

**www.GetPedia.com**

\* The Ebook starts from the next page : Enjoy !



# *6.891 Machine Learning and Neural Networks*

## *Lecture 1: Introduction and Examples*

6.891 Machine Learning

## *News*

- First problem set is available (short!)
  - » Due Sept 16
  - » All psets are due on Thursday
  - » Normally you will have two weeks
- Reading: DHS Ch 1-2.3 (for Friday)

6.891 Machine Learning

## *Review & Overview*

- Administrative information
- Course Goals
- Define Learning, Induction, Regression, Classification
- Give examples of learning applications
- Bayes Rule and classification
- Regression and Overfitting
- Ockham's Razor, Curse of Dimensionality
  - » Brief Mention of Probability

6.891 Machine Learning

## *Course Information*

- <http://www.ai.mit.edu/courses/6.891>
- Lecturer: Paul Viola
  - » Prof in the AI Lab NE43-773, x3-8828
  - » [viola@ai.mit.edu](mailto:viola@ai.mit.edu)
  - » Research: Learning and Computer Vision
    - <http://www.ai.mit.edu/projects/lv>
- TA: Kinh Tieu
  - » Ph.D. student in the AI Lab NE43-771, x3-7547
  - » [tieu@ai.mit.edu](mailto:tieu@ai.mit.edu)
  - » Research: Image Database Retrieval, Vision, Learning
    - <http://www.ai.mit.edu/people/tieu>

6.891 Machine Learning

## *Grading Experiment!!!*

- Problem sets will be self-graded (mostly)
- You will hand in the pset on Thursday. Kinh will record its presence (or absence), and glance to see if you attempted each problem.
- We will distribute the psets (on Friday) at random to the class. You will each grade one pset with help from a solution key. You have 6 days.
- Kinh will lead a 1 hour pset review session to go over correct solutions. Probably Monday afternoon.
- You will hand back the graded psets on Wednesday.
- Kinh will then grade 1 question (usually the toughest).
- The graded psets will be returned to you on Friday, 8 days after you turned them in.

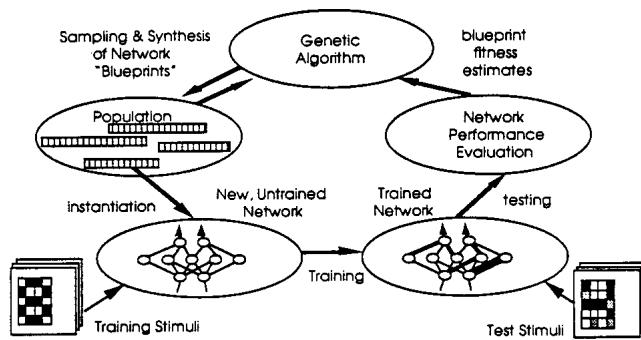
6.891 Machine Learning

## *Course Goals*

- Introduce, Motivate and Study concepts from machine learning. Focus both on fundamentals and applications.
  - » Second Time: *Watch Out!*
- Fundamentals:
  - » Follow text: Duda, Hart & Stork (from the Web page)
  - » Plus some supplemental handouts
- Applications:
  - » Read papers from literature.
- Reinforce:
  - » Six PSETs will require both thinking and hacking.
  - » One final project.
  - » Mid-term
  - » Final exam (??)

6.891 Machine Learning

## *Course Goals: NOT*

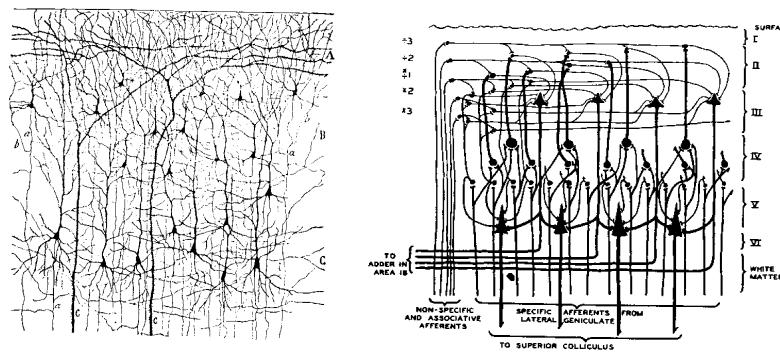


**Figure 1:** A population of network "blueprints" is cyclically updated by the genetic algorithm based on their fitness.

NIPS 1989

6.891 Machine Learning

## *Course Goals: BUT*



Pitts and McCulloch, 1947

6.891 Machine Learning

## Goals: Analysis and Computation

$$\begin{aligned}\bar{x} &= \int_V dy \int x\phi(x,y)dx, \\ \bar{y} &= \int_V dy \int y\phi(x,y)dx,\end{aligned}\quad (4)$$

where integration is over the whole visual field  $V$ . If  $\xi_R, \eta_R$  are respectively sagittal and lateral coordinates measuring position on the right colliculus  $C_R$ , and  $\xi_L$  and  $\eta_L$  their mirror images on the left colliculus  $C_L$ , there will be a mapping

$$x = x_R = p(\xi_L, \eta_L), \quad (5)$$

$$y = y_R = q(\xi_L, \eta_L), \quad \text{if } x > 0,$$

and

$$x = x_L = p(\xi_R, \eta_R),$$

$$y = y_L = q(\xi_R, \eta_R), \quad \text{if } x \leq 0.$$

To transform equations (4) into the coordinates of the colliculus will then yield

$$\begin{aligned}\bar{x} &= \bar{x}_R - \bar{x}_L, \\ \bar{y} &= \bar{y}_R + \bar{y}_L, \\ \bar{x}_R &= \int_{C_L} \int \Phi_L(\xi, \eta) p(\xi, \eta) J(\xi, \eta) d\xi d\eta, \\ \bar{y}_R &= \int_{C_L} \int \Phi_L(\xi, \eta) q(\xi, \eta) J(\xi, \eta) d\xi d\eta, \\ \bar{x}_L &= \int_{C_R} \int \Phi_R(\xi, \eta) p(\xi, \eta) J(\xi, \eta) d\xi d\eta, \\ &\quad \vdots\end{aligned}$$

vanish with them. For in any case, the eyes must be moved in such a direction as to diminish  $(u, v)$ , and *pari passu*  $(\bar{x}, \bar{y})$ , and finally they must remove  $(u, v)$ , and therefore  $(\bar{x}, \bar{y})$  to the origin at the visual axes. Thus, if the two quantities computed from  $\phi(x, y)$  to determine lateral and vertical motion respectively have the form

$$u = u_R - u_L, \quad v = v_R + v_L, \quad (6)$$

$$u_R = \int_{C_L} \int U(\xi, \eta) \Phi_L(\xi, \eta) d\xi d\eta,$$

$$v_R = \int_{C_L} \int V(\xi, \eta) \Phi_L(\xi, \eta) d\xi d\eta,$$

with a similar integral with  $\Phi_R$  for  $u_L$  and  $v_L$ , and any  $U$  and  $V$  fulfilling the condition that for every  $\eta$ ,  $U(\xi, \eta)$  is properly monotonic in  $\eta$ , and for every  $\xi$ ,  $V(\xi, \eta)$  is properly monotonic in  $\eta$ , then  $u$  and  $v$  will have the required properties that they shall vanish and vary monotonically with  $x$  and  $y$  respectively. J. Apter (1945, 1946) shows that one can write, approximately,

$$x = p(\xi), \quad (7)$$

$$y = q(\eta),$$

with  $p(\xi)$  and  $q(\eta)$  both properly monotonically increasing, neglecting the other variable. This would yield

$$U(\xi, \eta) = p(\xi)p'(\xi)q'(\eta),$$

$$V(\xi, \eta) = q(\eta)q'(\eta)p'(\xi),$$

so that

$$u_R = \int_{C_L} \int p(\xi) dm(\xi) \int \Phi_L(\xi, \eta) \sigma(\eta) d\eta,$$

6.891 Machine Learning

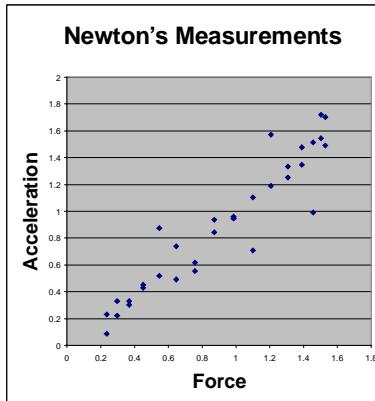
## What is Machine Learning?

- Induction of patterns, regularities, and rules from data.
  - » Leap to conclusions.
- Not Deduction
  - » {axioms, assumptions, rules}  $\rightarrow$  theorems
- Induction
  - » {tons of data}  $\rightarrow$  rules, axioms, laws
- Classic Examples:
  - » Newton's Laws, Kepler's Laws
  - » Periodic Table
  - » Mendel's laws of inheritance

6.891 Machine Learning

## Physical Laws

- Observe many experiments:

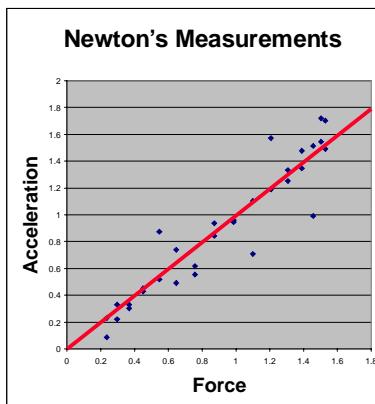


- Conjecture simple Rule:

6.891 Machine Learning

## Physical Laws: Theorize

- Observe many experiments:



- Conjecture simple Rule:  
»  $F=ma$

Ignore Errors & Inconsistencies

6.891 Machine Learning

## Different Types of Learned Relations

- Regression

  - » Continuous input; Continuous output

    - $F = ma, pv = nrt$
    - Interest Rates  $\rightarrow$  Stock Prices
    - Inches Rain  $\rightarrow$  Corn Production

- Classification

  - » Discrete input; Discrete output

    - {Red? & Round? & SmallSeed?}  $\rightarrow$  Apple
    - Alarm?  $\rightarrow$  BreakIn, Alarm? & Earthquake?  $\rightarrow$  NoBreakIn

  - » Continuous input; Discrete output

    - Midterm  $\rightarrow$  Final Grade
    - {Fever, Blood Pressure}  $\rightarrow$  Sick?
    - {Income, Current Debt}  $\rightarrow$  Issue Loan?
    - Sound  $\rightarrow$  Words, Images  $\rightarrow$  People

6.891 Machine Learning

## Some Notation

- In General a learning problem will have:

  - » Inputs:

$$\mathbf{x} = (x_1, x_2, \dots, x_d)^T$$

$$x^j \text{ or } \mathbf{x}^j$$

j-th example

  - » Outputs:

$$C = \{C_1, C_2, \dots\}$$

$$C^j \text{ & } y^j$$

$$\mathbf{y} = (y_1, y_2, \dots)^T$$

  - » Target (Correct label or value)

$$t^j$$

6.891 Machine Learning

## *Additional Notation (Abusive!)*

- Prediction Function

$$y^j = y(x^j) \quad C^j = C(x^j)$$

- Error Function

$$E = \sum_j l(t^j - y(x^j))$$

$$l(z) = \begin{cases} 0 & \text{if } z = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$l(z) = z^2$$

Loss

6.891 Machine Learning

## *Example: Digit Recognition*

- US Postal Service -- 100 Million Letters a day

- Reading Zip Codes

- » First find the Address Block
- » Then the zipcode
- » Normalize size and rotation
- » Separate digits
- » Digitize --  $10 \times 10 \rightarrow 2^{100}$  possible images

6.891 Machine Learning

## Character Recognition



FIG. 3. Mark I Perceptron at Cornell Aeronautical laboratory. (a) Overall view with sensory input at left, association units in center, and control panel and response units at far right. The sensory to associator plugboard, shown in (b) is located behind the closed panel to the right of the operator. The image of the letter "C" on the front panel is a repeater display, for monitoring sensory inputs.

6.891 Machine Learning

## Zip Code Recognition

Paul Viola  
545 Technology Sq  
Cambridge MA 02139

6.891 Machine Learning

## *Tremendous Variety*

40004      75216  
14199-2087 23505  
96205      14310  
44151      05753

6.891 Machine Learning

## *Hand Labeled Data*

1 4 1 6 1 1 9 1 5 4 8 5 7 9 6 8 0 3 2 4 6 4 1 4 1  
8 6 6 3 5 9 7 2 0 2 9 9 2 9 9 7 2 2 5 1 0 0 4 6 7  
0 1 3 0 8 4 1 1 1 5 9 1 0 1 0 6 1 5 4 0 6 1 0 3 6  
3 1 1 0 6 4 1 1 1 0 3 0 4 7 3 2 6 2 0 0 9 9 7 9 9  
6 6 8 9 1 2 0 8 6 7 0 8 5 5 7 1 3 1 4 2 2 9 5 5 4  
6 0 1 0 1 7 2 3 0 1 8 7 1 1 2 9 9 1 0 8 9 9 7 0 9  
8 4 0 1 0 9 7 0 7 5 9 7 3 3 1 9 7 2 0 1 5 5 1 9 0  
5 5 1 0 7 5 5 1 8 2 5 5 1 8 2 8 1 4 3 8 0 9 0 9  
4 3 1 7 8 7 5 4 1 6 5 5 4 6 0 5 5 4 6 0 3 5 4 6 0  
5 5 1 8 2 5 5 1 0 8 5 0 3 0 4 7 5 2 0 4 3 9 4 0 1

9 6 2 7 3

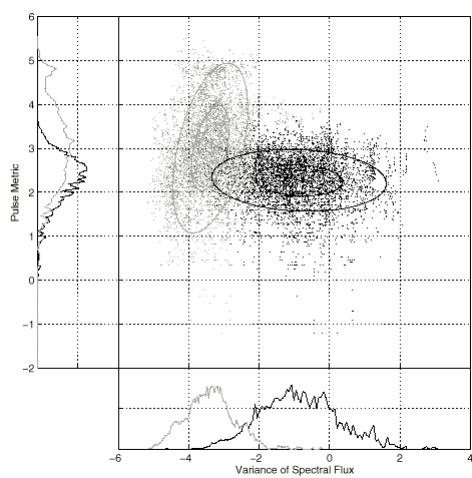
6.891 Machine Learning

## *Final Performance*

- US Postal Service -- 100 Million Letters a day
- Reading Zip Codes
  - » First find the Address Block
  - » Then the zipcode
  - » Normalize size and rotation
  - » Separate digits
  - » Digitize --  $10 \times 10 \rightarrow 2^{100}$  possible images
- Training: 400,000 example images
- Final Performance > 98%

6.891 Machine Learning

## *Differentiating Speech & Music*

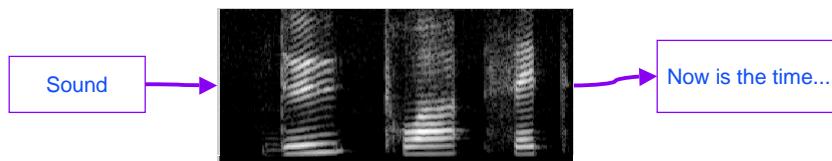


The Key Issue:  
Features!

6.891 Machine Learning

## *Speech Recognition*

- **Speech recognition:**
  - » Sound signals → cepstral coefficients → Word Sequence
  - » 10,000 / sec → 10 frames / sec → 5 words / sec



- **Key difficulties**
  - » Variations in pitch, pronunciation, speed

6.891 Machine Learning

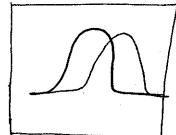
## *Evaluation of Credit Risk*

- **Feature vector:**
  - » Income level, time at current job, time at previous job, marital status, children?, paid previous bills?, own home?, location of home, etc.
- **Many thousands or millions of examples**
  - » Feature vector + outcome of loan
- **Difficulties:**
  - » Noise
  - » Insufficient data
  - » Missing data
  - » Generalization

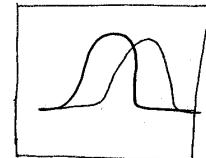
6.891 Machine Learning

## Digit Recognition in Detail

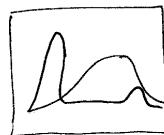
- Classifying 1 vs 2
- Define a set of features:



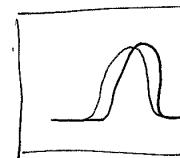
Perim



Num Black Pixels



Width



Height

- Look for separation

6.891 Machine Learning

## Rules for Classification

- Many schemes for classifying data:

- » Pick a threshold

$$C(x) = \theta(ax + b)$$

$$\theta(y) \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



- » Divide into regions

- $F_{[6,11]} = \{0 \rightarrow 1, 1 \rightarrow 2\}$

6.891 Machine Learning

## Using Bayes' Law

- Evaluate  $P(f | 1)$  and  $P(f | 2)$   
» Observing lots of data

- Use Bayes' Law:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

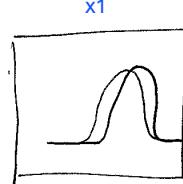
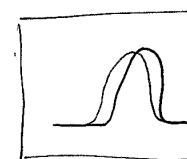
$$P("2" | F = f) = \frac{P(F = f | "2")P("2")}{P(F = f)}$$

$$P("1" | F = f) = \frac{P(F = f | "1")P("1")}{P(F = f)}$$

6.891 Machine Learning

## Combining Features

- Add features to separate



# *6.891 Machine Learning and Neural Networks*

## Lecture 2: The Probabilistic Approach

6.891 Machine Learning

## News

- For those of you that missed the first class...
  - » First problem set is on the web (Due: 9/16)
- The web page is getting updated regularly.
- We will hand out a grading guidelines when you are given the first problem set to grade.
  - » These grades will not assume perfect accuracy...

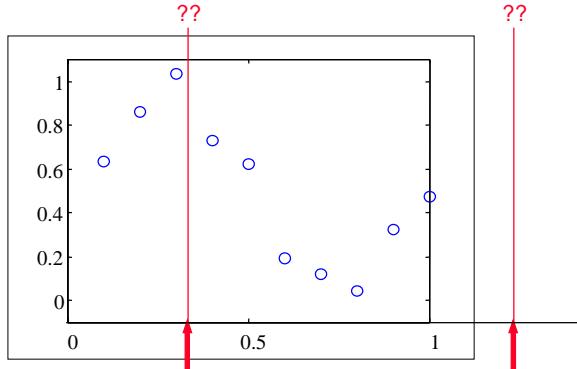
6.891 Machine Learning

## *Review & Overview*

- **Lecture 1:**
  - » Define Learning, Induction, Regression, Classification
  - » Show example applications: Digits, Sounds, Speech
  - » Brief Mention of Probability
- **Finish the introduction to Learning**
  - » Fitting functions to data...
  - » Overfitting
- **The Probabilistic Approach**
  - » Review some simple probability
  - » Apply it to classification tasks

6.891 Machine Learning

## Fitting a Curve to Data



$$\text{Data : } \{x^j, t^j\} \rightarrow y(x)$$

6.891 Machine Learning

You are given 10 example data points. These are samples of physical relationship, perhaps including noise.

Your challenge is to make prediction for this relationship

- Interpolation: between the example points
- Extrapolation: beyond the data.

In principle there are an infinite number of functions that could be associated with this data... our challenge is to pick one.

In the final analysis we may want to hedge our bets and return a probability distribution of functions.

## *Polynomial Fitting*

Data :  $\{x^j, t^j\}$

$$y(x^j) = w_0 + w_1(x^j)^1 + \dots + w_M(x^j)^M = \sum_l^M w_l(x^j)$$

$$y(x^j; \mathbf{w})$$

Weight vector :  $\mathbf{w} = \{w_0, w_1, \dots\}$

6.891 Machine Learning

Imagine that we are constraining ourselves to class of polynomials.

Each M-th order polynomial is parameterized by M+1 parameters.

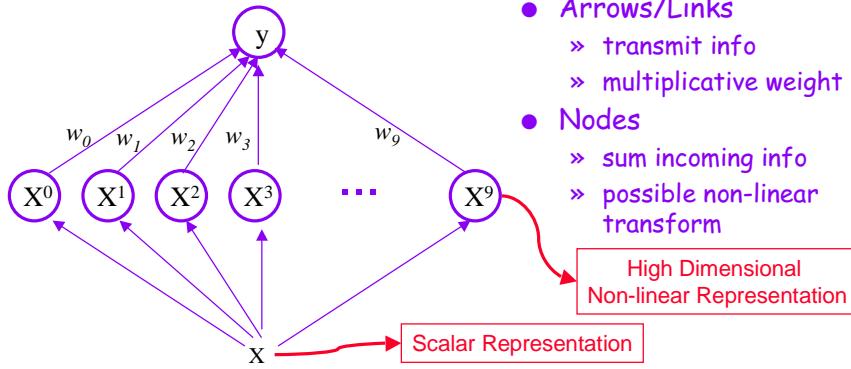
The learning process, becomes a process by which we select values of W\_i,

The dependency of the y(.) function on W will can be highlighted by the notation  $y(x; w)$ .

## Graphical Representation

$$y(x^j) = \sum_l^M w_l(x^j)$$

- Graph represents function
- Information Flows from bottom to top



6.891 Machine Learning

- Arrows/Links
  - » transmit info
  - » multiplicative weight
- Nodes
  - » sum incoming info
  - » possible non-linear transform

- While the algebraic notation for  $y()$  is clear and specific, we will see that sometimes it is also useful to develop a graphical notation of both classifiers and regression functions.
- This idea was originally popularized in the neural network literature, wherein neural networks were almost always drawn out in their graphical form.
- The graphical notation points out that an *intermediate* representation for  $X$  is formed ( $M+1$  exponentiations). The resulting problem is then one of learning the linear relationship between this high dimensional space and  $t$ .

## Choose the Best Polynomial

- Which polynomial function is best?
  - » Best predictions on training data...
  - » Best predictions on future data... interpolation/extrapolation
    - Best expected loss on future data.
    - Where do we get this data?

$$E = \frac{1}{2} \sum_j \text{loss}(y(x^j; \mathbf{w}) - t^j)$$

Empirical  
Loss

$$\hat{\mathbf{w}} = \min_w E$$

Find “Optimal”  
weights

6.891 Machine Learning

- What defines the best polynomial function? Perhaps it is the one which is most consistent with the training data?
- Actually we would rather return the function which makes the best predictions on future data - unfortunately there may be no source for this data.
- For the time being let's assume that we want to find the function which best agrees with training data... the function with the lowest loss.

## *Simple Loss Functions Simplify Learning*

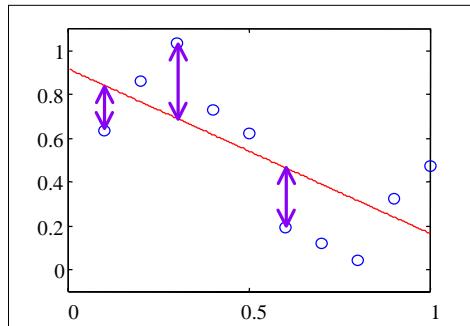
$$\text{loss}(\delta) = \delta^2 \quad \rightarrow \quad E = \frac{1}{2} \sum_j (y(x^j; \mathbf{w}) - t^j)^2$$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{1}{2} \sum_j 2(y(x^j; \mathbf{w}) - t^j)(x^j)_i = 0 \\ &= \sum_j (y(x^j; \mathbf{w}) - t^j)(x^j)_i = 0\end{aligned}$$

6.891 Machine Learning

- Certain simple loss functions lead to learning algorithms which are easy to derive and inexpensive to compute.
- For example, squared loss can be solved by differentiating and setting this to zero.
- The result is a set of linear equations that can be solved by inverting a matrix.

## *First order fit...*



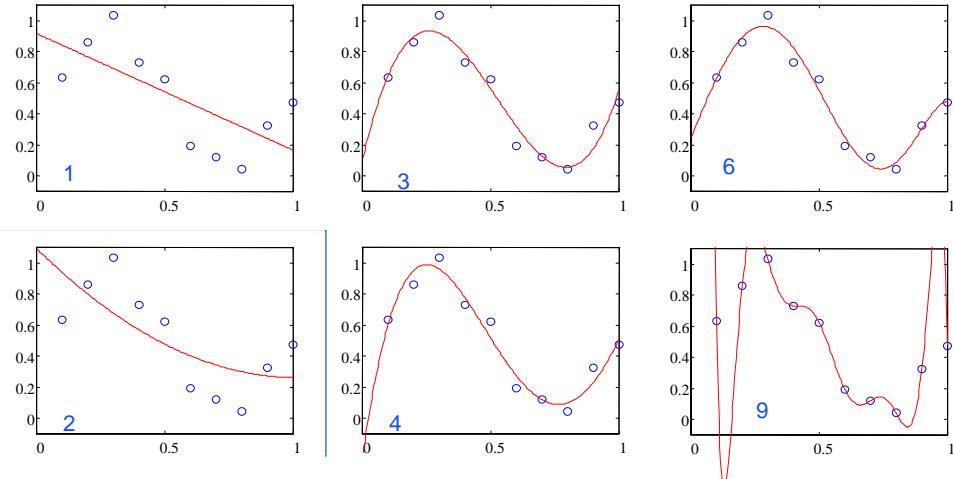
$$E = \frac{1}{2} \sum_j (y(x^j; \mathbf{w}) - t^j)^2$$

The optimal  
function minimizes  
the residual error

6.891 Machine Learning

- There is a pleasant physical analogy for the squared loss. The functions are connected by springs to the data. The system is then allowed to relax until the forces are balanced. The minimum energy solution is the one that is “closest” to the training data.

