^{-y_i h_t(x_i)}}{Z_t} = \frac{\prod_t e^{-y_i h_t(x_i)}}{\prod_t Z_t} = \frac{e^{-y_i \sum_t h_t(x_i)}}{\prod_t Z_t}$$

$$\prod_t Z_t = \sum_i e^{-y_i \sum_t h_t(x_i)} \qquad e^{-y_i \sum_t h_t(x_i)} \geq Loss(y_i, C(x_i))$$

$$\prod_t Z_t = \sum_i e^{-y_i \sum_t h_t(x_i)} > \sum_i Loss(y_i, C(x_i))$$



Viola 2003

AdaBoost:
Super Efficient Feature Selector

- Features = Weak Classifiers
- Each round selects the optimal feature given:
  - Previous selected features
  - Exponential Loss

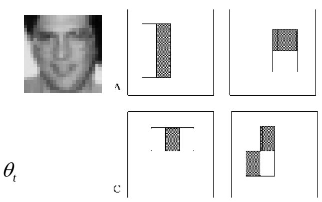
Viola 2003

Boosted Face Detection: Image Features

"Rectangle filters"

Similar to Haar wavelets

Papageorgiou, et al.

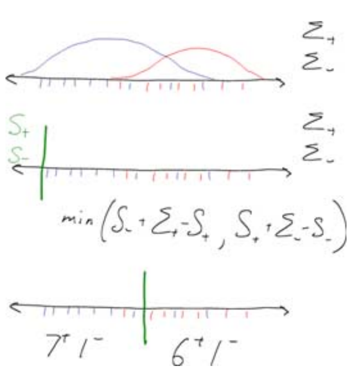


$$h_t(x_i) = \begin{cases} \alpha_t & \text{if } f_t(x_i) > \theta_t \\ \beta_t & \text{otherwise} \end{cases}$$

$$C(x) = \theta\left(\sum_t h_t(x) + b\right)$$

$60{,}000 \times 100 = 6{,}000{,}000$ Unique Binary Features

Viola 2003

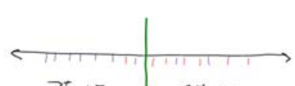


Viola 2003

$$h_t = \min_h \underbrace{\sum_i D_t(i)\, e^{-y_i h(x_i)}}_{Z_t}$$

$$\{\alpha_t, \beta_t\} = \frac{1}{2}\log\left(\frac{W_+}{W_-}\right)$$

$$D_{t+1}(i) = \frac{D_t(i) e^{-y_i h_t(x_i)}}{Z_t}$$



$7^+\ 1^-$ $\qquad$ $6^+\ 1^-$

$\alpha = \frac{1}{2}\log\left(\frac{7}{1}\right)$ $\qquad$ $\beta = \frac{1}{2}\log\left(\frac{1}{6}\right)$

$e^{-\frac{1}{2}\log\left(\frac{1}{7}\right)}$ $\qquad$ $e^{\frac{1}{2}\log\left(\frac{1}{7}\right)}$

$7 \times \sqrt{1/7}$ $\qquad$ $1 \times \sqrt{7/1}$

$\sqrt{7}$ $\qquad$ $\sqrt{7}$

Viola 2003

Feature Selection

- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min Z)
  - Select best filter/threshold (= Feature)
  - Reweight examples
- M filters, T thresholds, N examples, L learning time
  - $O(MT\,L(MTN))$ Naïve Wrapper Method
  - $O(MN)$ *Adaboost* feature selector

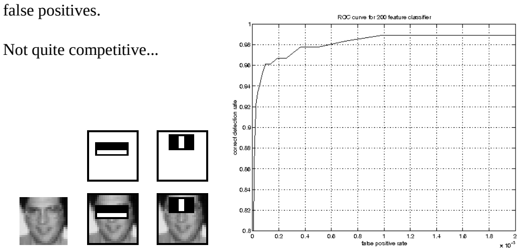
Viola 2003

Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

Not quite competitive...