## Fitting Different Polynomials



6.891 Machine Learning

Each order of polynomial leads to a different fit.

Higher order polynomials come closer to the training data.

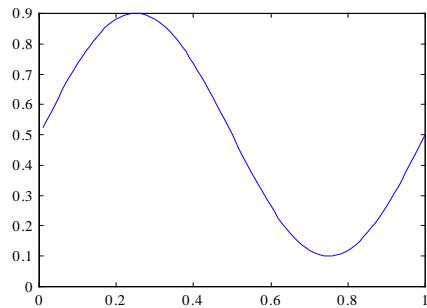
The 9th order polynomial can fit the 10 datapoints perfectly.

Which of these is the most likely to generalize

## *Target Function*

$$h(x) = 0.5 + 0.4 \sin(2\pi x)$$

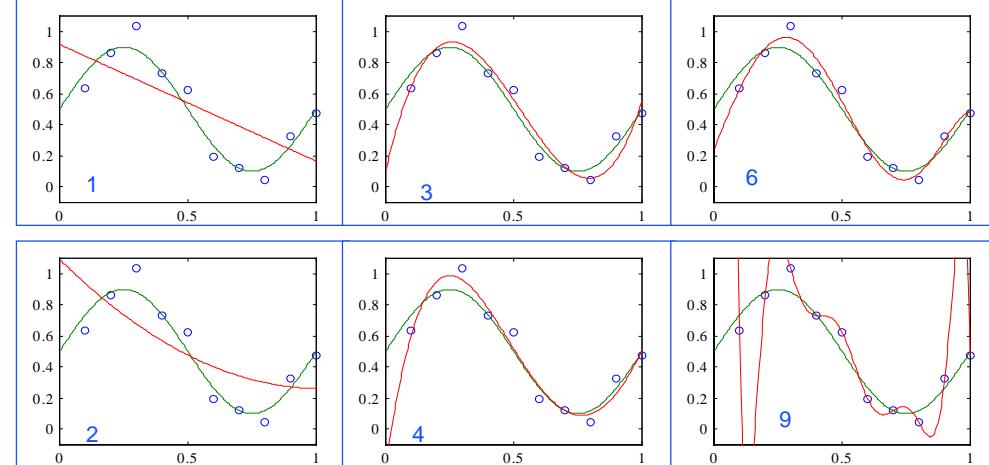
$$t^j = h(x^j)$$



6.891 Machine Learning

The function that generated the data was not a polynomial at all.

## Fitting Different Polynomials



6.891 Machine Learning

Probably the best approximation was 6th order (though 3rd is very good as well).

Ninth provides a terrible fit to the function, though it fits the training data perfectly. This is what is called overfitting...

## Matlab Code

```
% Construct training data
train_in = [1:10]/10;
train_out = 0.5 + 0.4 * sin(2 * pi * train_in) + 0.1 * randn(size(train_in));

% fit a polynomial
order = 3
p = polyfit(train_in, train_out, order)

% construct a test set
test_in = [1:300]/300;
true_out = 0.5 + 0.4 * sin(2 * pi * test_in);

% compute the polynomial prediction
fit_out = polyval(p, test_in);

% plot the results
% first: training data
% second: test data
% third: predictions
plot(train_in, train_out, 'o', test_in, true_out, test_in, fit_out)
axis([0 1 -0.1 1.1])
```

6.891 Machine Learning

Above is all the code used to generate the previous graphs.

As we can see Matlab allows us to explore issues in machine learning without much hacking.

## *First General Problem in Learning*

- **Control of complexity**

- » "Entities should not be multiplied without necessity."
  - W. Occam 12th Century
  - Occam's razor
- » "A physical theory should be as simple as possible, but no simpler." - A. Einstein
- » "Good theories are falsifiable." - V. Vapnik
- » Complex theories are likely to be wrong. - P. Viola

6.891 Machine Learning

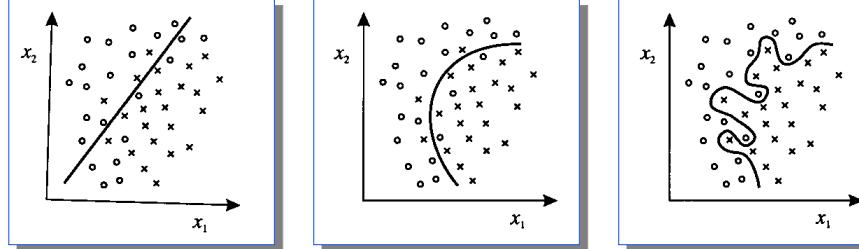
There are a few general (grand challenge) problems in machine learning. Perhaps the most important is the problem of controlling complexity. We have seen that a simpler approximator can fit an unknown function better than a more complex approximator. Building a theory for this is one grand challenge of learning.

Clearly this problem has been appreciated for a long time.

Vapnik's statement is perhaps the most confusing. When he says theory what he means is something like a learning algorithm. The learning algorithm which fits 9th order polynomials to 10 datapoints is not falsifiable.

- No set of datapoints would fail to be fit perfectly by this data.

## *Overfitting in Classification*



6.891 Machine Learning

This is not to say that such problems are unique to regression. Determining decision boundaries for classification is very similar.

We need to balance the complexity of the boundary against the accuracy on training data.

## Probabilistic Notation

$P(X = x)$  where  $X$  is a Random Variable  
 $P()$  is a Probability Distribution

Shorthand:

$P(X)$  the distribution function  
or  $P_X(.)$

$$P(x) = P(X = x) \quad P(y) = P(Y = y)$$

$$P_X(x) = P(X = x) \quad P_Y(y) = P(Y = y)$$

6.891 Machine Learning

Introduction of probabilistic notation.

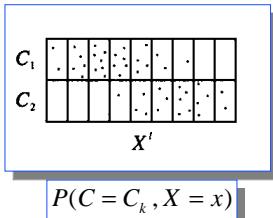
Note that there are several potentially confusing short hand notations.

## *Recall the probabilistic approach*

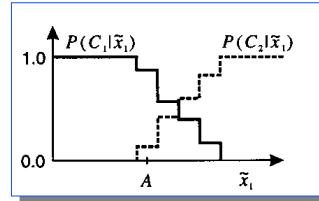
- Given a classification problem:
  - » Speech/Music, Bass/Salmon, Rotten/Ripe
- Choose a feature of your examples:
  - » Fish: width, height, color
  - » Fruit: color, weight
  - » Sounds: Spectrum Variance
- Record the distribution of Feature vs. Class
- Given an unclassified example
  - » Compute the  $P(F|C1)$  and  $P(F|C2)$
  - » Classify using Bayes Rule

6.891 Machine Learning

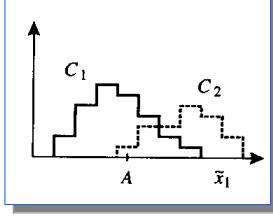
## Probabilistic Approach



$$P(C = C_k, X = x)$$



$$P(C = C_k | X = x) = \frac{P(X = x | C = C_k) P(C = C_k)}{P(X = x)}$$



$$P(X | C1) \text{ & } P(X | C2)$$

$$P(C_k | x) = \frac{P(x | C_k) P(C_k)}{P(x)}$$

Thomas Bayes  
1702-1761

6.891 Machine Learning

## Probability Densities

$$P(X \in [a, b]) = \int_a^b p(X = x) dx$$

$$p(X = x) = \frac{d}{db} P(X \in [a, b])$$

$$p(x) = p_x(x) = p(X = x)$$

$$P(C = C_k | X = x) = \frac{p(X = x | C = C_k) P(C = C_k)}{p(X = x)}$$

$$P(C_k | x) = \frac{p(x | C_k) P(C_k)}{p(x)}$$

6.891 Machine Learning

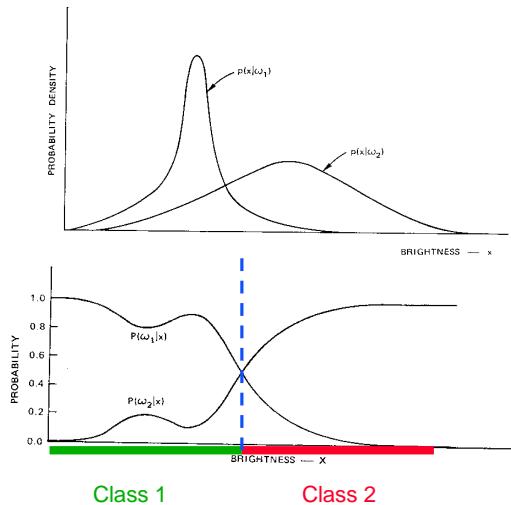
Probability Densities are necessary because for a continuous random variable the probability of every event is zero.

The density measures the slope of the cumulative distribution function.

Alternatively it is the probability per unit area (or length, or volume) measured over an infinitesimal area.

Somewhat surprisingly the density used in the same way that the distribution function is used. In other words the probability *distribution* of the class give the feature value can be found using the densities of the features.

## Bayes Law for Densities



$$C(x) = \begin{cases} \omega_1 & P(\omega_1 | x) > P(\omega_2 | x) \\ \omega_2 & \text{otherwise} \end{cases}$$

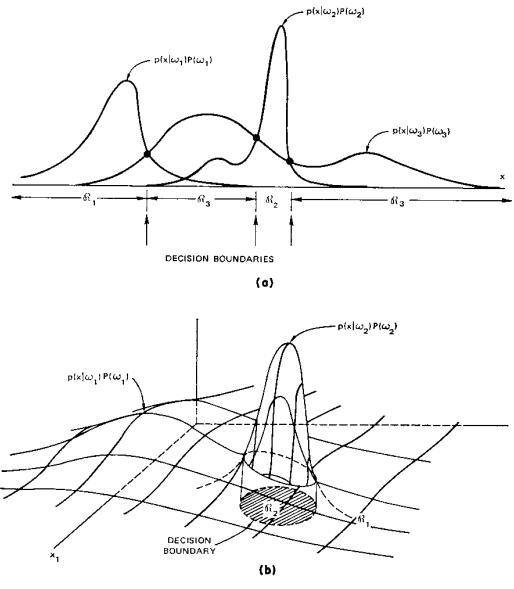
Duda & Hart, 1973

6.891 Machine Learning

Just as before we can graph the conditional probability of class given feature. The functions are now continuous...

Given the Bayes classification rule, a set of decision regions are defined.

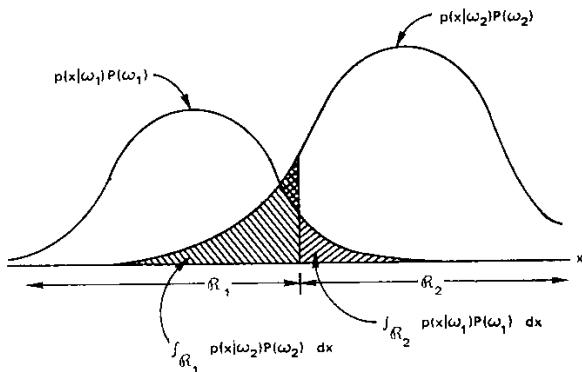
## Decisions



These analyses are easily generalized to:

- Multiple classes
- Multiple dimensions

## Analysis of Decision Rule



$$\begin{aligned} P(\text{error}) &= P(x \in R_2, C_1) + P(x \in R_1, C_2) \\ &= P(x \in R_2 | C_1)P(C_1) + P(x \in R_1 | C_2)P(C_2) \\ &= \int_{R_2} p(x | C_1)P(C_1)dx + \int_{R_1} p(x | C_2)P(C_2)dx \end{aligned}$$

6.891 Machine Learning

## *Minimize Expected Loss or Risk*

$$L_{kl} = \{ \text{Loss if } C(x^j) = C_l \text{ and } t^j = C_k \}$$

$$Risk_k = \sum_l L_{kl} \int_{R_l} p(x | C_k) dx$$

Risk for  
elements of  $C_k$

$$Risk = \sum_k R_k P(C_k)$$

Overall Risk

## *Probabilistic Classification Review*

- If we are given  $P(F|C)$  &  $P(C) \rightarrow P(F,C)$ 
  - » How the feature is distributed for each class
- We can use this information to classify new example using Bayes Rule.
  - » Minimizes the probability of error...
  - » We may instead wish to minimize risk
- Where is the machine learning?

6.891 Machine Learning

## *Information Retrieval*

- The Altavista Problem
- 125,000,000 documents on the web
  - » Take a long time to browse
- Simple Keyword Search
  - » Find documents with "German" and "car"
  - » Might miss "Germany" and "cars"
    - Stemming
  - » Misses "Mercedes" and "automobile"
- Machine Learning?
  - » Given 5, 10, 50 documents on German cars build a classifier.

6.891 Machine Learning

# *Keyword Search Works Well*

Dick Frantz' Home Page

**German Cars Index**

Subtopics:

Top

Porsche

BMW

Mercedes-Benz

Audi

Volkswagen

Opel

Photo Gallery

Sports Car Index

Home Page

---

**German Cars**

"Ve haf vays to make you enjoy dis sport." The word "technology" has a Greek root. It really should be German.

Come see the glorification of motoring in the ever increasing precision and dedication to purpose our German cousins have wrought.

BUT THEN.....

Technology is not the ONLY aspect of German automobiles alone. Italians have Style, and the French, Grace. "Purpose" would seem to define "German". So... what is the "purpose" of the New Beetle? Why, to DELIGHT, of course!



---

**Porsche**

The 911-GTR FIA-GT GT1 Racer



## Naïve Bayes Classifier

- Assume each word is an independent feature

$$f_i(Doc_j) \quad 1 \text{ if Doc } j \text{ has word } i.$$

$$P(F_i | C_j) \quad \text{Probability of word } i \text{ appearing in a Doc from Class } j$$

$$P(\{f_i\} | C_j) = \prod_i P(F_i = f_i | C_j)$$
$$P(C_j | \{f_i\}) = \frac{P(C_j) \prod_i P(F_i = f_i | C_j)}{\prod_i P(F_i = f_i)}$$

© 2023, Machine Learning

## *Estimating Probabilities*

- Maximum Likelihood

$$P(F_i) = \frac{\#\{\text{Docs containing word } i\}}{\#\{\text{Docs}\}}$$

$$P(F_i | C_j) = \frac{\#\{\text{Training Docs with word } i\}}{\#\{\text{Training Docs}\}}$$

Potential Bug:  
None of our Training Docs contain “Mercedes”

$$P(f_i | \chi) = p_i^{n_i} (1-p_i)^{N-n_i}$$

$$\frac{\partial}{\partial p_i} P_i^{n_i} (1-p_i)^{N-n_i} = p_i^{n_i-1} (1-p_i)^{N-n_i-1} (N-n_i)$$

$$n_i (1-p_i)^{N-n_i} p_i^{n_i-1} = (N-n_i) p_i^{n_i} (1-p_i)^{N-n_i-1}$$

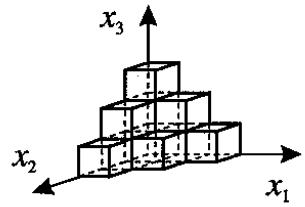
$$n_i (1-p_i) = (N-n_i) p_i$$

$$n_i - n_i p_i = N p_i - n_i p_i$$

$$N p_i = n_i \implies p_i = \frac{n_i}{N}$$

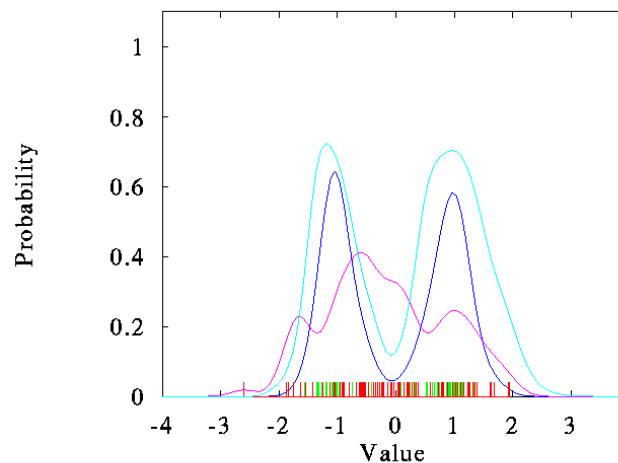
6.891 Machine Learning

## *Curse of Dimensionality*



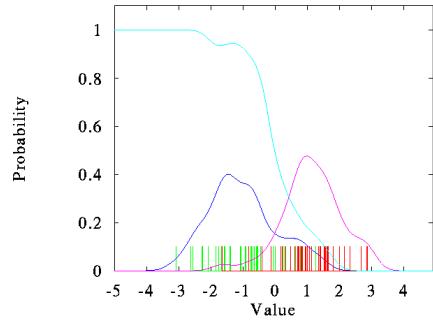
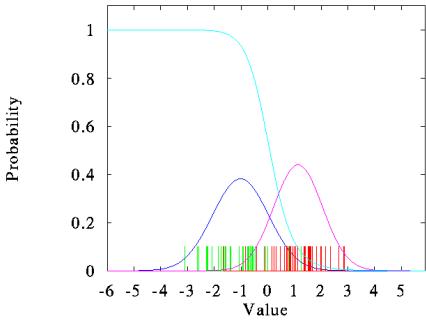
- It is not always better to measure more features
- New results seem to address this problem.
  - » Support Vectors, Boosting, etc.

## *Density Estimation is Ambiguous*



6.891 Machine Learning

## *Impacts Classification*



6.891 Machine Learning

# *6.891 Machine Learning and Neural Networks*

## *Lecture 3: Density Estimation*

Machine Learning

## *News*

- Sorry about the recitation mix up!!
  - » We will announce by email soon.
- Problem Set 1 is due tomorrow.
  - » See web for policy...
- Problem Set 2 will be available by tonight.
- Kinh and I will be taking photos

Machine Learning

## Review & Overview

- Lecture 2:

- » Overfitting: Polynomials
- » Reviewed the Probabilistic Approach
- » Information Retrieval Example

- Density/Distribution Estimation

- » Information Retrieval
  - estimating binary RV's
- » Gaussians
- » Multi-dimensional Gaussians
- » Non-parametric Densities

Machine Learning

## Keyword Search Works Well

Dick Frantz' Home Page

**German Cars Index**

Subtopics:

- [Blank box]

[Top](#)

[Porsche](#)

[B&W](#)

[Mercedes-Benz](#)

[Audi](#)

[Volkswagen](#)

[Opel](#)

[Photo Gallery](#)

[Sports Car Index](#)

[Home Page](#)

### German Cars

"We haf ways to make you enjoy dis sport." The word "technology" has a Greek root. It really should be German.

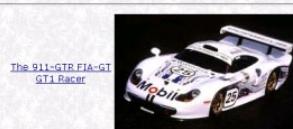
Come see the glorification of motoring in the ever increasing precision and dedication to purpose our German cousins have wrought.

BUT THEN.....

Technology is not the ONLY aspect of German automobiledom. Italians have Style, and the French, Grace. "Purpose" would seem to define "German". So... what is the "purpose" of the New Beetle? Why, to DELIGHT, of course!



### Porsche



## Bayesian Text Classification

$\{d_k\}$  : A collection of documents

$$W_i(d_k) : \begin{cases} 1 & \text{if } d_k \text{ contains word i} \\ 0 & \text{otherwise} \end{cases}$$

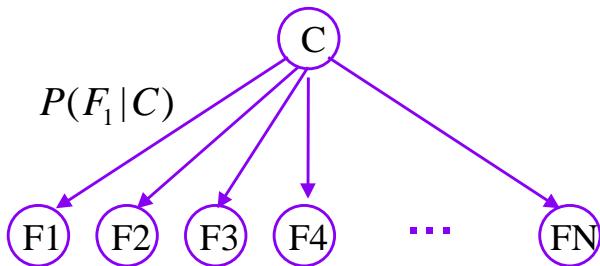
$$P(F_i = 1 | C = c_j) = p_{ij} \quad \text{Probability of word i appearing in a Doc from Class j}$$

$$\begin{aligned} P(F_1 = f_1, F_2 = f_2, \dots | C = c_j) \\ = P(\{f_1 - f_N\} | C = c_j) \\ \equiv \prod_i P(F_i = f_i | C = c_j) \end{aligned}$$

Assume  
Independence

## Bayes Nets Show Dependencies

$$P(\{f_i\} | C_j) = \prod_i P(F_i = f_i | C_j)$$



Bayes Nets show the dependencies between RV's

Machine Learning

## *Classification Using Bayes Law*

$$P(c_j | \{f_i\}) = \frac{P(c_j) \prod_i P(f_i | c_j)}{\prod_i P(f_i)}$$

$c_1$  = German Cars

$c_0$  = Other Documents

Machine Learning

## *Estimating Probability Distributions*

$\{d_k\}$  : A collection of documents

$$W_i(d_k) = f_{ki} : \begin{cases} 1 & \text{if } d_k \text{ contains word i} \\ 0 & \text{otherwise} \end{cases}$$

$$P(F_i = 1 | C = c_j) = p_{ij}$$

- How can we learn  $p_{ij}$ ?
  - » Maximum Likelihood Principle
  - » Choose  $p_{ij}$  so that the training data is most probable.

Machine Learning

## Maximum Likelihood

$$\begin{aligned} P(\{d_k\} | c_0) &= \prod_k P(d_k | c_0) = \prod_k P(\{f_{k1} - f_{kN}\} | c_0) \\ &= \prod_k \prod_i P(f_{ki} | c_0) \\ &= \prod_k \prod_i (p_{ij})^{f_{ki}} (1-p_{ij})^{(1-f_{ki})} \\ &= (p_{ij})^{n_i} (1-p_{ij})^{(N-n_i)} \end{aligned}$$

Machine Learning

## Log Likelihood

$$\begin{aligned} L &= \log \left( (p_{ij})^{n_i} (1-p_{ij})^{(N-n_i)} \right) \\ &= n_i \log(p_{ij}) + (N-n_i) \log(1-p_{ij}) \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial p_{ij}} &= n_i \frac{\partial \log(p_{ij})}{\partial p_{ij}} + (N-n_i) \frac{\partial \log(1-p_{ij})}{\partial p_{ij}} \\ &= n_i \frac{1}{p_{ij}} + (N-n_i) \frac{-1}{1-p_{ij}} \\ &= 0 \end{aligned}$$

Machine Learning

## Maximum Likelihood

$$n_i \frac{1}{p_{ij}} + (N - n_i) \frac{-1}{1 - p_{ij}} = 0$$

$$\frac{n_i}{p_{ij}} = \frac{(N - n_i)}{1 - p_{ij}}$$

$$\frac{n_i}{(N - n_i)} = \frac{p_{ij}}{1 - p_{ij}}$$

$$\frac{n_i/N}{\left(1 - n_i/N\right)} = \frac{p_{ij}}{1 - p_{ij}}$$



$$p_{ij} = \frac{n_i}{N}$$

## Estimating Probabilities

- Maximum Likelihood

$$P(F_i = 1) = \frac{\#\{\text{Docs containing word } i\}}{\#\{\text{Docs}\}}$$

$$P(F_i = 1 | C_j) = \frac{\#\{\text{Training Docs with word } i\}}{\#\{\text{Training Docs}\}}$$

Potential Bug:  
None of our Training Docs contain “Mercedes”

## Prior Expectations

- Given a small amount of data we can't be absolutely sure that "Mercedes" will never appear in documents from our class...
  - » We may have gotten unlucky
- Use prior expectations to improve our estimates
- Problem:
  - » Mercedes occurs in 10 out of 1000 total documents
  - » But never in the "German cars" training set
  - » What is a good estimate for  $p(\text{mercedes} / \text{GermanCars})$ ?

Machine Learning

## Bayesian Parameter Estimation

- Bayes to the rescue again!

Maximum Likelihood

$$\max_{p_{ij}} P(\{d_k\} | c_0, p_{ij})$$

Maximum A Posteriori

$$P(p_{ij} | \{d_k\}, c_0) = \frac{P(\{d_k\} | c_0, p_{ij}) p(p_{ij})}{P(\{d_k\} | c_0)}$$

This turns out to be more useful  
for continuous parameters

Machine Learning

## What is the right prior?

- The most agnostic prior is the uniform density.

$$\begin{aligned}\max P(p_{ij} | \{d_k\}, c_0) &= \max P(\{d_k\} | c_0, p_{ij}) p(p_{ij}) \\ &= \max P(\{d_k\} | c_0, p_{ij})\end{aligned}$$

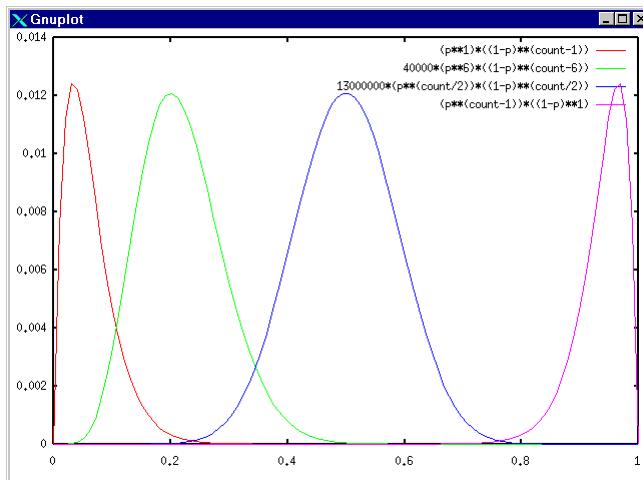
Maximum Likelihood

$$P(\{d_k\} | c_0, p_{ij}) = (p_{ij})^{n_i} (1 - p_{ij})^{(N - n_i)}$$

Machine Learning

## Probability of the parameters

$$P(\{d_k\} | c_0, p_{ij}) = (p_{ij})^{C_i} (1 - p_{ij})^{(N - C_i)}$$



## Bayesian Estimation

$$P(p_{ij} | \{d_k\} c_0) = \frac{P(\{d_k\} | c_0, p_{ij}) p(p_{ij})}{P(\{d_k\} | c_0)}$$

$$P(F_i = 1 | C = c_j, p_{ij}) = p_{ij}$$

$$\begin{aligned} P(F_i | c_j) &= \int P(F_i | c_j, p_{ij}) p(p_{ij} | \{d_k\} c_j) dp_{ij} \\ &= \int p_{ij} \frac{P(\{d_k\} | c_0, p_{ij}) p(p_{ij})}{P(\{d_k\} | c_0)} \\ &= \frac{\int p_{ij} P(\{d_k\} | c_0, p_{ij}) p(p_{ij})}{P(\{d_k\} | c_0)} \end{aligned}$$

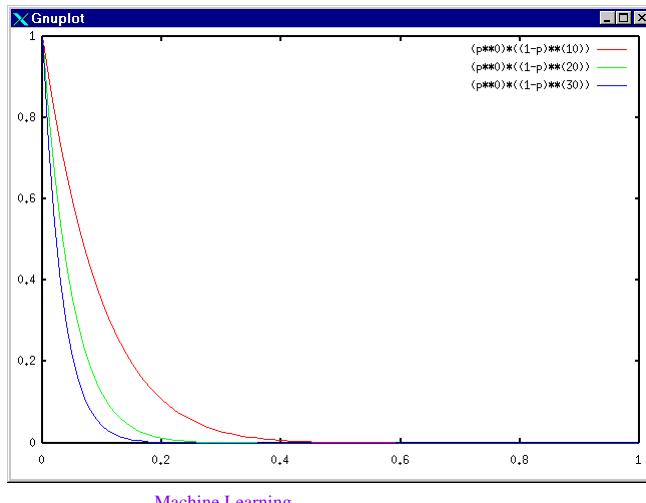
Machine Learning

*... Continued*

$$\begin{aligned} P(F_i | c_j) &= \frac{\int p_{ij} P(\{d_k\} | c_0, p_{ij}) p(p_{ij}) dp_{ij}}{\int P(\{d_k\} | c_0, p_{ij}) p(p_{ij}) dp_{ij}} \\ &= \frac{\int p_{ij} \left[ (p_{ij})^{n_i} (1-p_{ij})^{N-n_i} \right] \epsilon dp_{ij}}{\int \left[ (p_{ij})^{n_i} (1-p_{ij})^{N-n_i} \right] \epsilon dp_{ij}} \end{aligned}$$

Machine Learning

## *What if no Mercedes?*



Machine Learning

# *6.891 Machine Learning and Neural Networks*

## *Lecture 4: New Density Estimators*

Machine Learning

## *News*

- Problem set 1 will be handed out today.
- Problem set 2 is on the web.
  - » It is much harder than the first pset.
- Problem Sets:
  - » Please show some work.
  - » Make sure to get the pssets to Kinh
    - Especially if they are last minute

Machine Learning

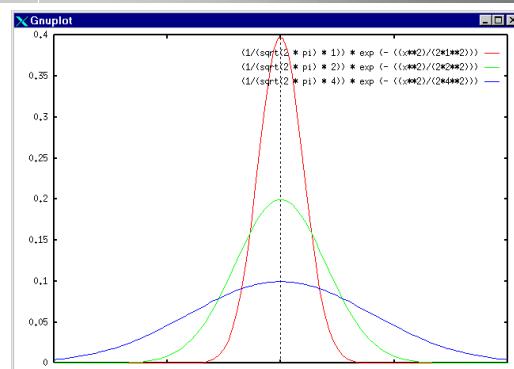
## Review & Overview

- Lecture 3:
  - » Talked about Information Retrieval
    - Need priors over parameters
  - » Derived Maximum Likelihood for Bernoulli RV's
  - » Discussed use of priors over parameters
- New Density Estimators (Continuous)
  - » Gaussian
  - » Non-parametric
  - » Mixture of Gaussians
- Quick tour of Expectation Maximization

Machine Learning

## Why Gaussians ?

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Analytically Tractable
- Central Limit Theorem
  - » Sum of many variables is Gaussian
- Linear Transforms of Gaussian are Gaussian
- Gaussians have the highest Entropy

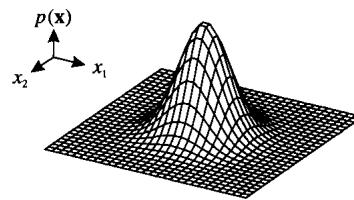
Machine Learning

## Multi-Dimensional Gaussian

$$P(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1} (x-\mu)}$$

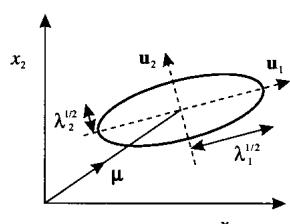
$$\mu = E[x]$$

$$\Sigma = E[(x-\mu)(x-\mu)^\top]$$



Machine Learning

## Eigen Structure



$$\begin{aligned} \Sigma &= E \Lambda E^\top \\ &= \begin{pmatrix} | & | & \dots \\ e_1 & e_2 & \end{pmatrix} \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} \begin{pmatrix} -e_1 & - \\ -e_2 & \vdots \end{pmatrix} \end{aligned}$$

$$E^\top E = I \quad \Sigma^{-1} = E \Lambda^{-1} E^\top$$

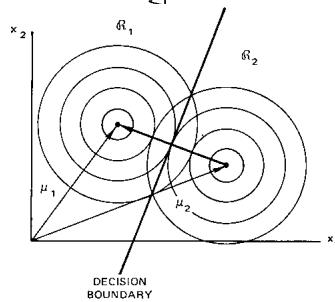
Machine Learning

## Recall: Bayes Decision Boundaries

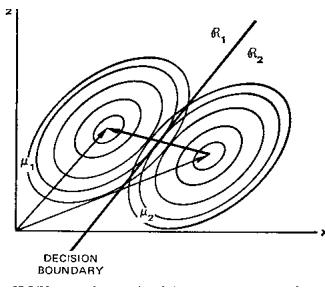
$$P(C_1|x) \stackrel{?}{>} P(C_2|x)$$

$$P(x|C_1)P(C_1) \stackrel{?}{>} P(x|C_2)P(C_2)$$

$$\log P(x|C_1) + \log P(C_1) \stackrel{?}{>} \log P(x|C_2) + \log P(C_2)$$



Machine Learning



## Discriminant Function

$$P(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} C^{-1/2} (x-\mu)^T \Sigma^{-1} (x-\mu)$$

$$\begin{aligned} \log P(x|C_k) &= \log \frac{1}{(2\pi)^{d/2}} + \log \frac{1}{|\Sigma_k|^{1/2}} - \frac{1}{2} (x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \end{aligned}$$

$$\begin{aligned} \gamma_k(x) &= -\frac{1}{2} \left[ \log |\Sigma_k| + (x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \right] \\ &\quad + \log P(C_k) \end{aligned}$$

Machine Learning

## Set Discriminants Equal

- Simplified Case  $\Sigma_1 = \Sigma_2 = \hat{\Sigma}$

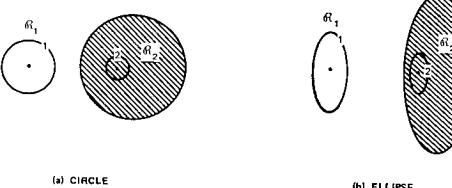
$$\gamma_1(x) = \gamma_2(x)$$

$$\log P(c_1) + (x - \mu_1)^T \hat{\Sigma}^{-1} (x - \mu_1) = (x - \mu_2)^T \hat{\Sigma}^{-1} (x - \mu_2) + \log P(c_2)$$

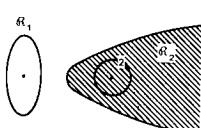
$$x^T \hat{\Sigma}^{-1} x - x^T \hat{\Sigma} \mu_1 - \mu_1^T \hat{\Sigma} x + \mu_1^T \hat{\Sigma} \mu_1$$

$$O = (\mu_1 - \mu_2)^T \hat{\Sigma}^{-1} x + \mu_1^T \hat{\Sigma} \mu_1 - \mu_2^T \hat{\Sigma} \mu_2 + \log P(c_1) - \log P(c_2)$$

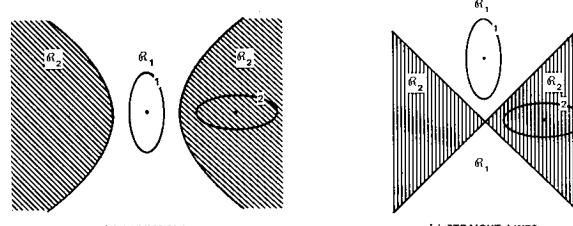
$$O = \tilde{W}^T X + b$$



(a) CIRCLE (b) ELLIPSE



(c) PARABOLA



(d) HYPERBOLA (e) STRAIGHT LINES

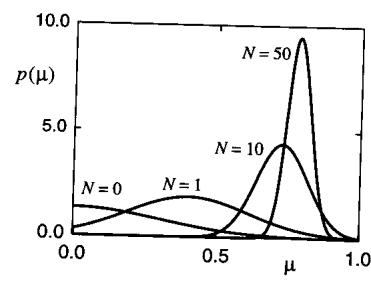
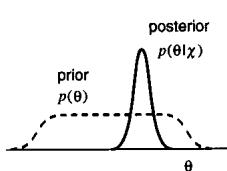
## Bayesian Parameter Estimation

- What if you little data...
- Or if you have strong expectations

$$\begin{aligned} p(x|\chi) &= \int p(x, \theta|\chi) d\theta \\ &= \int p(x|\theta, \chi) p(\theta|\chi) d\theta \\ &= \int p(x|\theta) p(\theta|\chi) d\theta \end{aligned}$$

Machine Learning

## Convergence of Probability

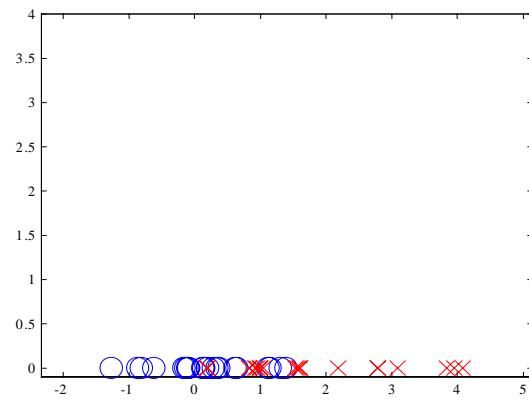


Machine Learning

## *Reminder: Why we are here*

-1.2660  
-0.8724  
-0.8081  
-0.6223  
-0.1624  
-0.1342  
-0.1098  
-0.0882  
0.1258  
0.1395  
0.1914  
0.2873  
0.3409  
0.3694  
0.6093  
0.6463  
1.1217  
1.1463  
1.3021  
1.3971

0.1781  
0.2013  
0.8293  
0.8299  
0.9217  
0.9434  
0.9851  
1.0079  
1.0539  
1.5355  
1.5621  
1.5875  
1.6015  
2.1811  
2.7845  
2.7879  
3.0956  
3.8428  
3.9562  
4.0800

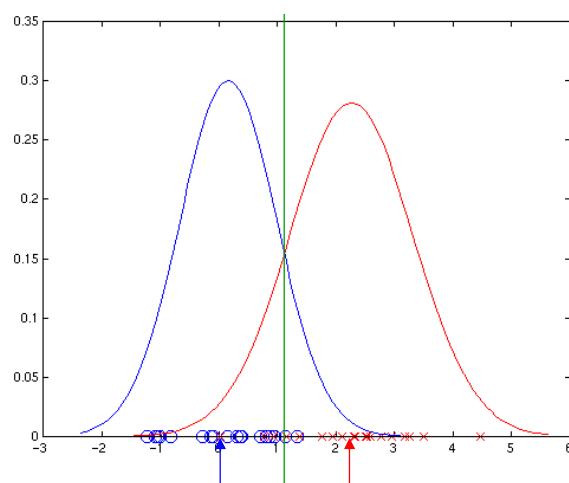


Machine Learning

## *Max Likelihood Gaussian*

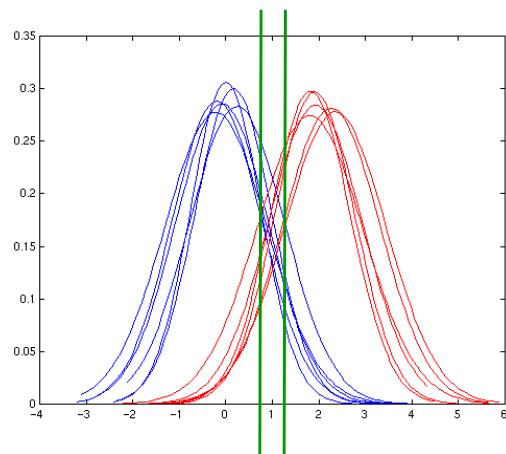
Mean: 0.16  
StDev: 0.8

Mean: 2.2  
StDev: 1.0



Machine Learning

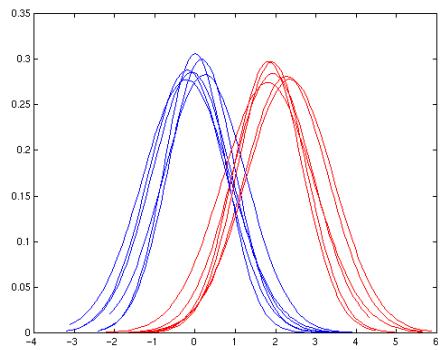
## *Different Samples, Different Decisions*



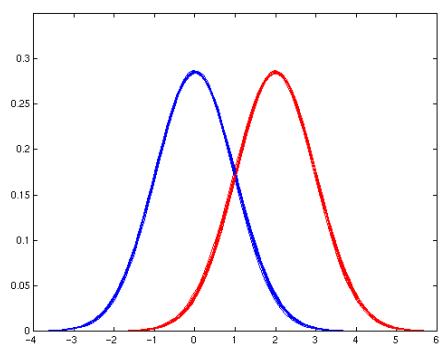
**Concept: Variance**  
The Variation you observe  
when training on different  
independent training sets

Machine Learning

## *Variance depends on data set size...*



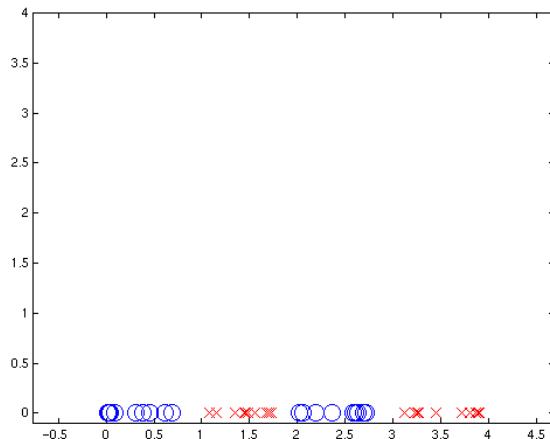
20 points



2000 points

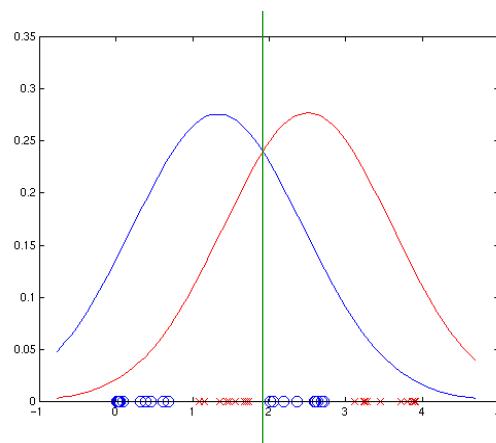
Machine Learning

*But when data gets more complex...*



Machine Learning

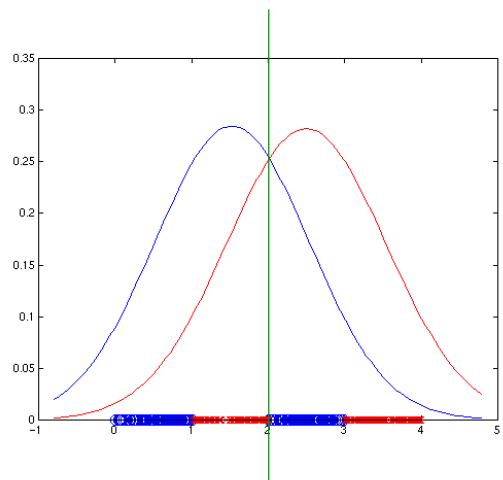
*... Gaussian don't work well*



Concept: Training Error  
Error in your classifier  
on the training set

Machine Learning

*Even if you had “infinite” data ...*

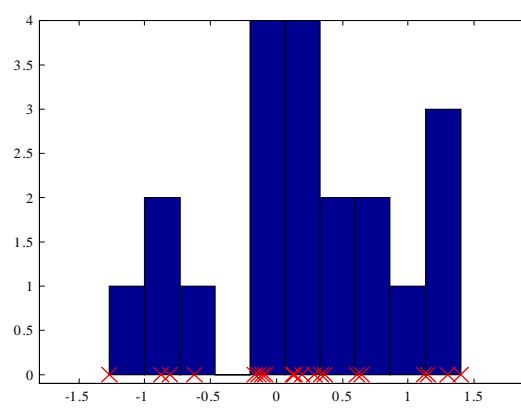


Related Concept: Bias  
Error in your classifier  
in the limit as size of  
training data grows.

Machine Learning

*Histogram*

-1.2660  
-0.8724  
-0.8081  
-0.6223  
-0.1624  
-0.1342  
-0.1098  
-0.0882  
0.1258  
0.1395  
0.1914  
0.2873  
0.3409  
0.3694  
0.6093  
0.6463  
1.1217  
1.1463  
1.3021  
1.3971



Divide by N to  
yield a probability

Machine Learning

# *6.891 Machine Learning and Neural Networks*

## *Lecture 5: Density Estimation and Classification*

© Paul Viola 1999

Machine Learning

1

## *News*

- **No Lecture on Wednesday**
  - » Be sure to get Kinh your graded psets by Wednesday
    - Recitation
    - Drop it off
- **Guest Lecture by Leslie Kaelbling on Friday**
  - » Reinforcement Learning

© Paul Viola 1999

Machine Learning

2

## Review & Overview

- Lecture 4:
  - » Gaussian Density Estimation
  - » Covariance
  - » Linear and Quadratic Discriminants
- New Density Estimators
  - » Non-parametric
  - » Mixture of Gaussians
  - » Quick tour of Expectation Maximization
- Application: Face Detection
  - » Mixture of Gaussians

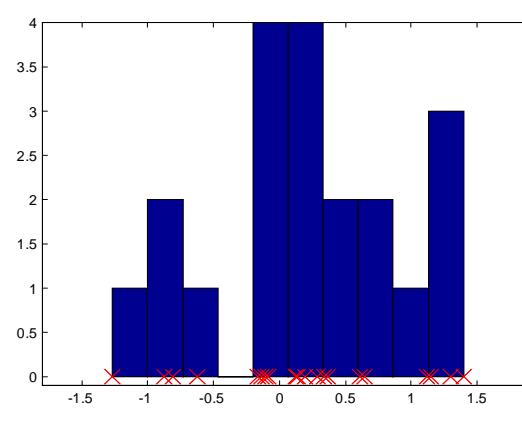
© Paul Viola 1999

Machine Learning

3

## Histogram

-1.2660  
-0.8724  
-0.8081  
-0.6223  
-0.1624  
-0.1342  
-0.1098  
-0.0882  
0.1258  
0.1395  
0.1914  
0.2873  
0.3409  
0.3694  
0.6093  
0.6463  
1.1217  
1.1463  
1.3021  
1.3971



Divide by N to  
yield a probability

© Paul Viola 1999

Machine Learning

4

## *Simple Algorithm*

```
function counts = myhist(data, centers)

% Initialize counts
counts = zeros(size(centers));

numdata = size(data,1);

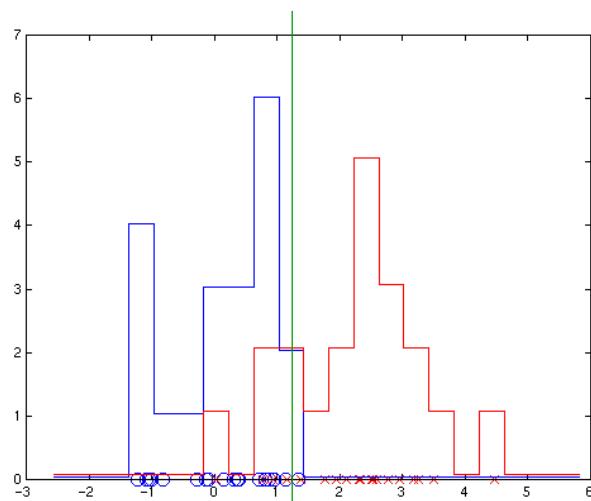
% For each datapoint compute distance to every center
for i = 1:numdata
    diffs = data(i,1) - centers;
    dists = diffs.^2;
    [minval mindist] = min(dists);
    counts(mindist) = counts(mindist) + 1;
end
```

© Paul Viola 1999

Machine Learning

5

## *Histogram*



© Paul Viola 1999

Machine Learning

6

## *Max Likelihood Gaussian*

Mean: 0.16  
StDev: 0.8

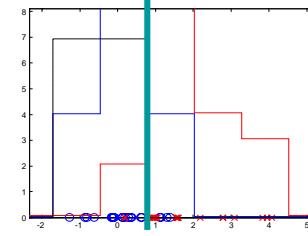
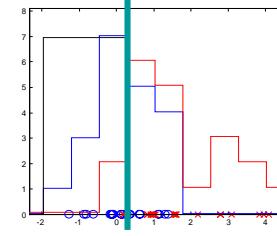
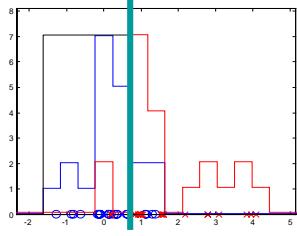
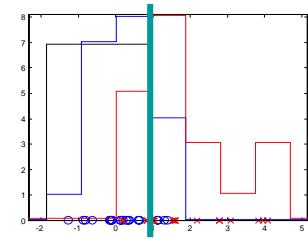
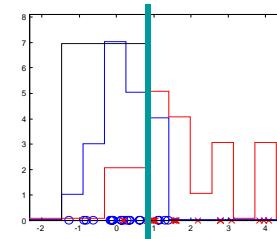
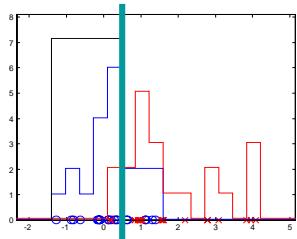
Mean: 2.2  
StDev: 1.0

© Paul Viola 1999

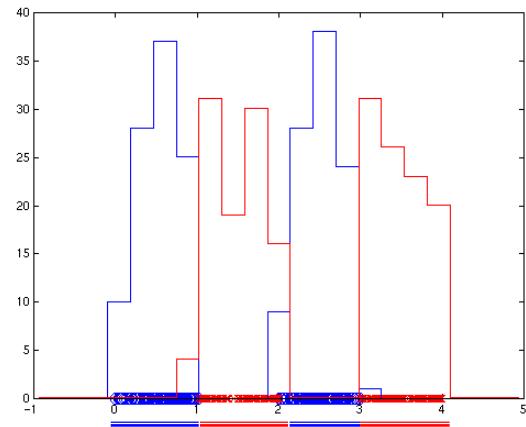
Machine Learning

7

## *Histogram Flexibility is Adjustable*



*Histograms have lower bias ...*

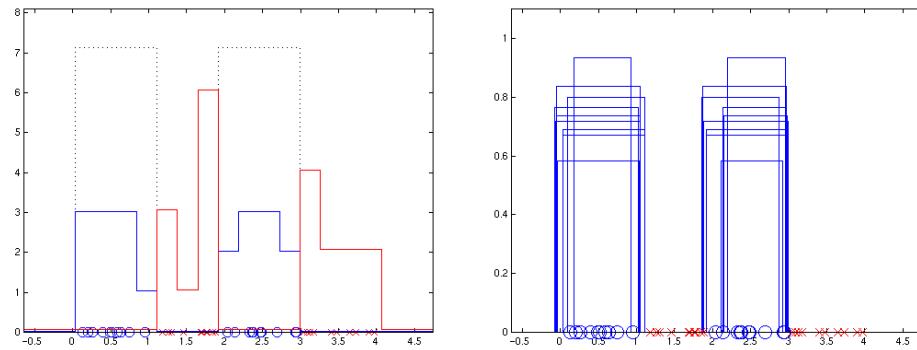


© Paul Viola 1999

Machine Learning

9

*... but higher variance.*

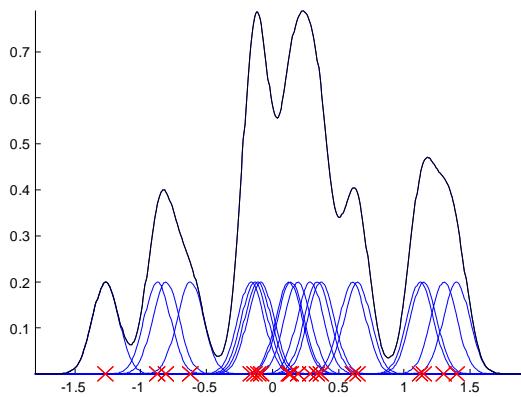


© Paul Viola 1999

Machine Learning

10

## Parzen: One Bump per Data Point



© Paul Viola 1999

Machine Learning

11

## Parzen Algorithm

```
function [func, range] = parzen(data, sigma)
range = splitrange(min(data), max(data), 500);
numdata = size(data, 1);

% For each point on range compute distance to every
% datapoint.
for i = 1:size(range, 2)
    gaussvals = gauss(range(i) - data, 0, sigma);
    func(i) = sum(gaussvals)/numdata;
end

plot(range, func)
```