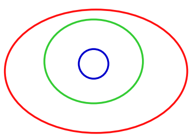


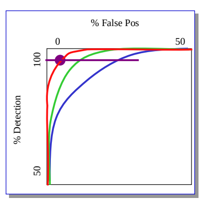
ROC curve for 200 feature classifier

Viola 2003

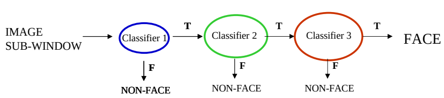
Building Fast Classifiers

- Given a nested set of classifier hypothesis classes





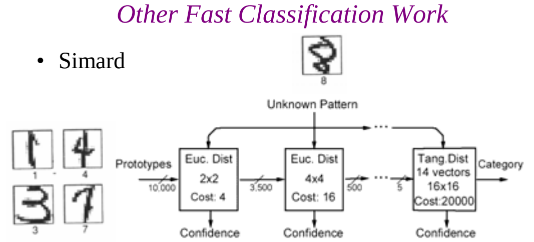
- Computational Risk Minimization



Viola 2003

Other Fast Classification Work

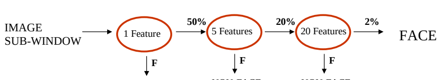
- Simard



- Rowley (Faces)
- Fleuret & Geman (Faces)

Viola 2003

Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

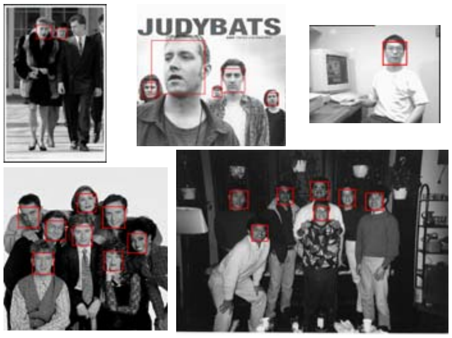
Viola 2003

Comparison to Other Systems

Detector \ False Detections	10	31	50	65	78	95	110	167	422
Viola-Jones	78.3	85.2	88.8	90.0	90.1	90.8	91.1	91.8	93.7
Rowley-Baluja-Kanade	83.2	86.0				89.2		90.1	89.9
Schneiderman-Kanade				94.4					
Roth-Yang-Ahuja					(94.8)				

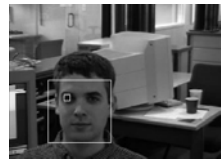
Viola 2003

Output of Face Detector on Test Images

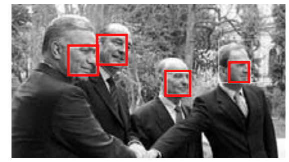


Viola 2003

Solving other "Face" Tasks

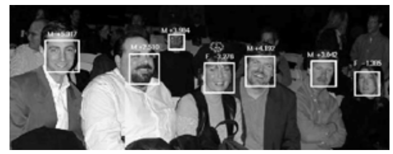


Facial Feature Localization



Profile Detection

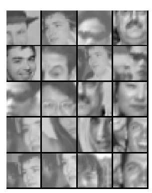
Demographic Analysis



Viola 2003

Feature Localization

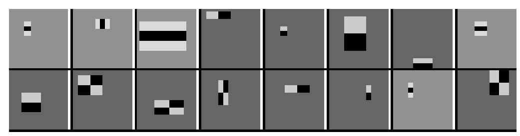
- Surprising properties of our framework
  - The cost of detection is not a function of image size
    - Just the number of features
  - Learning automatically focuses attention on key regions
- Conclusion: the "feature" detector can include a large contextual region around the feature



Viola 2003

Feature Localization Features

- Learned features reflect the task



Viola 2003

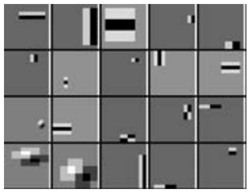
Profile Detection

Viola 2003

More Results

Viola 2003

Profile Features



Viola 2003

Features, Features, Features

- In almost every case:

  Good Features beat Good Learning

  Learning beats No Learning

- Critical classifier ratio: $\frac{\text{quality}}{\text{complexity}}$

  - AdaBoost >> SVM

Viola 2003