© Paul Viola 1999

Machine Learning

12

## *Parzen and Histogram are Similar*

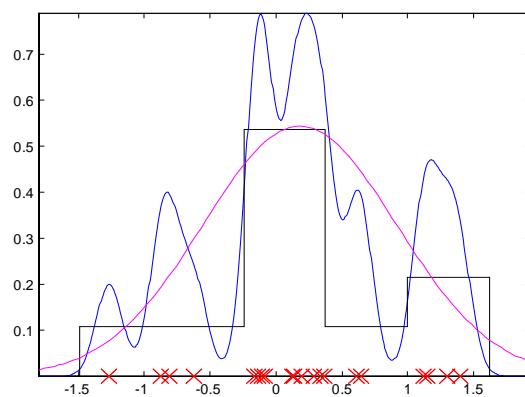
- Both can model any type of distribution
  - » Given plenty of data
- Both are simple
- Parzen is differentiable - Histogram is not
- Parzen is smooth - Histogram is not
- Histogram density Evaluation  $p(x)$  is cheap
- Parzen density Evaluation is linear in data size

© Paul Viola 1999

Machine Learning

13

## *All Three at Once*



© Paul Viola 1999

Machine Learning

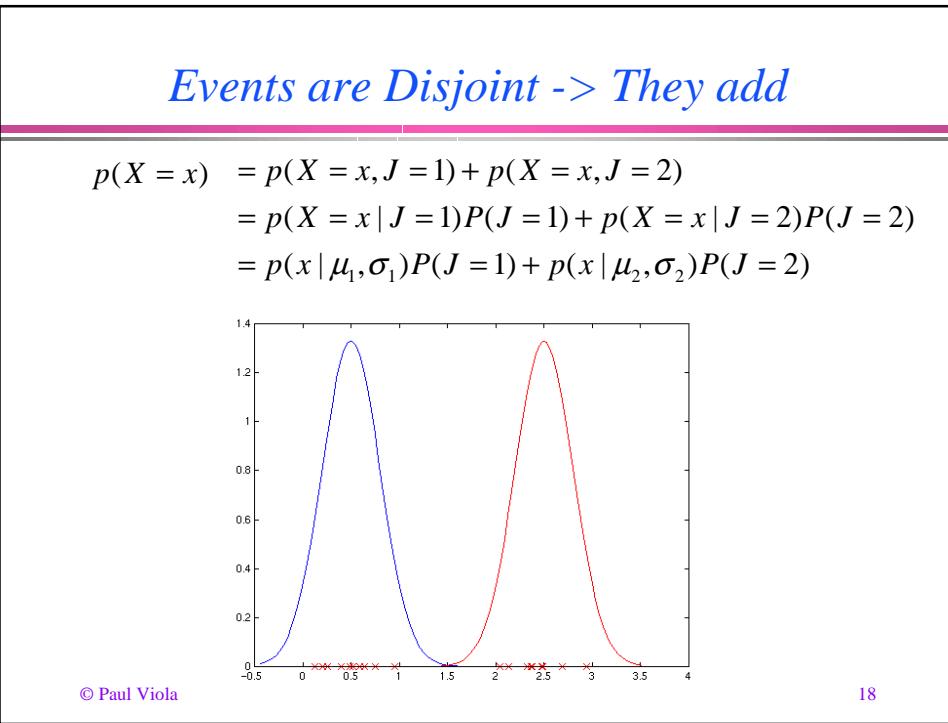
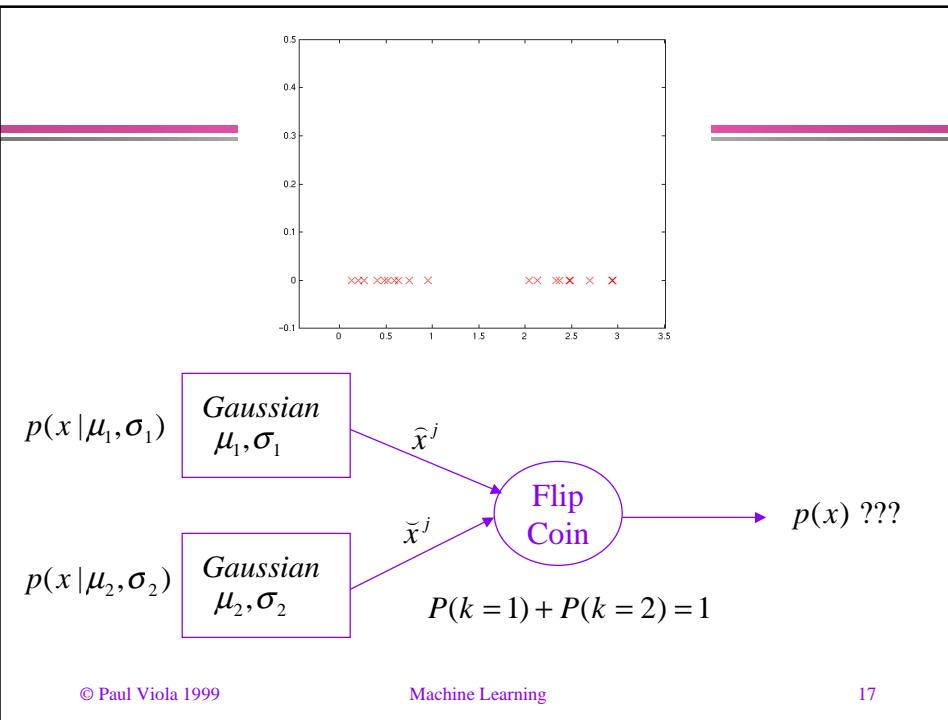
14

## *Properties of Non-parametric Techniques*

- Density is analytical function of data
- Bias and variance of density estimator can be adjusted to the problem
- Many more parameters must be estimated.
  - » Histogram:  $N^D$  bins
- Lose many of the simple properties of Gaussians

## *Semi-Parametric Models*

- Have more flexibility than parametric models
  - » like Gaussians
- Have less variance than non-parametric models
- Evaluation of  $p(x)$  is cheap
- Determination of parameters is expensive



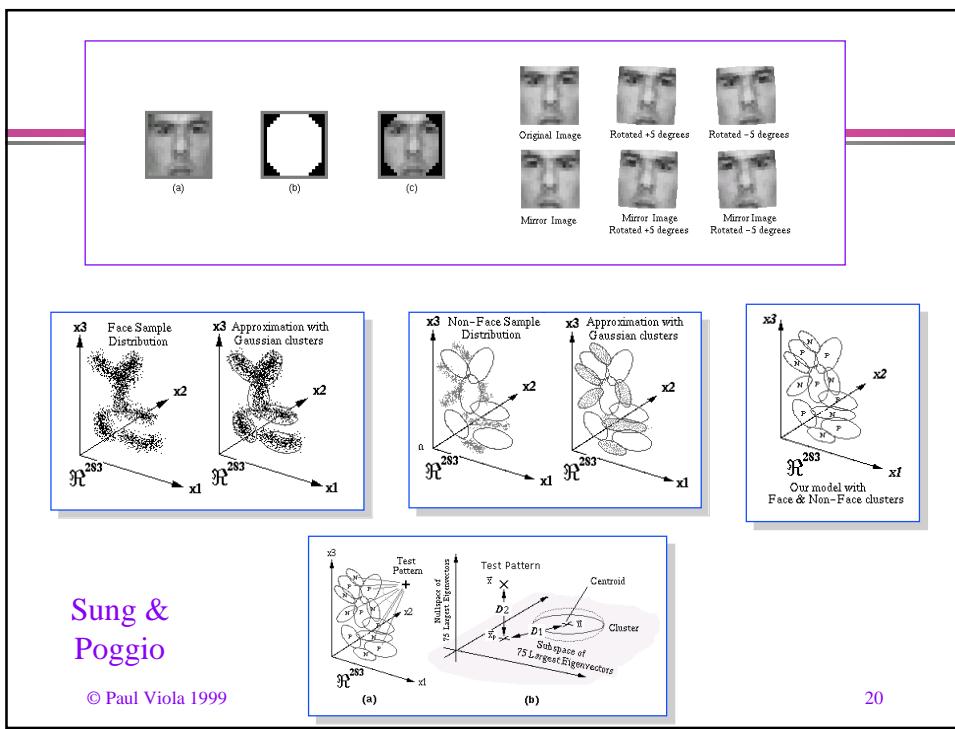
## Face Detection



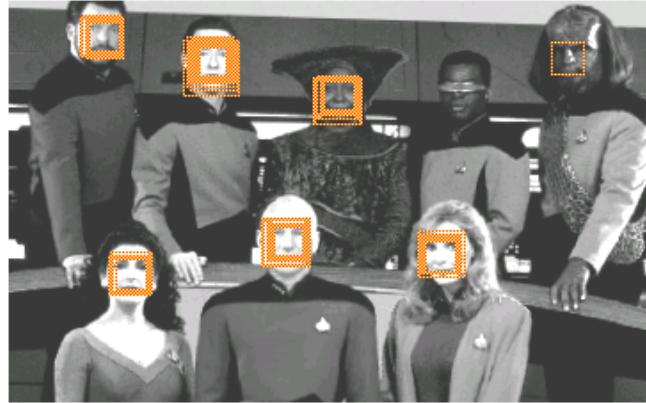
© Paul Viola 1999

Machine Learning

19



## *Results*



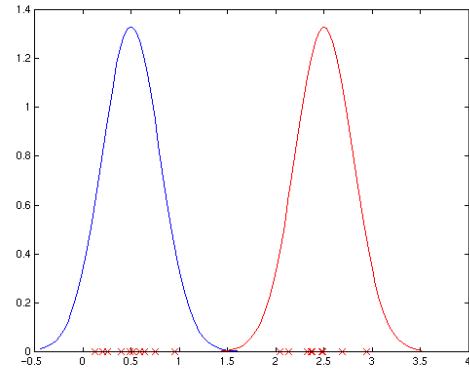
But, it takes minutes per image...

## *Face Detection*

- Great application of probabilistic classification
  - » Works very well
  - » Requires many thousands of parameters
  - » Computation time is very long
  
- Is there an Alternative? -> Discriminants
  - » Also works well
  - » Require fewer parameters
  - » Computation time is very short

## *Events are Disjoint -> They add*

$$\begin{aligned} p(X = x) &= p(X = x, J = 1) + p(X = x, J = 2) \\ &= p(X = x | J = 1)P(J = 1) + p(X = x | J = 2)P(J = 2) \\ &= p(x | \mu_1, \sigma_1)P(J = 1) + p(x | \mu_2, \sigma_2)P(J = 2) \end{aligned}$$



© Paul Viola

23

## *Expectation Maximization*

$$\mu_k = \frac{\sum_j P(k | x^j) x^j}{\sum_j P(k | x^j)}$$

$$\sigma_k^2 = \frac{\sum_j P(k | x^j) (x^j - \mu_k)^2}{\sum_j P(k | x^j)}$$

$$P(k) = \sum_j P(k | x^j)$$

$$E = -\log l(\{\mu_k, \sigma_k, q_k\})$$

Bounded Below?

Decreases?

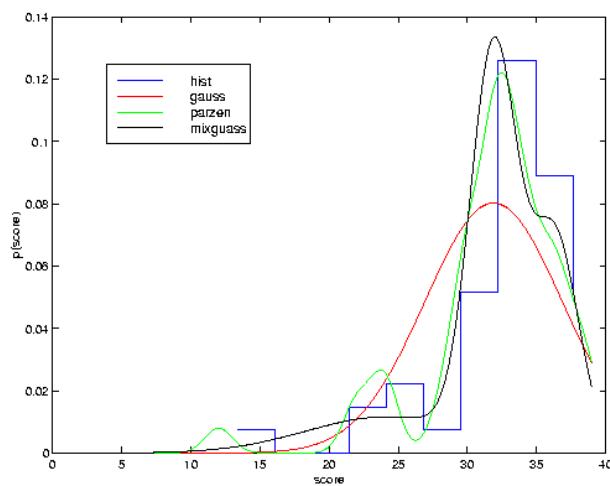
## News

- Sorry about missing last week...
  - » Scheduling hiccup which push Perceptrons out of Pset 2
- Pset 3 will be out by tonight
  - » Please get started early
- Pset 2 is due tomorrow
- Cross-grading worked out well!
  - » But, we noticed that a few people were not grading carefully.
  - » I would like you to take this task very seriously.

© Paul Viola 1999

Machine Learning

## Distribution of Grades: Pset 1



© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 5:
  - » Non-parametric Density Estimation
    - Histograms and Parzen Densities
  - » Semi-parametric, Mixture of Gaussians
  - » Application: Face Detection (... very complex)
  
- Perceptrons
- Training Perceptrons
- Generalized Perceptrons
- Multi-Layer Perceptrons

© Paul Viola 1999

Machine Learning

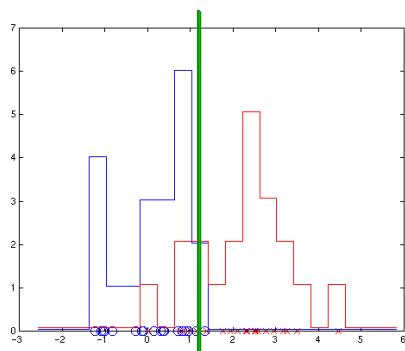
## *Where are we?*

- Introduced density estimation
  - » Discrete data
  - » Continuous data
  - » Parametric, Non-parametric, and Semi-parametric
- Used Bayes' law to classify new examples
  - » Minimizing either error or Risk.
  
- But, this is not the only way...
- In fact this approach has come under sustained attack recently.

© Paul Viola 1999

Machine Learning

## Between density and classification



- Often the details of the density do not matter

© Paul Viola 1999

Machine Learning

## Gaussians vs. Discriminants

- Example: Two classes, Gaussian classes, equal covariance.
  - The density estimator of  $N^2$  parameters
  - The resulting linear discriminant has  $N$  parameters
  - Why estimate the extra parameters???

$$y(x) = \mathbf{w}^T x + w_o$$

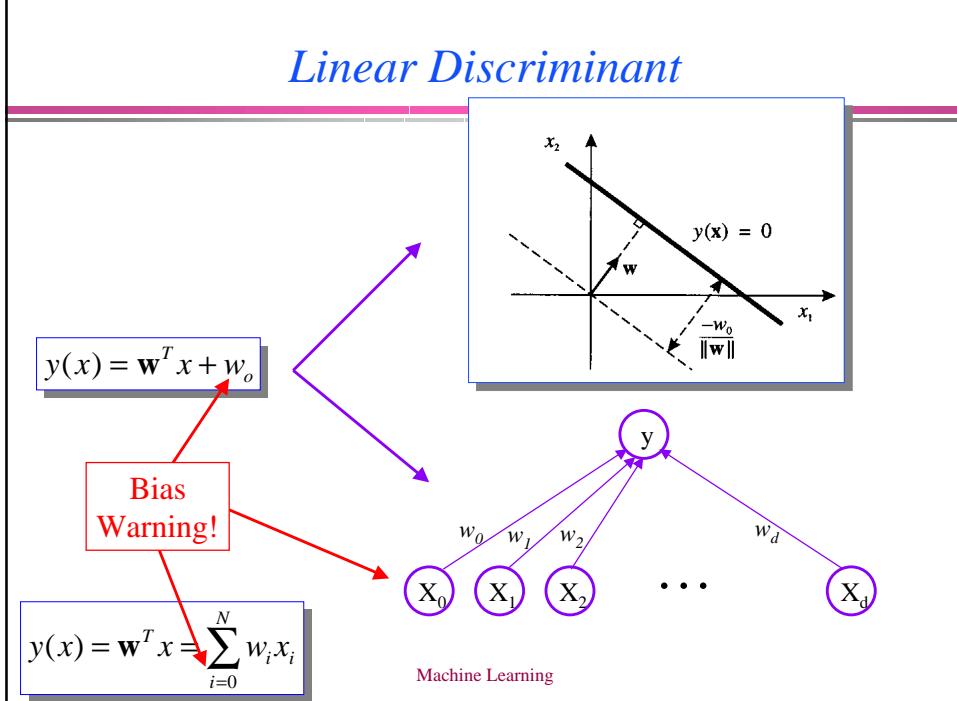
Two Class Gaussian  
same Covariance

- Alternatively, you may not know much about the density of your classes.
- Construct a function that classifies directly...

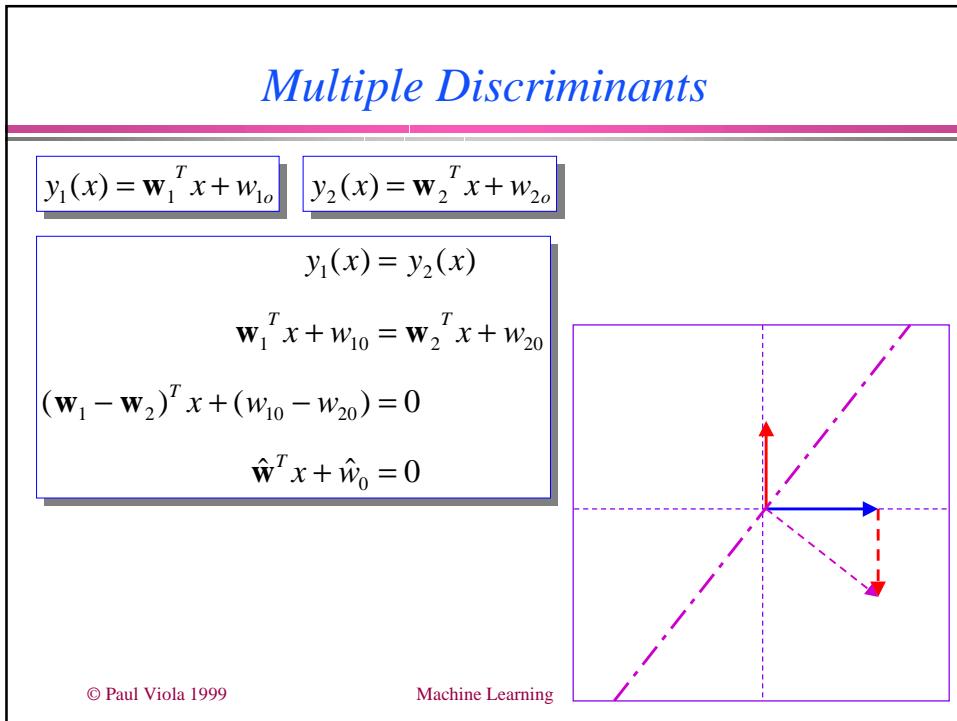
© Paul Viola 1999

Machine Learning

## Linear Discriminant



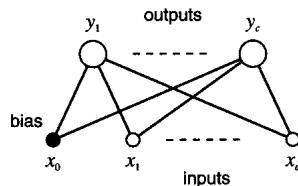
## Multiple Discriminants



*... in a single network*

$$y_k(x) = \sum_i w_{ki} x_i + w_{ko}$$

$w_{ki}$  weight matrix

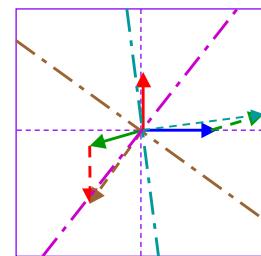
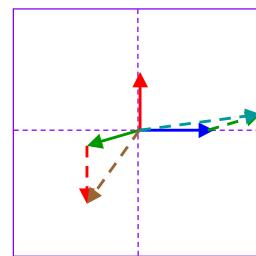
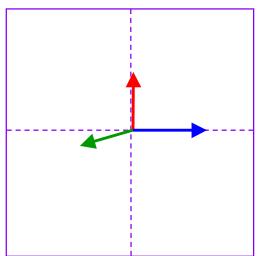


$$C(x) = C_k \text{ if } k = \arg \max_i y_i(x)$$

© Paul Viola 1999

Machine Learning

*Multiple Discriminants*



Intersection  
of Half Planes

© Paul Viola 1999

Machine Learning

## *How do we learn linear discriminants?*

- What are the principles?
  - » In density estimation we maximize likelihood.
  - » In classification we minimize errors.
- How do search for the best classifier?
- Will the search have local minima?

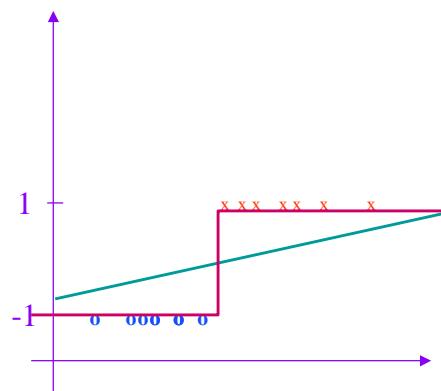
© Paul Viola 1999

Machine Learning

## *Perhaps this is really Regression?*

$$\begin{aligned} E(\mathbf{w}) &= \sum_j (y(x^j) - t^j)^2 \\ &= \sum_j (\mathbf{w}^T \mathbf{x}^j - t^j)^2 \end{aligned}$$

Minimize the squared error.



© Paul Viola 1999

Machine Learning

## Quadratic cost is very simple...

$$E(\mathbf{w}) = \sum_j (y(x^j) - t^j)^2 \\ = \sum_j (\mathbf{w}^T x^j - t^j)^2$$

$$E(W) = (XW - T)^T (XW - T) \\ = W^T X^T XW - 2X^T W^T T + T^T T$$

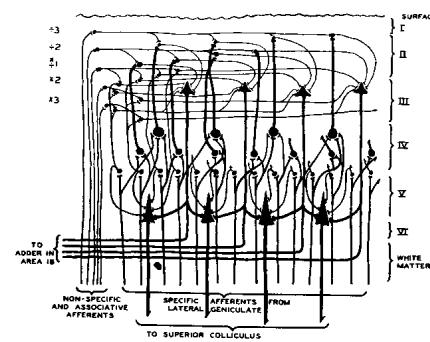
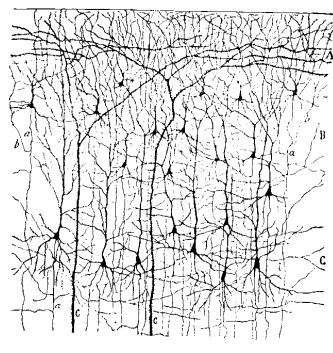
$$\frac{dE(W)}{dW} = 2X^T XW - 2X^T T = 0 \\ X^T XW = X^T T \\ W = (X^T X)^{-1} X^T T$$

- Direct linear expression for the weights given the training data.

© Paul Viola 1999

Machine Learning

## Is this a model for the brain?



Pitts and McCulloch, 1947

© Paul Viola 1999

Machine Learning

## What about Gradient Descent?

$$\begin{aligned} E(\mathbf{w}) &= \sum_j (y(x^j) - t^j)^2 \\ &= \sum_j (\mathbf{w}^T \mathbf{x}^j - t^j)^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial E(\mathbf{w})}{\partial w} &= 2 \sum_j (\mathbf{w}^T \mathbf{x}^j - t^j) \mathbf{x}^j \\ &= 2 \sum_j \delta^j x^j \end{aligned}$$

$$w_t = w_{t-1} - \eta \sum_j \delta^j x^j$$

© Paul Viola 1999

Machine Learning

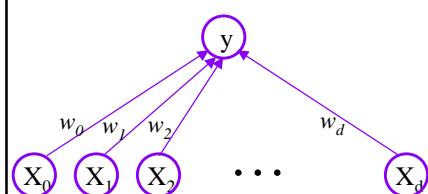
## Batch vs. On-line

$$E(\mathbf{w}) = \sum_j E^j = \sum_j (\delta^j)^2$$

Error has many components

$$w_t = w_{t-1} - \eta \frac{\partial E^j}{\partial w} = w_{t-1} - \eta \delta^j x^j$$

Pick an example at Random



- Pick Random Example
- Observe Output/Error
- Adjust Weights to Reduce Error

© Paul Viola 1999

Machine Learning

## *Can't Always Solve for the Weights...*

$$y(x) = g(w^T x)$$

$$g(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

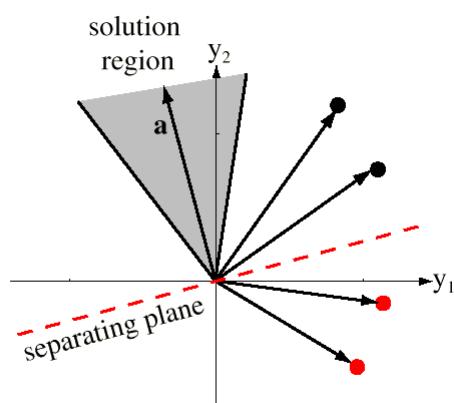
$$g(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Perceptrons - McCulloch and Pitts (1943)
  - » Originally as a model for real neurons

© Paul Viola 1999

Machine Learning

## *Perceptron*



© Paul Viola 1999

Machine Learning

## Perceptron Cost Function

$$E(\mathbf{w}) = \sum_j (g(\mathbf{w}^T \mathbf{x}^j) - t^j)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = -2 \sum_j (g(\mathbf{w}^T \mathbf{x}^j) - t^j) \frac{\partial g(\mathbf{w}^T \mathbf{x}^j)}{\partial \mathbf{w}}$$

Simple Gradient  
Descent does not work

$$E(\mathbf{w}) = - \sum_j (g(\mathbf{w}^T \mathbf{x}^j) - t^j)^2 (\mathbf{w}^T \mathbf{x}) t^j$$

$$= - \sum_{\text{errors}} (\mathbf{w}^T \mathbf{x}) t^j$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = -2 \sum_{\text{errors}} t^j \mathbf{x}^j$$

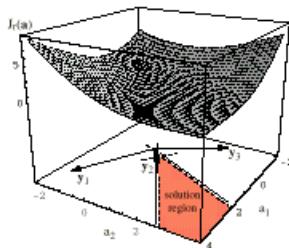
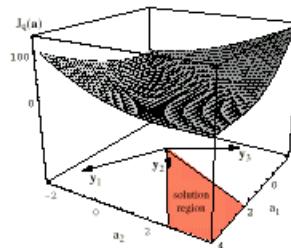
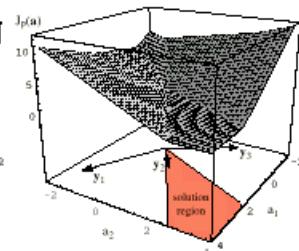
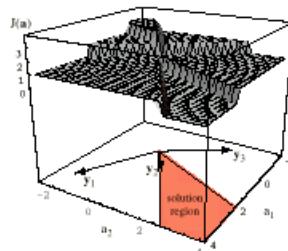
Perceptron  
Criterion

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta t^j \mathbf{x}^j$$

© Paul Viola 1999

Machine Learning

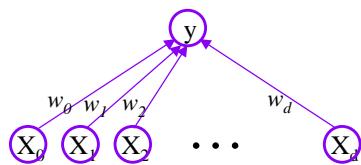
## Different Error Measures



© Paul Viola 1999

Machine Learning

## Perceptron Learning



© Paul Viola 1999

Machine Learning

## Real Perceptrons

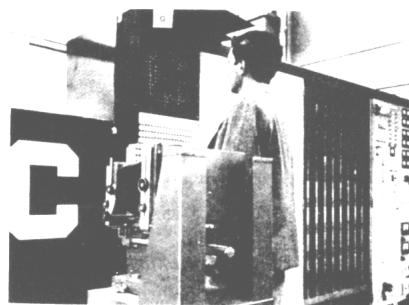
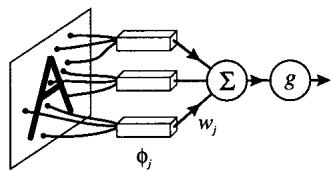
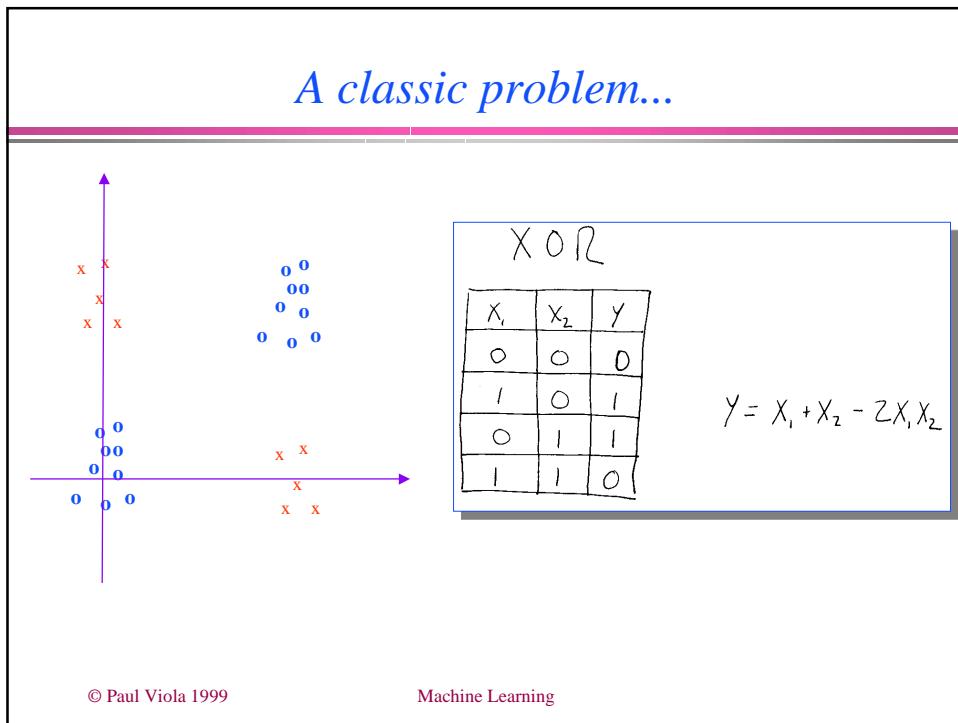
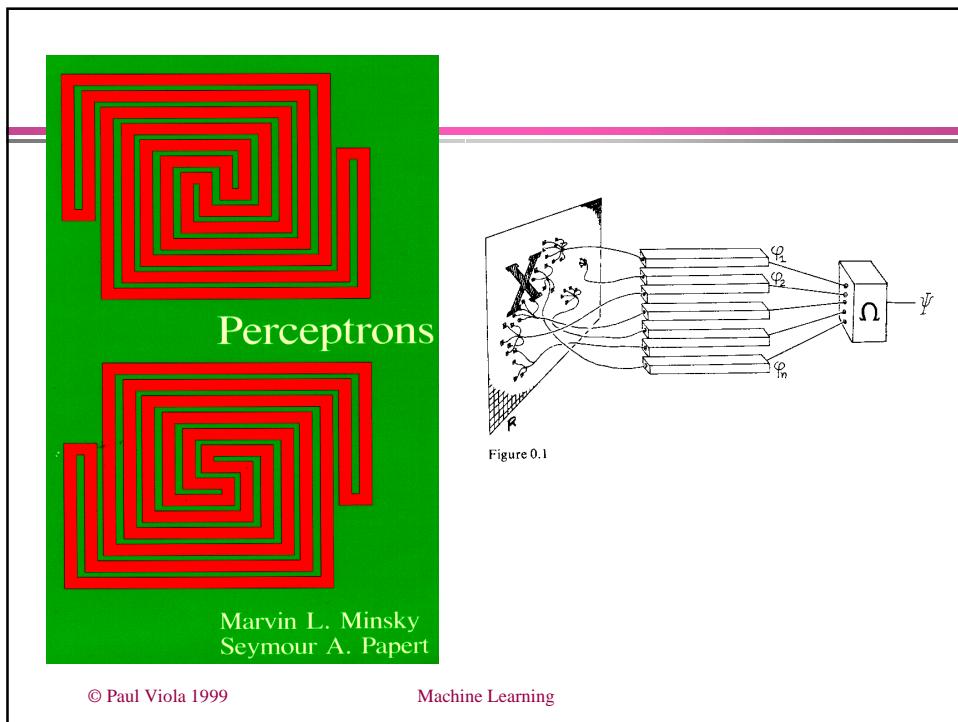


FIG. 3. Mark I Perceptron at Cornell Aeronautical laboratory. (a) Overall view with sensory input at left, association units in center, and control panel and response units at far right. The sensory to associator plugboard, shown in (b) is located behind the closed panel to the right of the operator. The image of the letter "C" on the front panel is a repeater display, for monitoring sensory inputs.



© Paul Viola 1999

Machine Learning



# *6.891 Machine Learning and Neural Networks*

## Lecture 7: Multi-Layer Perceptrons Back Propagation

© Paul Viola 1999

Machine Learning

## *News*

- Pset 3 is on the web
  - » Includes a classifier "shootout"
  - » The mystery dataset has 20 dimensions and two classes
  - » Winner gets \$10 of Toscanini's
- Pset 2 looks great ...
  - » Many of you did a lot of work.

© Paul Viola 1999

Machine Learning

## Review & Overview

- Lecture 6:
  - » Linear Discriminants
  - » Perceptrons
  - » Training Perceptrons
- Generalized Perceptrons
- Multi-layer Perceptrons
  - » Multi-Layer Derivatives
  - » Back Propagation
- Examples:
  - » NET Talk

© Paul Viola 1999

Machine Learning

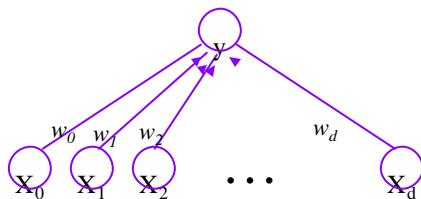
## On-line learning of Perceptrons

- 1: Error Function  
(Criteria)

$$E(\mathbf{w}) = \sum_j E^j$$

- 2: Update Rule

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \mathbf{h} \frac{\partial E^j}{\partial \mathbf{w}} = \mathbf{w}_{t-1} - \mathbf{h} \mathbf{d}^j x^j$$



- Pick Random Example
- Observe Output/Error
- Adjust Weights to Reduce Error

© Paul Viola 1999

Machine Learning

## Different Criteria...

### Linear Discriminant

$$E(\mathbf{w}) = \sum_j (\mathbf{w}^T \mathbf{x}^j - t^j)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w} = 2 \sum_j (\mathbf{w}^T \mathbf{x}^j - t^j) \mathbf{x}^j$$

$$= 2 \sum_j \mathbf{d}^j \mathbf{x}^j$$

$$w_t = w_{t-1} - \mathbf{h} d^j x^j$$

### Perceptron

$$E(\mathbf{w}) = - \sum_j (g(\mathbf{w}^T \mathbf{x}^j) - t^j)^2 (\mathbf{w}^T \mathbf{x}) t^j$$

$$= - \sum_{\text{errors}} (\mathbf{w}^T \mathbf{x}) t^j$$

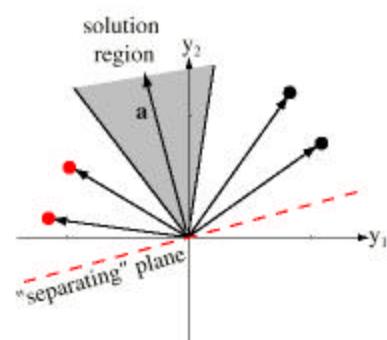
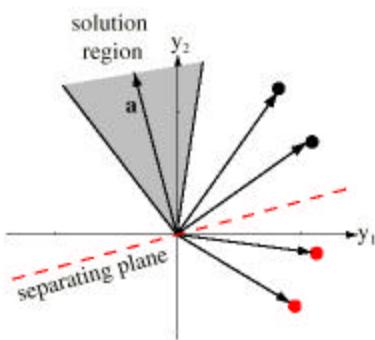
$$\frac{\partial E(w)}{\partial w} = -2 \sum_{\text{errors}} t^j x^j$$

$$w_t = w_{t-1} - \mathbf{h} t^j x^j$$

© Paul Viola 1999

Machine Learning

## Normalizing examples...



$$w_t = w_{t-1} - \mathbf{h} x^j$$

For errors only!

© Paul Viola 1999

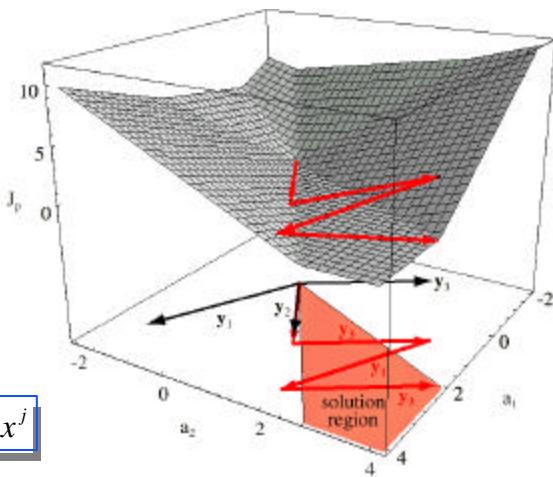
Machine Learning

## The update rule in action...

$$w_t = w_{t-1} - x^j$$

© Paul Viola 1999

Machine Learning



## Real Perceptrons

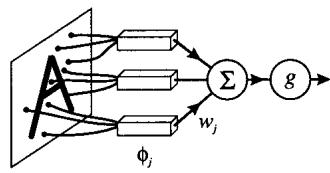


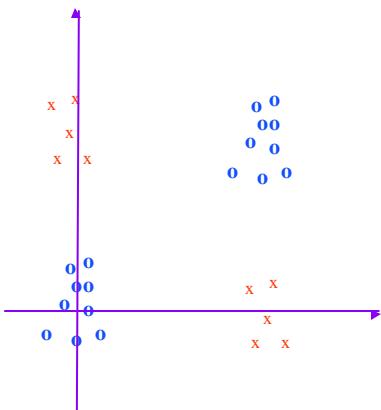
FIG. 3. Mark I Perceptron at Cornell Acoustical laboratory. (a) Overall view with sensory input at left, association units in center, and control panel and response units at far right. The sensory to associator plugboard, shown in (b), is located behind the closed panel to the right of the operator. The image of the letter "C" on the front panel is a repeater display, for monitoring sensory inputs.



© Paul Viola 1999

Machine Learning

## A classic problem...



XOR

| $X_1$ | $X_2$ | $Y$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 1     | 0     | 1   |
| 0     | 1     | 1   |
| 1     | 1     | 0   |

$$Y = X_1 + X_2 - 2X_1X_2$$

© Paul Viola 1999

Machine Learning

## Generalized Perceptron

$$XOR(x) = x_1 + x_2 - 2x_1x_2$$

$$x_i \in \{0,1\}$$

$$y(x) = g(w^T x)$$

Can't do that!

$$\hat{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_1x_2 \end{bmatrix}$$



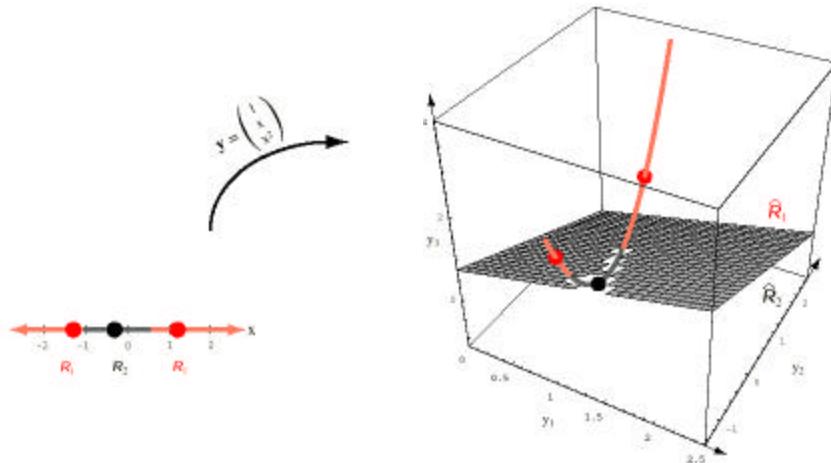
$$\hat{w} = \begin{bmatrix} 1 \\ 1 \\ -2.1 \end{bmatrix}$$

Works Great

© Paul Viola 1999

Machine Learning

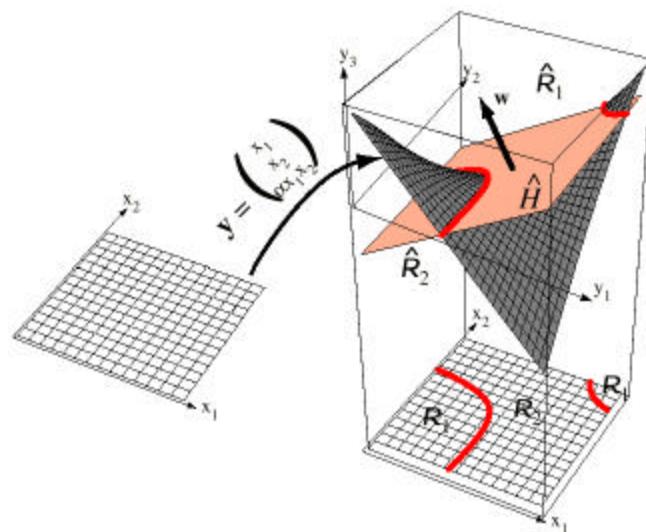
## Another Generalized Perceptron



© Paul Viola 1999

Machine Learning

*Adding a single feature can yield complex classifications...*



© Paul Viola 1999

Machine Learning

## Two Dilemmas

- How does one find/define the correct set of features?
- How many will you need?
- 1950's answers:
  - » Don't know... we'll just think them up.
  - » Don't know... we'll just keep adding wires.

© Paul Viola 1999

Machine Learning

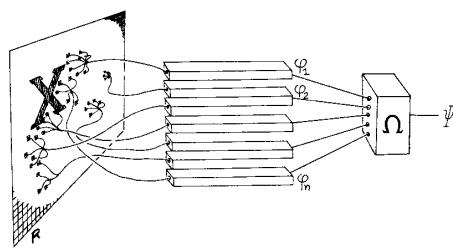
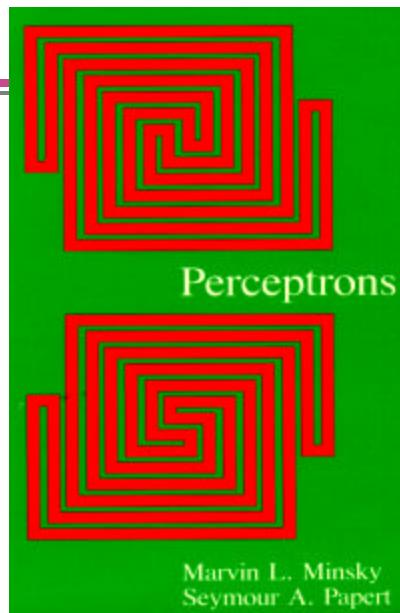


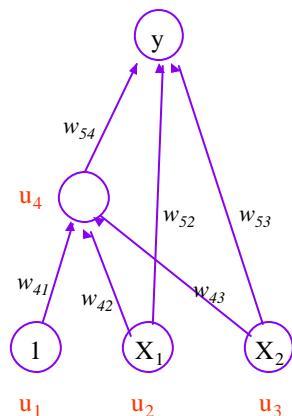
Figure 0.1

1968: The Death of  
Neural Networks

© Paul Viola 1999

Machine Learning

## Multiple Layers



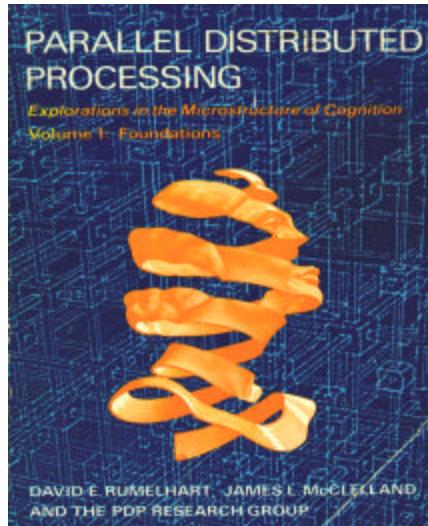
$$W = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1.5 & 1 & 1 & 0 \\ 0 & 1 & 1 & -2.5 \end{bmatrix}$$

How can we learn this??

© Paul Viola 1999

Machine Learning

## 1986: The Rebirth of Neural Networks



- PDP Group had Huge Impact

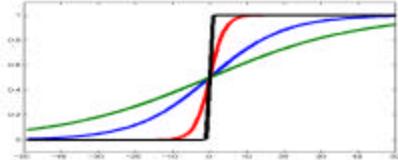
© Paul Viola 1999

Machine Learning

## 1980's: Perhaps Gradient Descent?

$$y(x) = s(w^T x)$$

$$s(a) = \frac{1}{1 + e^{-a}}$$



$$E(\mathbf{w}) = \sum_j (s(w^T x^j) - t^j)^2$$

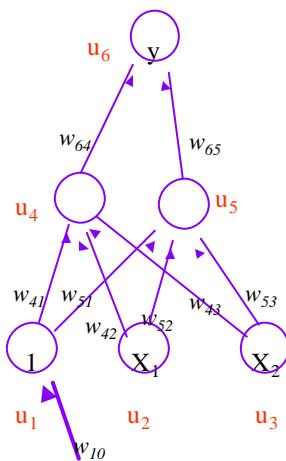
$$\frac{\partial E(w)}{\partial w} = -2 \sum_j (s(w^T x^j) - t^j) \frac{\partial s(w^T x^j)}{\partial w}$$

$$\frac{\partial s(u)}{\partial w} = s(u)(1 - s(u)) \frac{\partial u}{\partial w}$$

$$\frac{\partial E(w)}{\partial w} = -2 \sum_j (s(w^T x^j) - t^j) s(w^T x^j) (1 - s(w^T x^j)) x^j$$

© Paul Viola

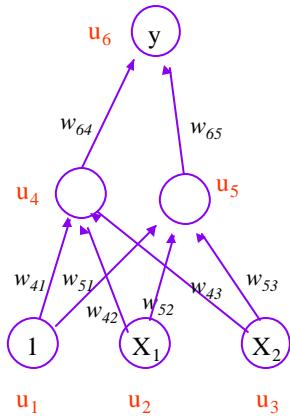
## Sigmoid Multi-Layer Network



$$y(x) = s(w_{54}u_4 + w_{55}u_5) \\ = s(w_{54} s(w_{41}u_1 + w_{42}u_2 + w_{43}u_3) \\ + w_{55}s(w_{51}u_1 + w_{52}u_2 + w_{53}u_3))$$

$$E(\mathbf{w}) = \sum_j (s(w^T x^j) - t^j)^2 \\ \frac{\partial E(w)}{\partial w} = -2 \sum_j (s(w^T x^j) - t^j) \frac{\partial s(w^T x^j)}{\partial w}$$

## Multi-Layer Conventions



$$u_k = g\left(\sum_j w_{kj} u_j\right)$$

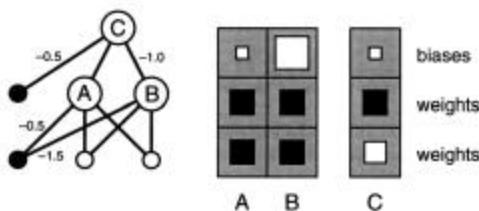
$$a_k = \sum_j w_{kj} u_j$$

- Networks must not have loops...
- Units are ordered:  
»  $i > k \rightarrow u_i$  is not an input to  $u_k$
- Compute Units in order

© Paul Viola 1999

Machine Learning

## More Conventions



- If Units are organized in Layers  
» Layers can be computed in parallel.

$$u_2 = g\left(\left[\begin{array}{c} W_{21} \end{array}\right] * \left[\begin{array}{c} u_1 \end{array}\right]\right)$$

$$u_3 = g\left(\left[\begin{array}{c} W_{32} \end{array}\right] * \left[\begin{array}{c} u_2 \end{array}\right]\right)$$

© Paul Viola 1999

Machine Learning

Gradient Descent or Error

$$E = \sum_i (y(i) - t^i)^2 \quad a_2 = \sum_{k=1}^K u_k \cdot a_k$$

$$= \sum_i E^i \quad u_k = g(a_k)$$

$$y(i) = g\left(\sum_{k=1}^K u_k \cdot j\left(\frac{2}{m} u_k + \frac{1}{m} b_k\right)\right) = g(a_k)$$

$$\frac{\partial E^i}{\partial u_j} = 2(y(i) - t^i)$$

$$\frac{\partial y(i)}{\partial a_k} = g'(a_k)(1-g(a_k))$$

$$\frac{\partial a_k}{\partial u_j} = u_k$$

$$\frac{\partial E^i}{\partial u_j} = \frac{\partial E^i}{\partial y(i)} \cdot \frac{\partial y(i)}{\partial a_k} \cdot \frac{\partial a_k}{\partial u_j}$$

$$\frac{\partial a_k}{\partial u_j} = \frac{\partial}{\partial u_j} \left[ \sum_{k=1}^K u_k \cdot j\left(\frac{2}{m} u_k + \frac{1}{m} b_k\right) \right] = \frac{\partial u_k}{\partial u_j} = 1$$

$$\frac{\partial E^i}{\partial u_j} = \frac{\partial E^i}{\partial u_j} \left[ u_j \cdot u_k (1-u_k) \right] \delta_k$$

© Paul Viola 1999      macmine

$$\frac{\partial E^i}{\partial w_{3A}} = \frac{\partial E^i}{\partial a_3} \cdot \frac{\partial a_3}{\partial w_{3A}}$$

$$= \delta_3 \cdot u_A$$

$$= \left( \sum_{k=1}^K \frac{\partial E^i}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_3} \right) u_A$$

$$= \underbrace{\left[ \sum_{k=1}^K \delta_k \left( u_{k3} \cdot u_k (1-u_k) \right) \right]}_{\delta_3} u_A$$

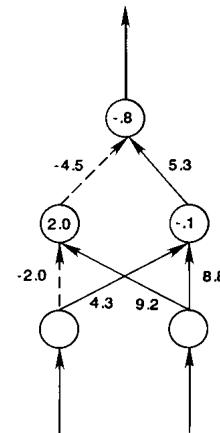
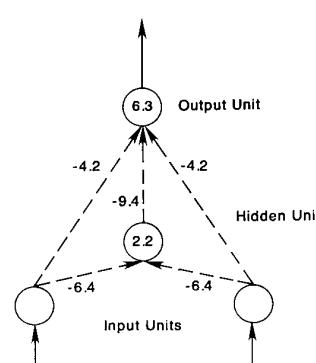
$$\delta_j = u_j (1-u_j) \left[ \sum_{i: a_{ij} \neq a_{ti}} w_{ij} \delta_i \right]$$

© Paul Viola 1999

## Solving XOR (big deal?)

XOR

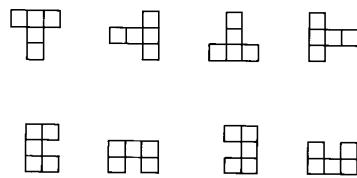
| $X_1$ | $X_2$ | $Y$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 1     | 0     | 1   |
| 0     | 1     | 1   |
| 1     | 1     | 0   |



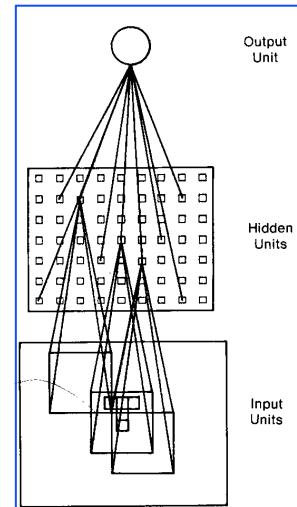
© Paul Viola 1999

Machine Learning

## Vision Applications (sort of)



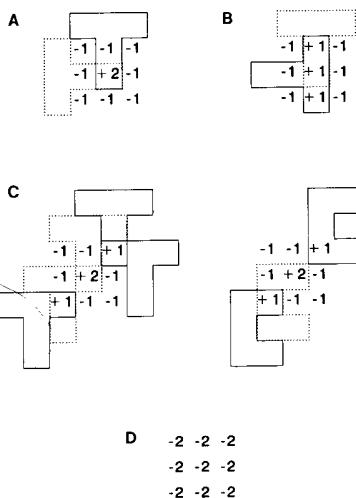
T's vs. C's



© Paul Viola 1999

Machine Learning

## Very Simple Solution



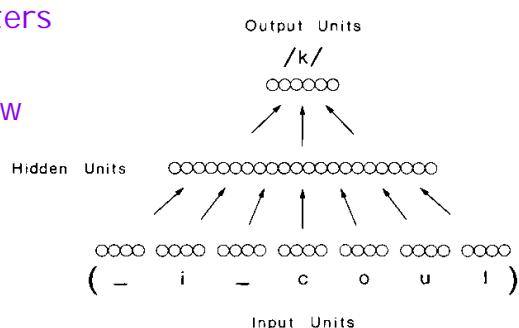
© Paul Viola 1999

Machine Learning

## NETtalk (1986) First Real Application

- Task: Pronounce English text
  - » Text → Phonemes
- Example: "This is it." /s/ vs. /z/
- 29 possible characters
- 26 phonemes
- 7 Character Window
- Structure
  - » 203 Inputs
  - » 26 Output
  - » 80 Hidden

95% Accurate



© Paul Viola 1999

Machine Learning

# *6.891 Machine Learning and Neural Networks*

## Lecture 8: Back Prop and Beyond

© Paul Viola 1999

Machine Learning

## *News*

- Mid-term will be on 10/20
  - » Here in this room.
  - » It should take about 1 hour... but we will give you 1.5
    - Show up on time, please.
  - » Coverage: Psets 1, 2 and 3.
    - Density estimation (Parametric, Semi and Non-parametric)
    - Bayesian Classification
    - Discriminants (Linear, Perceptron, Multi-layer)

© Paul Viola 1999

Machine Learning

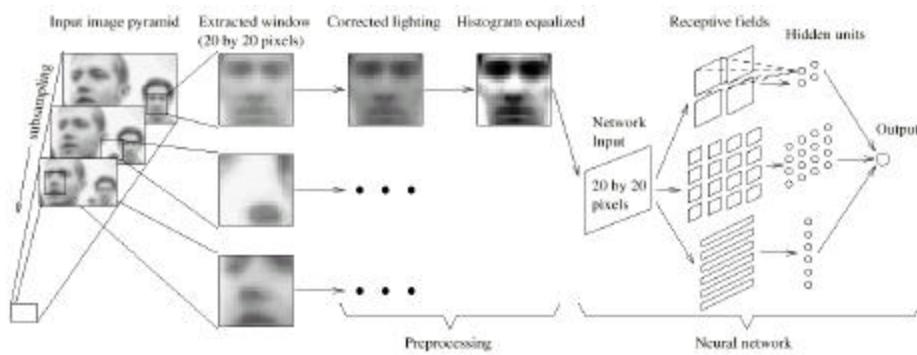
## *Review & Overview*

- Lecture 7:
  - » Multi-Layer Derivatives
  - » Back Propagation
  - » Examples:
    - NET Talk
- Why 6.891 is not Over!
  - » Bugs with Gradient Descent
  - » Local Min
  - » Bias and Variance
    - How many units?
  - » Variants

© Paul Viola 1999

Machine Learning

## *Face Detection Network: General Layout*



Baluja, Rowley, and Kanade

© Paul Viola 1999

Machine Learning

## Intensity Preprocessing

Oval mask for ignoring background pixels:



Original window:



Best fit linear function:



Lighting corrected window:  
(linear function subtracted)



Histogram equalized window:



© Paul Viola 1999

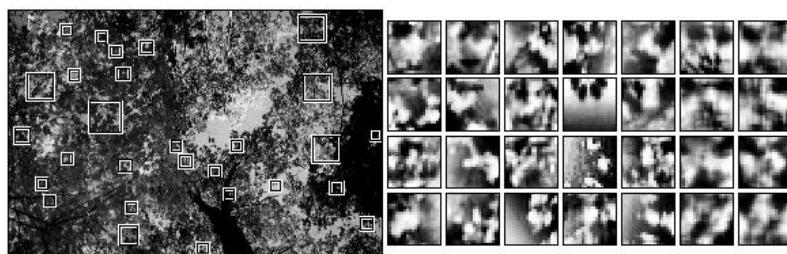
Machine Learning

## Training Data

Positives



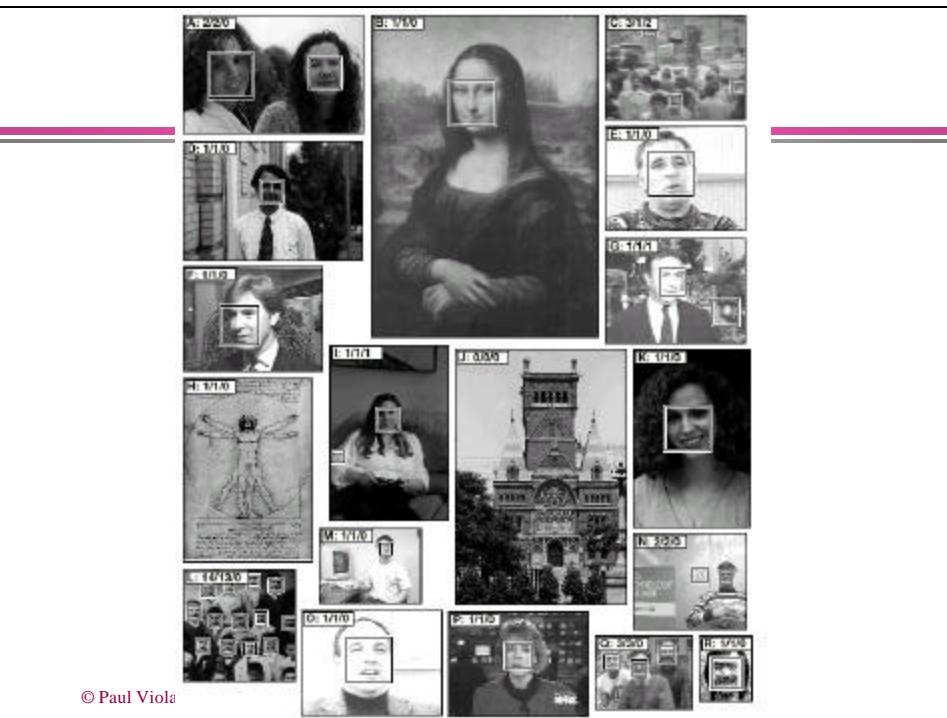
Negatives



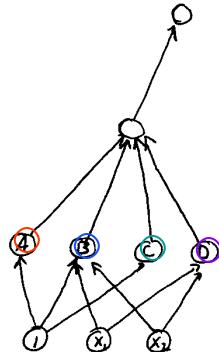
© Paul Viola 1999

Machine Learning

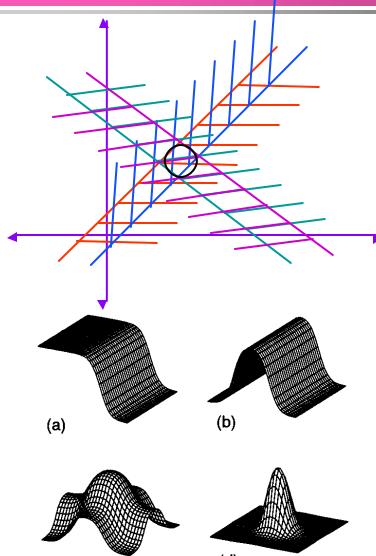
## Performance



## MLP: How Powerful?



© Paul Viola 1999



Machine I

(a)

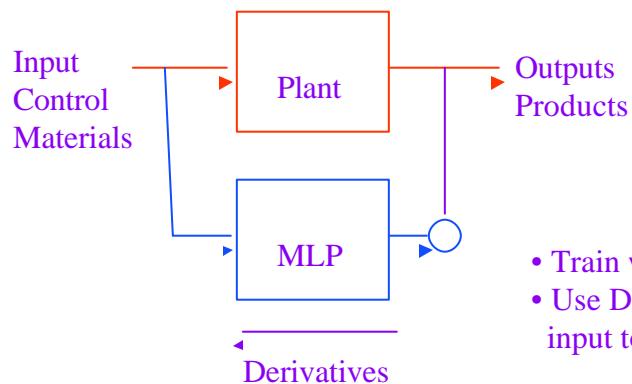
(b)

(c)

(d)

## Derivatives are Cheap

- Modeling Factories/Plants



- Train with Back Prop
- Use Derivs to Modify input to improve output

© Paul Viola 1999

Machine Learning

## 1990: The height of MLP's and Back Prop

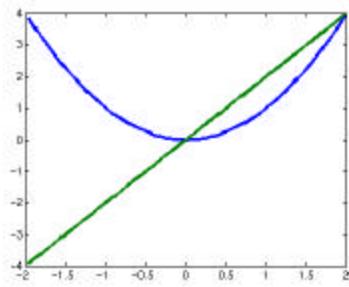
- Multi-layer perceptrons can solve any approximation problem (in principle)
  - » Given 3 layers
  - » Given an infinite number of units and weights
- There is no direct technique for finding the weights (unlike linear discriminants)
- Gradient descent (using Back Prop) comes to dominate discussion in the Neural Net community
  - » Can you find a good set of weights quickly?
    - How can you speed things up?
  - » Will you get stuck in local minima?
- A small group in the community also worries about generalization.

© Paul Viola 1999

Machine Learning

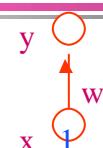
## How long 'til we find the Min?

- Simplest Case
  - » 1 Weight, Quadratic Error Function



$$E(w) = (wx - y)^2$$

$$\begin{aligned} &= (w - 0)^2 \\ &= w^2 \end{aligned}$$



$$\frac{\partial E}{\partial w_{t-1}} = 2w$$

$$\boxed{h = \frac{1}{2}}$$

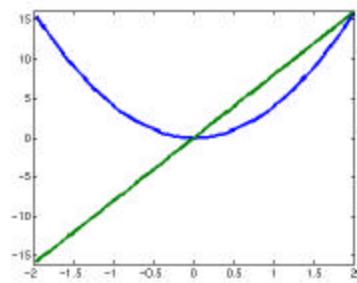
© Paul Viola 1999

Machine Learning

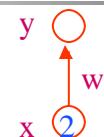
## Scale the Input

### ● Simplest Case

» 1 Weight, Quadratic Error Function



$$E(w) = (w - 0)^2 \\ = 4w^2$$



$$w_t = w_{t-1} - \eta \frac{\partial E}{\partial w_{t-1}}$$

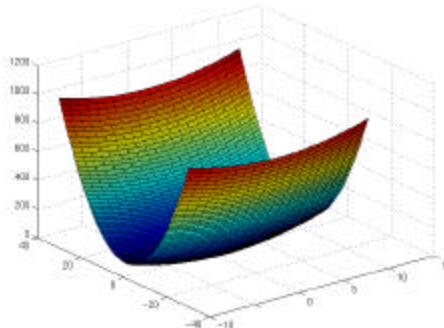
$$\frac{\partial E}{\partial w_{t-1}} = 8w$$

Hack 1: Start eta very small

Increase if Error decreases ↴

More Hacks Coming!

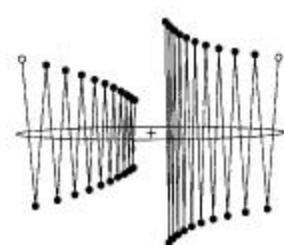
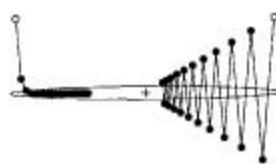
## Multiple Weights



Hack 2: Momentum

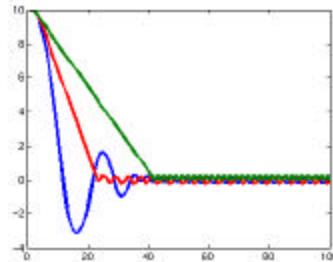
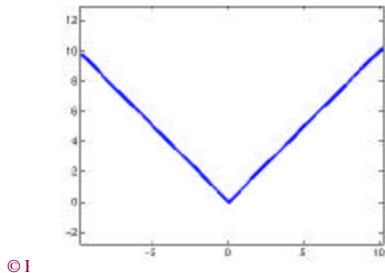
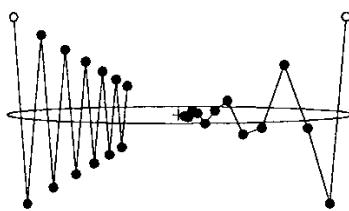
0.020  
0.047  
0.049  
0.050

© Paul Viola 1999



## Momentum

$$\Delta w_t = -\mathbf{h} \frac{\partial E}{\partial w_{t-1}} + \mathbf{a} \Delta w_{t-1}$$



## Second Order Techniques

- Gradient descent assumes a locally linear cost function:

$$\Delta w_t = -\mathbf{h} \frac{\partial E}{\partial w_{t-1}} = -\mathbf{h} E'$$

$$\begin{aligned} E(w+\Delta) &= E(w) + E'(w)\Delta + \mathbf{e} \\ &= E(w) - (E'(w))^2 \end{aligned}$$

- Second order techniques assume locally quadratic:

$$\Delta w_t = -\mathbf{h} \frac{E'}{E''}$$

$$\begin{aligned} E(w) &= aw^2 + bw + c \\ E' &= 2aw + b \\ E'' &= 2a \\ \frac{E'}{E''} &= w + \frac{b}{2a} \end{aligned}$$

$$\begin{aligned} w_1 &= w_0 + \Delta w \\ &= w_0 - \left( w_0 + \frac{b}{2a} \right) \\ &= -\frac{b}{2a} \end{aligned}$$

© Paul Viola 1999

Machine Learning

## *More Principled Hacks...*

- Second Order Techniques
  - » N weights -->  $N^2$  Hessian entries
  - » Also Destabilizes learning
- Line Search
  - » Expensive but hard to beat



© Paul Viola 1999

Machine Learning

## *Local Minima*

- Number of Papers
  - » 1000's of local minima in simple problems (XOR)
- One More Trick
  - » Linear is good...
- Small Input Range
  - » Sigmoid is almost linear
- \*\* Start weights near zero...

© Paul Viola 1999

Machine Learning

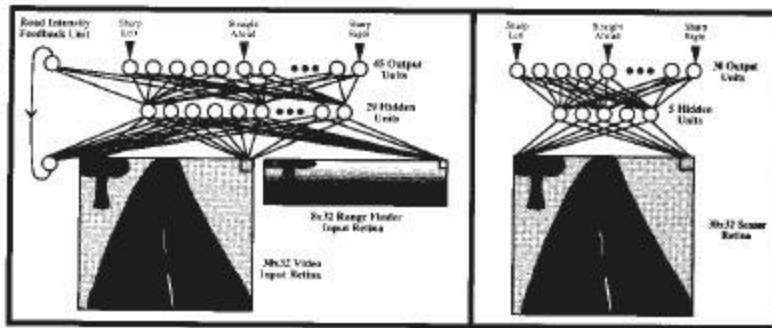
## Bias and Variance

- How many layers are right?
- How many units per layer?
- What about structural constraints?
- \*\*\* We don't know the answers \*\*\*

© Paul Viola 1999

Machine Learning

## ALVINN



Pomerleau

© Paul Viola 1999

Machine Learning

## *No Hands Across America*

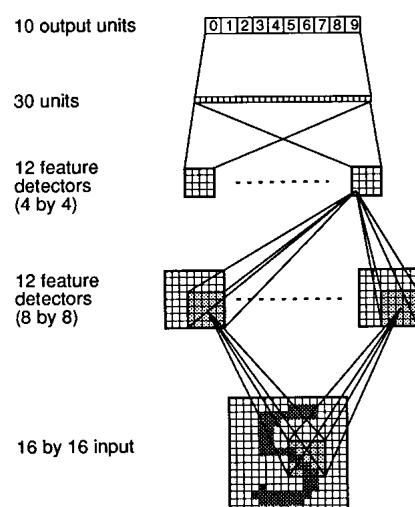


© Paul Viola 1999



Machine Le:

## *Zip Codes*



Le Cun

© Paul Viola 1999

# *6.891 Machine Learning and Neural Networks*

## Lecture 9: On to Support Vector Techniques

© Paul Viola 1999

Machine Learning

## *News*

- Final will be 12/13 at 1:30PM
  - » If you have a conflicting final let us know.
- Remember that almost all the material appears in the book...
  - » Right now we are jumping back and forth between
    - Chapter 5
    - Chapter 6

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 8:
  - » Multi-layer Perceptrons
  - » Back propagation
  - » Hacks (... many)
- Why did we discard Perceptrons?
- Kernel function network
- Define the Support Vector framework

© Paul Viola 1999

Machine Learning

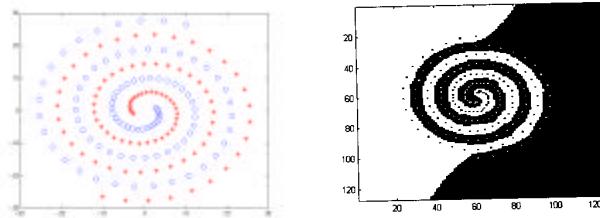
## *History Lesson*

- 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

© Paul Viola 1999

Machine Learning

## Why did we need multi-layer perceptrons?

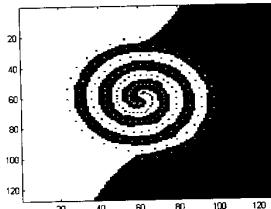


- Problems like this seem to require very complex non-linearities.
- Minsky and Papert showed that an exponential number of features is necessary to solve generic problems.

© Paul Viola 1999

Machine Learning

## Why an exponential number of features?



$$\Phi(x) = \begin{Bmatrix} x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5 \dots \\ x_1^4, x_1^3x_2, x_1^2x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5 \\ \vdots \end{Bmatrix}$$

14th Order???  
120 Features

$n$ : variables

$k$ : order poly

$$\binom{n+k}{k} = \frac{(n+k)!}{k!n!} \in O(\min(n^k, k^n))$$

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

© Paul Viola 1999

## *MLP's vs. Perceptron*

- MLP's are incredibly 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.

© Paul Viola 1999

Machine Learning

## *Perceptron Training is Linear Programming*

$$\sum_i w_i \mathbf{x}_i^l > 0 \quad \forall l$$

- After Normalization
- After adding bias
- Assumes no errors

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

What about linearly inseparable?

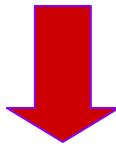
$$\sum_i w_i \mathbf{x}_i^l + s_l > 0 \quad \forall l \qquad \min \sum_l s_l$$
$$s_l > 0 \quad \forall l$$

© Paul Viola 1999

Machine Learning

## *Rebirth of Perceptrons*

- How to train efficiently.
  - » Linear Programming (... later quadratic programming)
- How to get so many features inexpensively?!?
- How to generalize with so many features?
  - » Occam's revenge.



## Support Vector Machines

© Paul Viola 1999

Machine Learning

### *Lemma 1: Weight vectors are simple*

$$w_0 = 0 \quad \Delta w_t = \mathbf{h} \mathbf{x}^t$$

$$w_t = \sum_{\text{errors}} \mathbf{h} \mathbf{x}^t = \sum_l b_l \mathbf{x}^l \quad w_t = \sum_l b_l \Phi(\mathbf{x}^l)$$

- 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.

© Paul Viola 1999

Machine Learning

## Lemma 2: Only need to compare examples

$$\begin{aligned}\gamma(\tilde{x}) &= \sum_i w_i \phi_i(\tilde{x}) \\ &= \sum_i \left[ \sum_{\ell} b_{\ell} \phi(x^{\ell}) \right]_i \phi_i(x) \\ &= \sum_{\ell} b_{\ell} \sum_i \phi_i(x^{\ell}) \phi_i(x) \\ &= \sum_{\ell} b_{\ell} k(x^{\ell}, x)\end{aligned}$$

$$\begin{aligned}k(x^{\ell}, x) &= \sum_i \phi_i(x^{\ell}) \phi_i(x) \\ &= k_{\ell}(x)\end{aligned}$$

© Paul Viola 1999

Machine Learning

## Perceptron Rebirth: Generalization

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

$$\begin{aligned}\sum_j b_j K(\mathbf{x}^l, \mathbf{x}^j) + s_l > 0 \quad \forall l \quad &\min \sum_l s_l \\ s_l > 0 \quad \forall l \quad &\min \sum_j b_j^2\end{aligned}$$

Smoother

But this is unstable!!

© Paul Viola 1999

Machine Learning

## *Linear Program is not unique*

The linear could return any multiple of the correct weight vector...

$$\sum_i \hat{w}_i \mathbf{x}_i^l > 0 \quad \forall l \quad \sum_i (\mathbf{I} \hat{w}_i) \mathbf{x}_i^l > 0 \quad \forall l$$

Slack variables & Weight prior

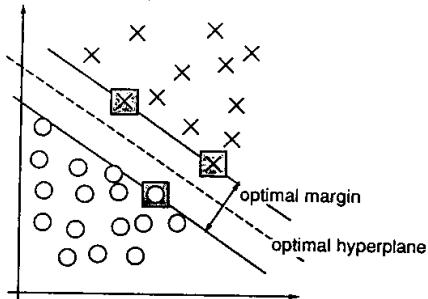
- Force the solution toward zero

$$\begin{aligned} \sum_i w_i \mathbf{x}_i^l + s_l &> 0 \quad \forall l & \min \sum_l s_l \\ s_l &> 0 \quad \forall l & \min \sum_i w_i^2 \end{aligned}$$

© Paul Viola 1999

Machine Learning

## *Definition of the Margin*



- Margin: Gap between negatives and positives measured perpendicular to a hyperplane

© Paul Viola 1999

Machine Learning

## *Require non-zero margin*

$$\sum_i w_i \mathbf{x}_i^l + s_l > 0 \quad \forall l \quad \begin{array}{l} \text{Allows solutions} \\ \text{with zero margin} \end{array}$$

$$\sum_i w_i \mathbf{x}_i^l + s_l > 1 \quad \forall l \quad \begin{array}{l} \text{Enforces a non-zero} \\ \text{margin between examples} \\ \text{and the decision boundary.} \end{array}$$

© Paul Viola 1999

Machine Learning

## *Constrained Optimization*

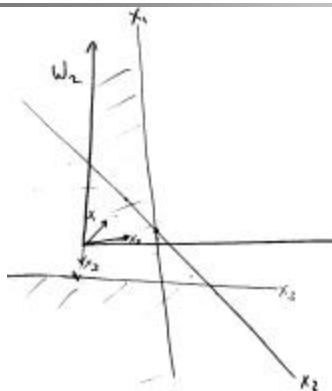
$$\begin{aligned} \sum_j b_j K(\mathbf{x}^l, \mathbf{x}^j) + s_l &> 1 \quad \forall l & \min \sum_l s_l \\ s_l &> 0 \quad \forall l & \min \sum_j b_j^2 \end{aligned}$$

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

© Paul Viola 1999

Machine Learning

## Constrained Optimization 2



$$\min \frac{1}{2} w^T w$$

$$w^T x' \geq 1$$

$$w^T x'' \geq 1$$

$$w^T x''' \leq -1$$

$$w^T x' = 1$$

$$w = \alpha_1 x' + \beta_1 x''$$

$$w^T x'' = 1$$

$$w = \alpha_2 x'' + \beta_2 x'''$$

$$\begin{bmatrix} -x_1 \\ -x_2 \end{bmatrix} \begin{bmatrix} w \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$w = \text{One } \setminus X$$

© Paul Viola 1999

$x^3$  is inactive

## Support Vectors

- Many of the B's are zero -- inactive constraints
- Guaranteed to generalize well
  - » VC Dimension -- end of semester

© Paul Viola 1999

Machine Learning

## SVM: examples

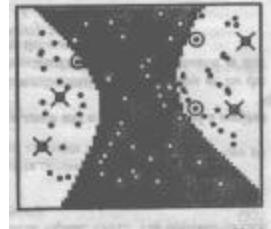
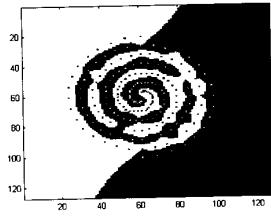
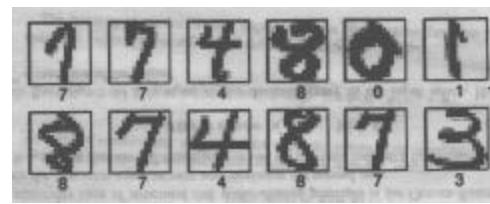
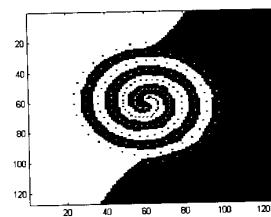


Figure 1: Kernel-Perceptron (small margin) clearly fails.



Machine Learning

## SVM: Key Ideas

- Augment inputs with a very large feature set  $\Phi(x)$ 
  - » Polynomials, etc.
- Use Kernel Trick(TM) to do this efficiently
- Enforce/Encourage Smoothness with weight penalty
  - » Minimize  $\|w\|^2 = \sum_i w_i^2$
- Introduce Margin so that:  $w_i \neq 0 \forall i$ 
  - » Set of linear inequalities
- Find best solution using Quadratic Programming

## SVM: Difficulties

- How do you pick the kernels?
  - » Intuition / Luck / Magic / ...
- What if the data is not linearly separable?
  - » i.e. the constraints cannot be satisfied

$$\forall j \quad (2t^j - 1) \left[ \sum_i w_i K(x^j, c^i) \right] \geq 1 - \epsilon_j$$

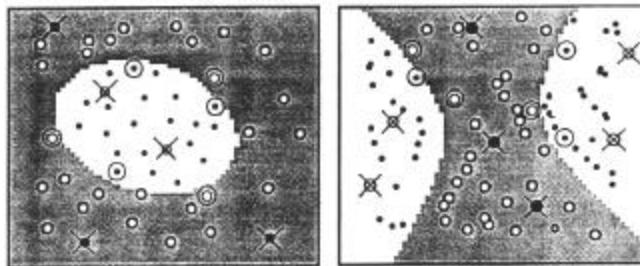
$$\min \left( w^T w + c \sum_j |\epsilon_j| \right) \quad \text{Slack Variables}$$

© Paul Viola 1999

Machine Learning

## SVM: Simple Example

6 weights



- Data dimension: 2
- Feature Space: 2nd order polynomial
  - » 4 dimensional

© Paul Viola 1999

Machine Learning

## SVM versus Perceptron

- Why not just use a perceptron?
  - » Use all training points as centers

$$y(x) = \Theta\left(\sum_i w_i^T K(x, c^i)\right)$$

- » Update using perceptron rule:

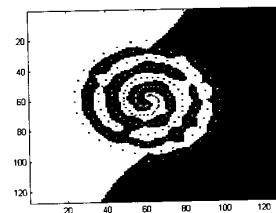
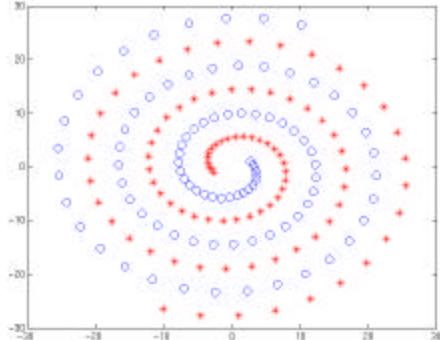
$$w_i^t = w_i^{t-1} + h K(x, c^i)$$

- Perceptron is not necessarily smooth...

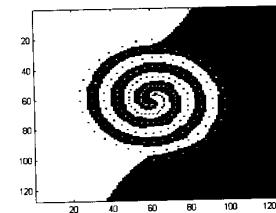
© Paul Viola 1999

Machine Learning

## Perceptrons are not smooth...



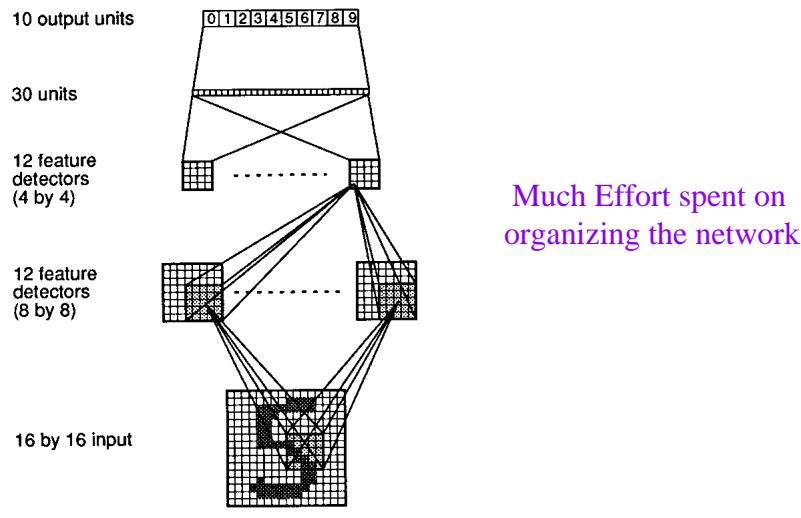
e 1: Kernel-Perceptron (small margin) clear  
ts



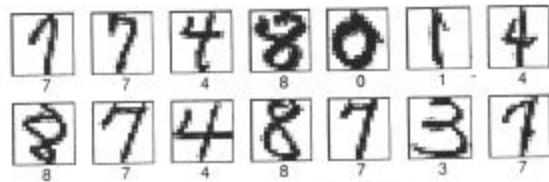
© Paul Viola 1999

Machine Learning

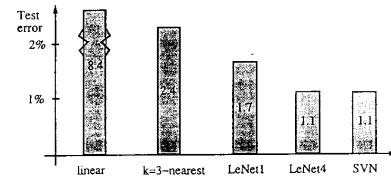
## Zip Codes



## 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    |

© Paul Viola

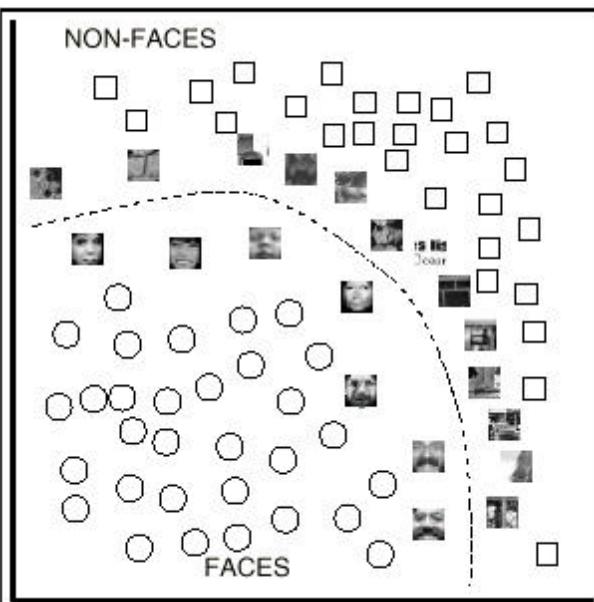


### SVM: Faces

|                    | Test Set A  |              | Test Set B  |              |
|--------------------|-------------|--------------|-------------|--------------|
|                    | Detect Rate | False Alarms | Detect Rate | False Alarms |
| SVM                | 97.1 %      | 4            | 74.2%       | 20           |
| Sung <i>et al.</i> | 94.6 %      | 2            | 74.2%       | 11           |

Machine Learning

### Support Vectors



© Paul V

# *6.891 Machine Learning and Neural Networks*

Lecture 10:  
Support Vector Machines  
More Details and Derivations

© Paul Viola 1999

Machine Learning

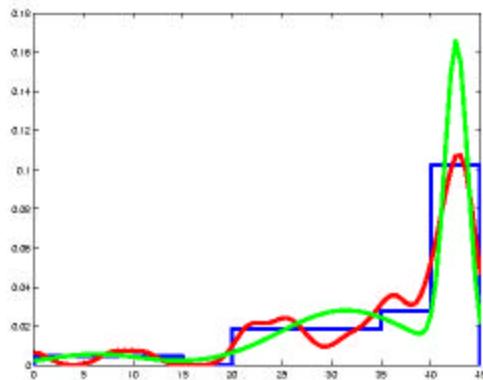
## *News*

- Quiz is 1 week from today.
- Problem set 4 will go out right after the quiz
  - » In one week... it's a pain to do two things at once.
- Problems set are very good (once again).
  - » On an absolute scale many of you are getting A's.

© Paul Viola 1999

Machine Learning

## *Pset 2*



© Paul Viola 1999

Machine Learning

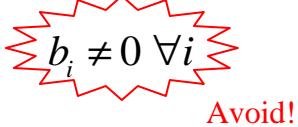
## *Review & Overview*

- Lecture 9:
  - » Resurrecting Perceptrons
  - » Setting up Support Vector Machines
- SVM review
- Why is it called "Support Vectors"??
- Derivation of some simpler properties.

© Paul Viola 1999

Machine Learning

## SVM: Key Ideas

- Augment inputs with a very large feature set  $\Phi(x)$ 
  - » Polynomials, etc.
- Use Kernel Trick(TM) to do this efficiently
- Enforce/Encourage Smoothness with weight penalty
  - » Minimize  $b^T b = \sum_i b_i^2$
- Find best solution using Quadratic Programming

© Paul Viola 1999

Machine Learning

## Support Vectors

$\min(w^T w)$  subject to constraint

$$\forall j \left[ \sum_i b_i K(x^j, c^i) \right] \geq 1$$

$$\Rightarrow y(x) = \Theta\left(\sum_i b_i K(x, c^i)\right)$$

- Many of the b's are zero -- inactive constraints
  - » Only keep examples where  $b_i \neq 0$
- Likely to generalize well
  - » VC Dimension -- later in the semester

© Paul Viola 1999

Machine Learning

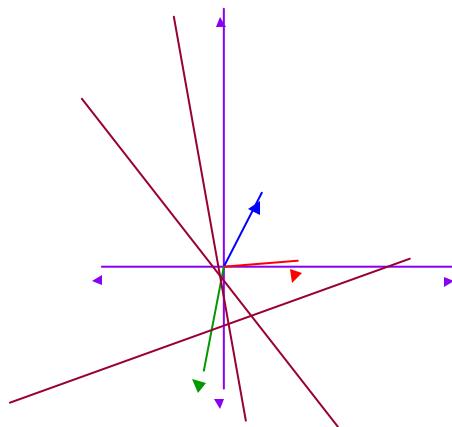
## *An alternative motivation*

- Like all good ideas, Support Vector Machines can be motivated in several different ways.

© Paul Viola 1999

Machine Learning

## *The optimal dividing line...*

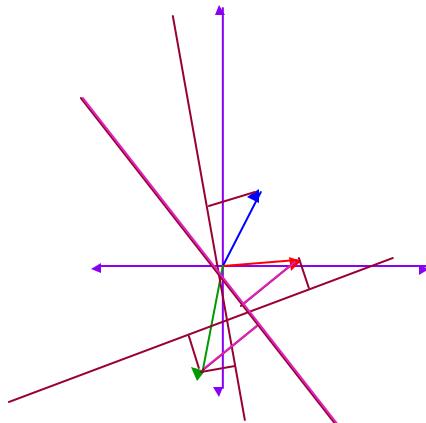


© Paul Viola 1999

Machine Learning

## *The optimal dividing line...*

- The optimal separator maximizes the **margin** between positive and negative examples



$$d_- = \max_{\text{negatives}} w^T x^i$$

$$d_+ = \min_{\text{positives}} w^T x^i$$

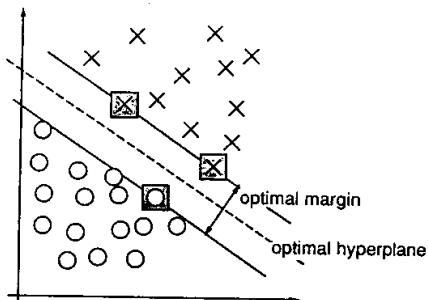
$$\text{margin} = \frac{d_+ - d_-}{|w|}$$

$$\max_w (\text{margin}) = \max_w \frac{d_+ - d_-}{|w|}$$

© Paul Viola 1999

MacL

## *Definition of the Margin*



- Margin: Gap between negatives and positives measured perpendicular to a hyperplane

© Paul Viola 1999

Machine Learning

## *Optimal dividing line=Support Vectors*

$$d_- = \max_{negatives} w^T x^i$$

$$d_+ = \min_{positives} w^T x^i$$

$$\max_w \frac{d_+ - d_-}{|w|}$$

$$\forall_{negatives} w^T x^i \leq -1$$

$$\forall_{positives} w^T x^i \geq 1$$

$$\min w^T w$$

© Paul Viola 1999

Machine Learning

## *Lemma 1: Weight vectors are simple*

$$w = \sum_l b_l \mathbf{x}^l$$

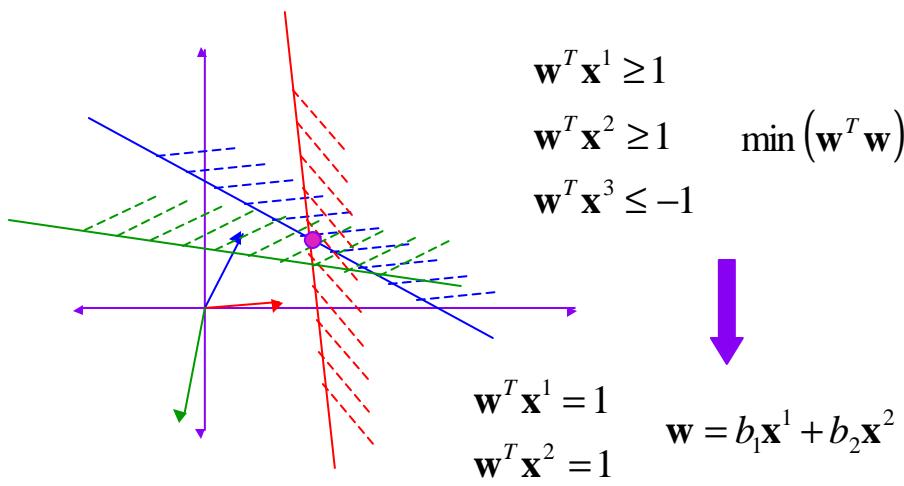
$$w = \sum_l b_l \Phi(\mathbf{x}^l)$$

- The weight vector lives in a sub-space spanned by the examples...
- Proved this to you by analyzing the perceptron weight update rule...
  - » But we no longer use that rule!!!
  - » Instead we use Quadratic Programming

© Paul Viola 1999

Machine Learning

## Lemma 1: Kuhn-Tucker Conditions



© Paul Viola 1999

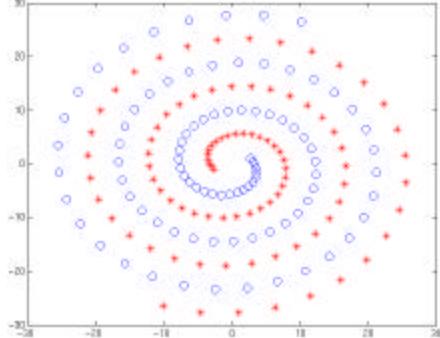
## SVM versus Perceptron

- Why not just use a perceptron?
  - » Use all training points as centers
$$y(x) = \Theta\left(\sum_i w_i^T K(x, c^i)\right)$$
  - » Update using perceptron rule:
$$w_i^t = w_i^t + h(t^t - y(x^t))K(x, c^i)$$
- Perceptrons do not maximize the margin
  - » The estimated function is not terribly smooth...
- Perceptrons do not rely on very few support vectors
  - » Yields a much more efficient classifier.

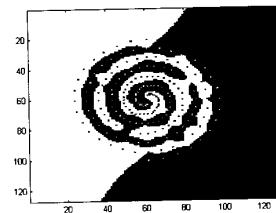
© Paul Viola 1999

Machine Learning

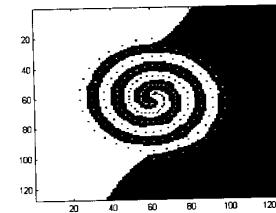
## *Perceptrons are not smooth...*



© Paul Viola 1999



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



Machine Learning

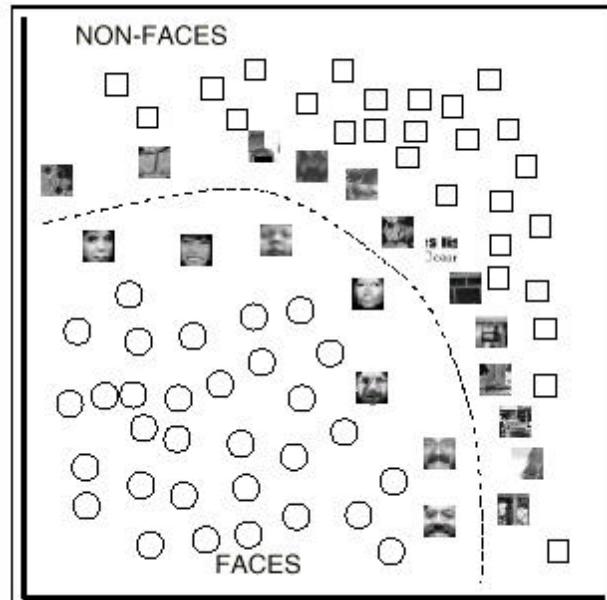
## *SVM: Faces*



|                    | Test Set A  |              | Test Set B  |              |
|--------------------|-------------|--------------|-------------|--------------|
|                    | Detect Rate | False Alarms | Detect Rate | False Alarms |
| SVM                | 97.1 %      | 4            | 74.2%       | 20           |
| Sung <i>et al.</i> | 94.6 %      | 2            | 74.2%       | 11           |

Machine Learning

## Support Vectors



## SVM: Difficulties

- How do you pick the kernels?
  - » Intuition / Luck / Magic / ...
- What if the data is not linearly separable?
  - » i.e. the constraints cannot be satisfied

$$\forall j \left[ \sum_i b_i K(x^j, c^i) \right] + s_j \geq 1$$

$$\min(b^T b + c \sum_j |s_j|) \quad \text{Slack Variables}$$

## *SVM: Generalization??*

- Is there a formal proof that SVM's will work better than Perceptrons or MLPs??
  - » Perhaps...
- There is a tenuous relationship between maximizing the margin and reducing the complexity of the classifier.
  - » The complexity of the classifier is reduced to the number of support vectors.
  - » Hard problems require more support vectors.
- The VC-Dimension of a support vector machine is controlled by maximizing the margin.

© Paul Viola 1999

Machine Learning

## *Margin is the Key Concept*

- As the margin is increased, so too does generalization.
- We will see other types of algorithms which will attempt to maximize the margin between positive and negative examples...

© Paul Viola 1999

Machine Learning

## Can we regain the simplicity of Perceptrons

KA algorithm without bias

1. Initialise  $\alpha_i^0 = 0$ .
2. For  $i = 1, m$  execute steps 3 and 4 below.
3. For labelled points  $(x_i, y_i)$  calculate:

$$z_i = \sum_{j=1}^m \alpha_j y_j K(x_i, x_j)$$

4. Calculate  $\delta\alpha_i^t = \eta(1 - z_i y_i)$ :
  - 4.1. If  $(\alpha_i^t + \delta\alpha_i^t) \leq 0$  then  $\alpha_i^t = 0$
  - 4.2. If  $(\alpha_i^t + \delta\alpha_i^t) > 0$  then  $\alpha_i^t \leftarrow \alpha_i^t + \delta\alpha_i^t$
5. If a maximum number of iterations is exceeded or the margin:

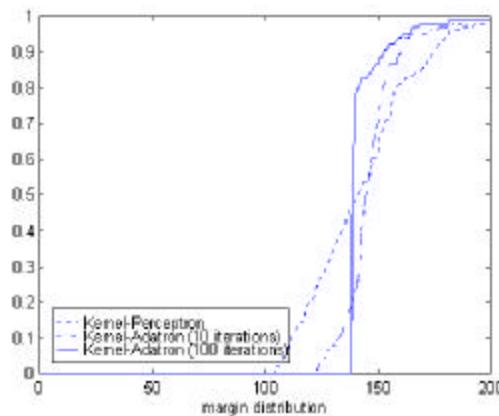
$$\gamma = \frac{1}{2} \left( \min_{\{i|y_i=-1\}} (z_i) - \max_{\{i|y_i=1\}} (z_i) \right)$$

is approximately 1 then **stop**, otherwise return to 2 for the next epoch  $t$ .

© Paul Viola 1999

Machine Learning

## How are the Margins effected??



© Paul Viola 1999

Machine Learning

# *6.891 Machine Learning and Neural Networks*

## Lecture 11: More Kernel Networks

© Paul Viola 1999

Machine Learning

## *News*

- Matlab was down at the AI lab for a few hours.
  - » I am not terribly sympathetic... since it was after the official deadline for the pset.
  - » Just hand it in as soon as you can.
- Cross-grading for next week.
  - » Please have it done by Thursday (earlier is better).

© Paul Viola 1999

Machine Learning

## Review & Overview

- Lecture 10:
  - » The Support in Support Vectors
  - » The Margin is a key concept
- The SVM criteria (one last time... )
- Smooth Regression
  - » Another way of motivating Kernel networks

© Paul Viola 1999

Machine Learning

## Optimal dividing line=Support Vectors

$$d_- = \max_{negatives} w^T x^i$$

$$d_+ = \min_{positives} w^T x^i$$

$$\max_w \frac{d_+ - d_-}{|w|}$$

$$\forall_{negatives} w^T x^i \leq -1$$

$$\forall_{positives} w^T x^i \geq 1$$

$$\min w^T w$$

© Paul Viola 1999

Machine Learning

## Optimal dividing line=Support Vectors

$$d_- = \max_{negatives} w^T x^i$$

$$d_- = \frac{-1}{|w|}$$

$$d_+ = \min_{positives} w^T x^i$$

$$d_+ = \frac{1}{|w|}$$

$$\max_w \frac{d_+ - d_-}{|w|}$$

$$\frac{d_+ - d_-}{|w|} = \frac{\frac{1}{|w|} - \frac{-1}{|w|}}{|w|} = \frac{1}{|w|^2} = \frac{1}{w^T w}$$

$$\forall_{negatives} w^T x^i \leq -1$$

$$\max \frac{1}{w^T w} \quad \min w^T w$$

© Paul Viola 1999

Machine Learning

## Kernel Networks are Good for Regression

$$y(x) = \Theta\left(\sum_i b_i K(x, c^i)\right) \quad y(x) = \sum_i b_i K(x, c^i)$$

- The form of the Kernel determines the form of the final function
  - » Polynomial Kernels → Polynomial Function
  - » Gaussian Kernels → Sum of Gaussians
- The Common Error Criteria is squared error...

$$Error = \sum_i (y(x^i) - t^i)^2$$

This ends up being exactly like polynomial fitting...  
except that there is one weight per data point

© Paul

## Radial Basis Function Networks

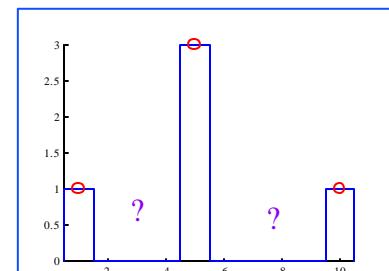
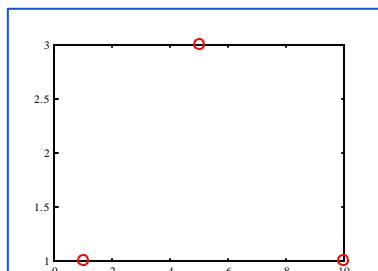
$$K(x, c) = K(|x - c|) \quad y(x) = \sum_i b_i K(|x - c^i|)$$

- When we restrict ourselves to Kernels which are radially symmetric, the resulting network is called a Radial Basis Function Network
  - »  $K$  only depends on the radial distance from some datapoint  $c$ .
  - » Poggio & Girosi pioneered the use of these.

© Paul Viola 1999

Machine Learning

## From Smoothness to Kernels: Assumptions are Necessary



Intuition

© Paul Viola 1999

Machine Learning

## Setting up the problem

$$Cost = (WY - T)^2$$

$$Y = (W^T W)^{-1} W^T T \quad \text{Not Invertible!}$$

$$T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

© Paul Viola 1999

Machine Learning

## Conditioning the Problem

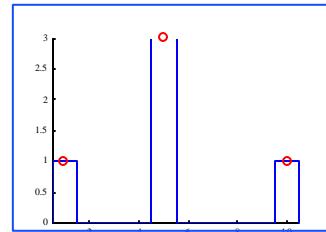
$$Cost = (WY - T)^2 + I Y^T Y \quad Y^T Y = \sum_i y_i^2$$

$$Y = (W^T W + I)^{-1} W^T T$$

Small solution  
vectors are best.

$$T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

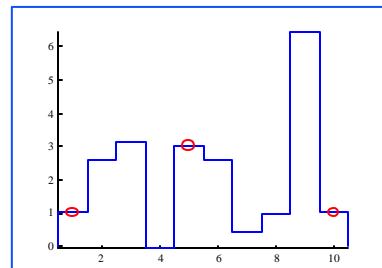
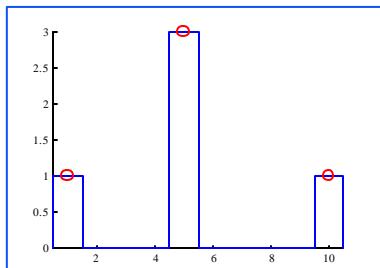
$$Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



© Paul Viola 1999

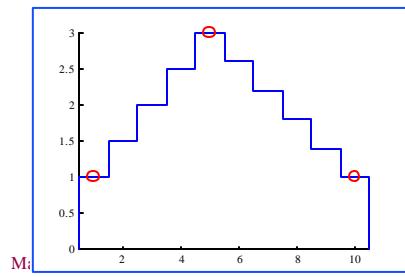
Machine Learning

*... and the winner is?*



This is not always true...  
remember to think like a  
Bayesian

© Paul Viola 1999



## *Smooth is Good: Regularization*

- Alternative way to motivate Kernel Networks.

$$Cost(y) = Error(y) + Smoothness(y)$$

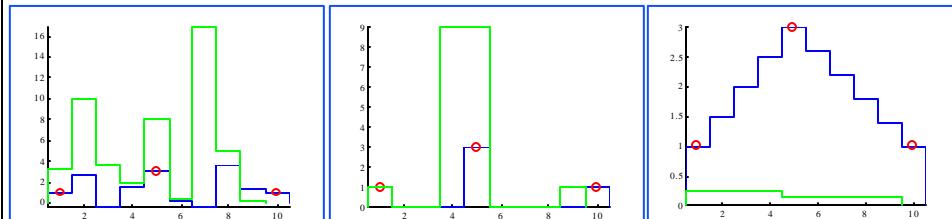
$$= \sum_j (y(x^j) - t^j)^2 + \int \left( \frac{\|y\|}{\|\hat{x}\|} \right)^2 d\hat{x}$$

© Paul Viola 1999

Machine Learning

## Derivative Measures Smoothness

### Squared 1st Derivative



Sum = 49.7

Sum = 20

Sum = 1.8

© Paul Viola 1999

Machine Learning

## Setting up the Problem

$$Cost = (WY - T)^2 + I(DY)^2$$

$$Y = (W^T W + I D^T D)^{-1} W^T T$$

$$T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1.0000 \\ 1.5000 \\ 2.0000 \\ 2.5000 \\ 3.0000 \\ 2.6000 \\ 2.2000 \\ 1.8000 \\ 1.4000 \\ 1.0000 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

© Paul Viola 1999

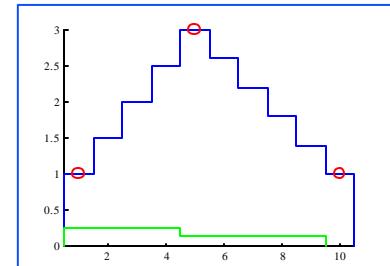
Machine Learning

*Need to find lambda ...*

$I = 0.001$

Y=

|        |
|--------|
| 1.0000 |
| 1.5000 |
| 2.0000 |
| 2.5000 |
| 3.0000 |
| 2.6000 |
| 2.2000 |
| 1.8000 |
| 1.4000 |
| 1.0000 |



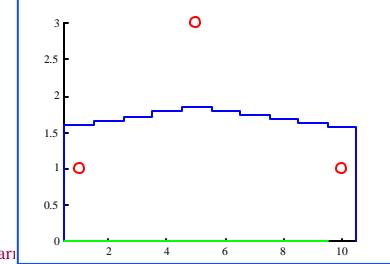
$I = 10$

© Paul Viola 1999

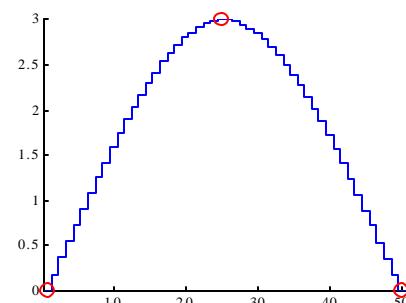
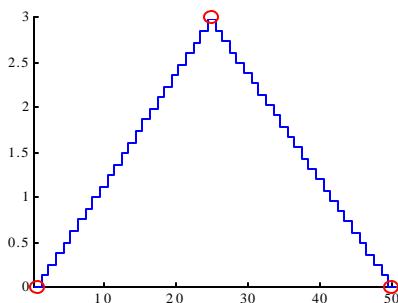
Y =

|        |
|--------|
| 1.6000 |
| 1.6600 |
| 1.7200 |
| 1.7800 |
| 1.8400 |
| 1.7840 |
| 1.7280 |
| 1.6720 |
| 1.6160 |
| 1.5600 |

Machine Lear



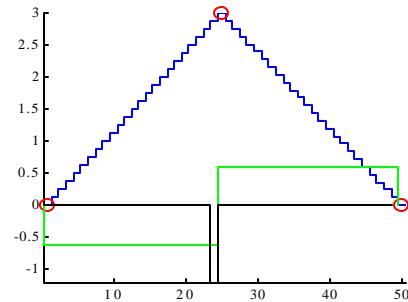
*Derivative Order Controls Shape*



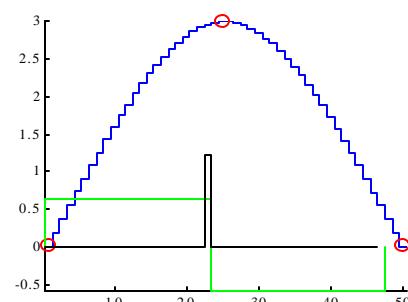
© Paul Viola 1999

Machine Learning

## A Closer Look



Linear + Kinks

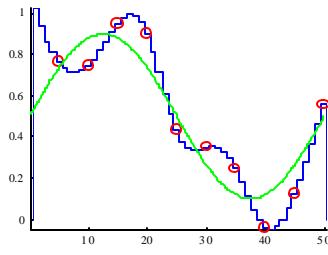
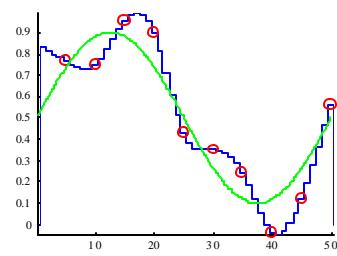
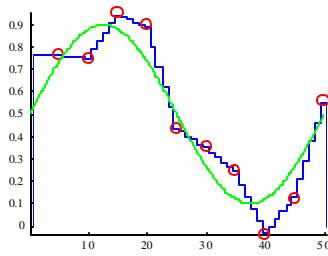


Cubics + Kinks

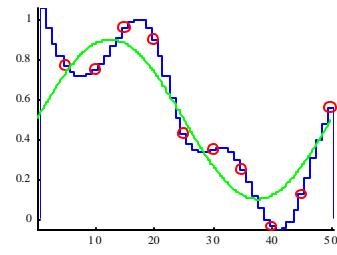
© Paul Viola 1999

Machine Learning

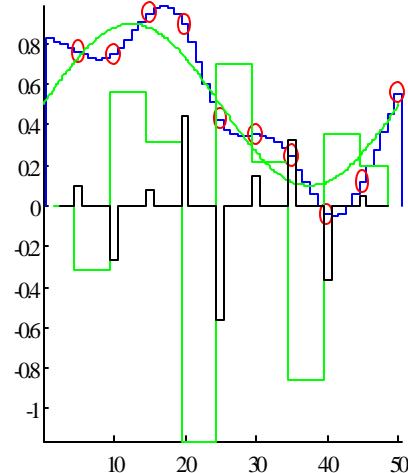
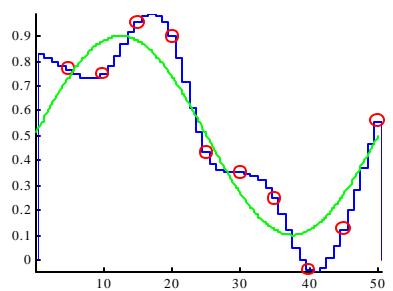
## Fitting More Data



hine Lea:



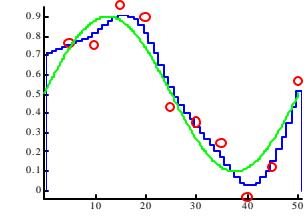
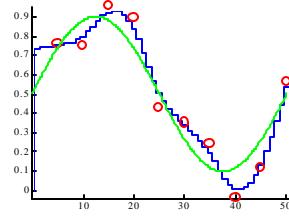
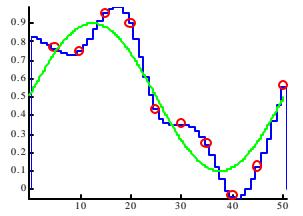
## *Still Piecewise Cubic*



© Paul Viola 1999

Machine Learning

## *Smoothness is easily controlled*



© Paul Viola 1999

Machine Learning

## Regularization to RBF's

- Alternative way to motivate RBF's

$$E(y) = \text{Error}(y) + \text{Smoothness}(y)$$

$$E = \frac{1}{2} \sum_n \{y(\mathbf{x}^n) - t^n\}^2 + \frac{\nu}{2} \int |Py|^2 d\mathbf{x}$$

$$y(\mathbf{x}) = \sum_n w_n G(\|\mathbf{x} - \mathbf{x}^n\|)$$

Every Training Point

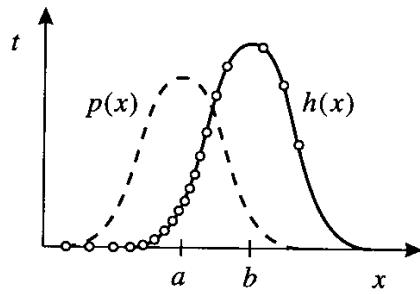
$$\int |Py|^2 d\mathbf{x} = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l!2^l} \int |D^l y(\mathbf{x})|^2 d\mathbf{x} \quad \rightarrow \quad \text{Gaussian}$$

© Paul Viola 1999

Machine Learning

## Problem: To many centers (Old solutions... )

- One per data point can be way to many...
- Choose a random subset of the points
  - Hope you don't get unlucky
- Distribute them based on the density of points
  - Perhaps EM clustering...



© Paul Viola 1999

## *Too Many Centers 2*

- Put them where you need them...
  - » To best approximate your function
- Compute the derivative of  $E(y)$  w.r.t. the centers
  - » This gets very hairy and does not work well
    - Too many local minima -- no small weight trick

© Paul Viola 1999

Machine Learning

## *Too Many Centers 3*

- Support Vector Regression... Next time.

© Paul Viola 1999

Machine Learning

# *6.891 Machine Learning and Neural Networks*

## Lecture 12: Smooth Functions and Kernel Networks

© Paul Viola 1999

Machine Learning

## *News*

- Quiz was too hard...
  - » I am trying to come up with a creative grading scheme.
    - Best 5 out of 6 problems???
    - First let us do the grading.
- Problem set will be out by tonight.

© Paul Viola 1999

Machine Learning

## Review & Overview

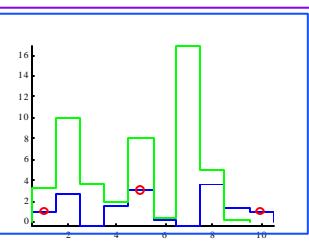
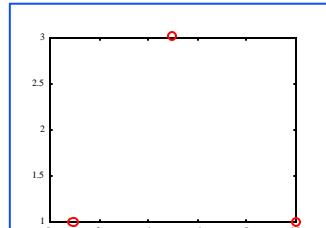
- Lecture 11:
  - » Trying to find smooth functions.
- Smooth Regression
  - » Another way of motivating Kernel networks

© Paul Viola 1999

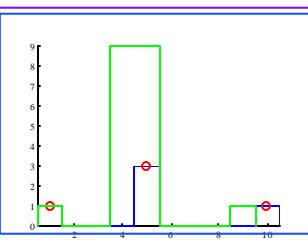
Machine Learning

... where we were last time.

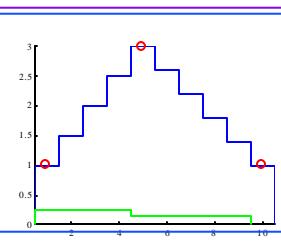
Squared 1st  
Derivative



Sum = 49.7



Sum = 20



Sum = 1.8

## Regression Review

- Up until now we have been mostly analyzing classification:
  - »  $X$ , inputs.  $Y$ , classes. Find the best  $c(x)$ .
- Today: Regression.
  - »  $X$ , inputs.  $Y$ , outputs. Find the best  $f(x)$ .
  - » Predict the stock's value next week.
  - » "Picture of Road" -> "Car steering wheel"
  - » etc.

$$\min_w \sum_j (f(x^j, w) - y^j)^2$$

© Paul Viola 1999

Machine Learning

## Schemes for motivating regression...

- Prior assumptions
  - » Find the best polynomial which fits the data:

$$f(x, w) = w_0 + w_1 x + w_2 x^2 + \dots \quad \min_w \sum_j (f(x^j, w) - y^j)^2$$

» Or find the best neural network, or ???

- Bayesian Approach
  - » Find the most likely function:

$$\max_f p(f | \{x^j, y^j\}) = \max_f \frac{p(\{x^j, y^j\} | f) p(f)}{p(\{x^j, y^j\})}$$

© Paul Viola 1999

Machine Learning

## Bayesian framework captures many approaches

$$\max_f p(f | \{x^j, y^j\}) = \max_f \frac{p(\{x^j, y^j\} | f) p(f)}{p(\{x^j, y^j\})}$$

$$\max_f [\log p(\{x^j, y^j\} | f) + \log p(f) - \log p(\{x^j, y^j\})]$$

$$\begin{aligned}\log p(\{x^j, y^j\} | f) &= \log \prod_j p(x^j, y^j | f) \\ &= \sum_j \log p(x^j, y^j | f) \\ &= \sum_j \log G(f(x^j) - y^j) \\ &= -\sum_j c(f(x^j) - y^j)^2\end{aligned}$$

$$p(f) = \begin{cases} e & \text{if } f \text{ is a poly} \\ 0 & \text{otherwise} \end{cases}$$

The polynomial that fits the data best is the most likely function

## Bayesian framework captures many approaches

$$\begin{aligned}\log p(\{x^j, y^j\} | f) \\ = -\sum_j c(f(x^j) - y^j)^2\end{aligned}$$

$$\log p(f) = - \int \left( \frac{\partial f}{\partial x} \right)^2$$

The function which both fits the data and has a small derivative is the most likely

Also popular...

$$\log p(f) = - \int \left( \frac{\partial^2 f}{\partial x^2} \right)^2$$

## *A closer look...*

$$C(f) = \sum_j c(f(x^j) - y^j)^2 + I \int \left( \frac{\partial f}{\partial x} \right)^2$$

$$X = \begin{bmatrix} 1 \\ 5 \\ 10 \end{bmatrix} \quad Y = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$$

Data

- How do we minimize this function?
  - » The set of possible functions is infinite
  - » The space of functions is infinite dimensional
- Constrain  $f$  to be: polynomial, sum of exponentials, etc??
- What about unconstrained solutions...

© Paul Viola 1999

Machine Learning

## *A slightly simpler problem*

$$C(f) = \sum_j c(f(x^j) - y^j)^2 + I \int \left( \frac{\partial f}{\partial x} \right)^2$$

$$\frac{dC(f)}{df} = 0$$

- But  $f$  is not a scalar!!
- In fact  $f$  is more like an infinite dimensional vector.
- If  $f$  were a finite vector:

$$\forall j \quad \frac{\partial C(f)}{\partial f_j} = 0$$

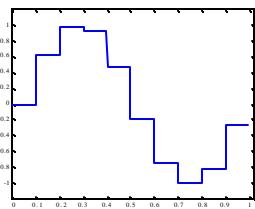
Can not reduce  $C()$  by adjusting any of the parameters of  $f$ .

© Paul Viola 1999

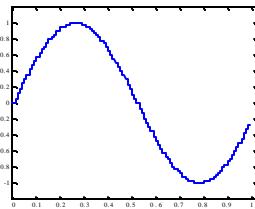
Machine Learning

*We could approximate  $f$ .*

10



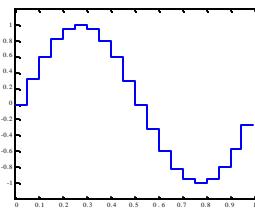
100



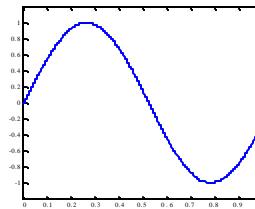
© Paul Viola 1999

Machine Learning

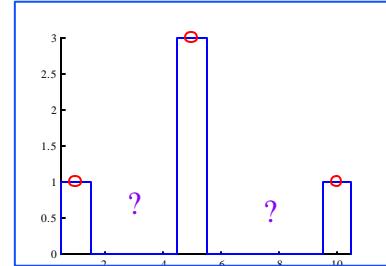
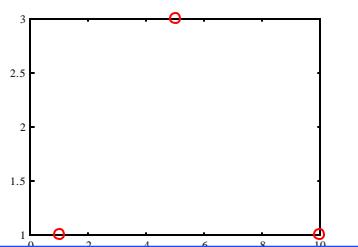
20



1000



*Looking for smooth solutions...*



Intuition

© Paul Viola 1999

Machine Learning

## Setting up the problem

$$Cost = (WF - Y)^2$$

$$F = (W^T W)^{-1} W^T Y \quad \text{Not Invertible!}$$

$$Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

=

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 \\ 22 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 6 \\ 0 \\ 1 \end{bmatrix}$$

© Paul Viola 1999

Machine Learning

## Conditioning the Problem

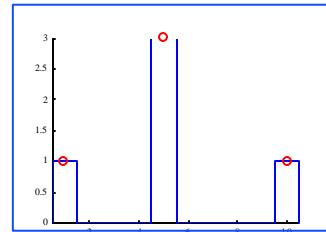
$$Cost = (WF - Y)^2 + I F^T F \quad F^T F = \sum_i f_i^2$$

$$F = (W^T W + I)^{-1} W^T Y$$

Small solution  
vectors are best.

$$Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

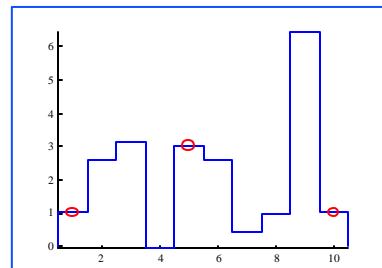
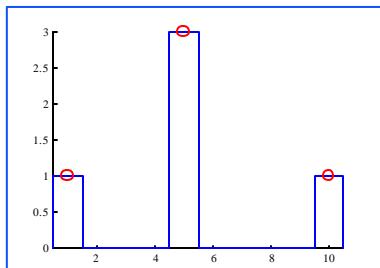
$$F = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



© Paul Viola 1999

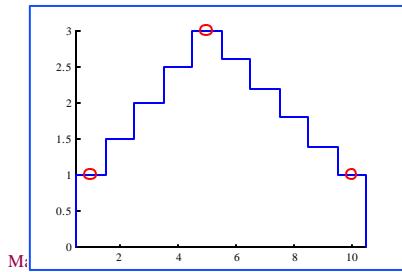
Machine Learning

*... and the winner is?*



This is not always true...  
remember to think like a  
Bayesian

© Paul Viola 1999



## *Smooth is Good: Regularization*

- Alternative way to motivate Kernel Networks.

$$Cost(f) = Error(f) + Smoothness(f)$$

$$= \sum_j (f(x^j) - y^j)^2 + \int \left( \frac{\|f\|}{\|\hat{x}\|} f(\hat{x}) \right)^2 d\hat{x}$$

© Paul Viola 1999

Machine Learning

## Setting up the Problem

$$Cost = (WF - T)^2 + I(DF)^2$$

$$F = (W^T W + I D^T D)^{-1} W^T Y$$

$$Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$F = \begin{bmatrix} 1.0000 \\ 1.5000 \\ 2.0000 \\ 2.5000 \\ 3.0000 \\ 2.6000 \\ 2.2000 \\ 1.8000 \\ 1.4000 \\ 1.0000 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

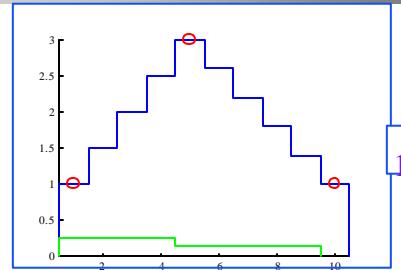
© Paul Viola 1999

Machine Lear

*Need to find lambda ...*

$$I = 0.001$$

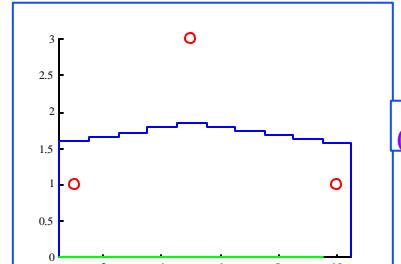
$$F = \begin{bmatrix} 1.0000 \\ 1.5000 \\ 2.0000 \\ 2.5000 \\ 3.0000 \\ 2.6000 \\ 2.2000 \\ 1.8000 \\ 1.4000 \\ 1.0000 \end{bmatrix}$$



1.8

$$I = 10$$

$$F = \begin{bmatrix} 1.6000 \\ 1.6600 \\ 1.7200 \\ 1.7800 \\ 1.8400 \\ 1.7840 \\ 1.7280 \\ 1.6720 \\ 1.6160 \\ 1.5600 \end{bmatrix}$$

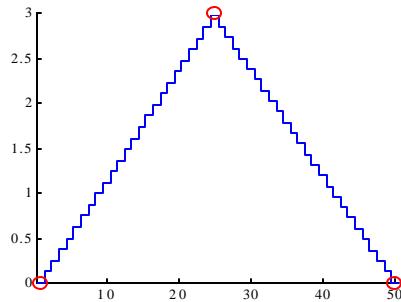


0.3

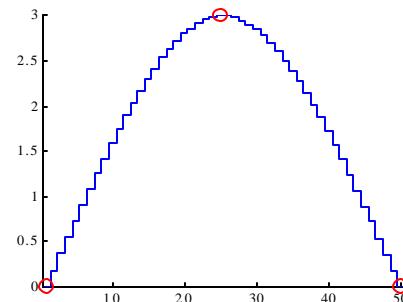
© Paul Viola 1999

Machine Lear

## Derivative Order Controls Shape



$$\int \left( \frac{\partial f}{\partial x} \right)^2$$

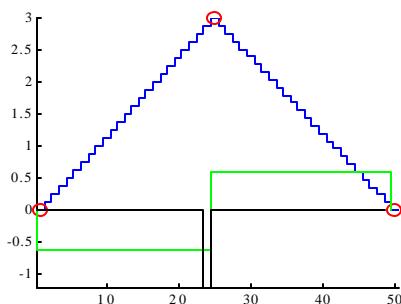


$$\int \left( \frac{\partial^2 f}{\partial x^2} \right)^2$$

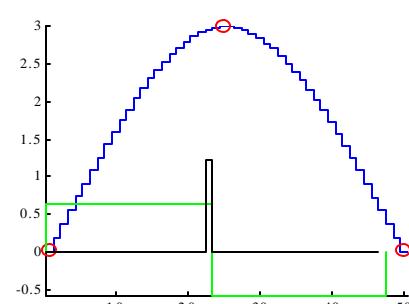
© Paul Viola 1999

Machine Learning

## A Closer Look



Linear + Kinks



Cubics + Kinks

© Paul Viola 1999

Machine Learning

*Look at the regularizer...*

$$Cost = (WF - T)^2 + I(DF)^2$$

$$F = (W^T W + I D^T D)^{-1} W^T Y$$

$$D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D' * D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

© Paul Viola 1999

*Second Deriv -> Fourth Deriv*

$$D = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$D' * D = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 5 \end{bmatrix}$$

© Paul Viola 1999

Machine Learning

## What about continuous functions??

$$C(f) = \mathbf{I} \int \left( \frac{\partial f}{\partial x} \right)^2$$

$$\forall x \quad \frac{\partial C(f)}{\partial f(x)} = 0$$

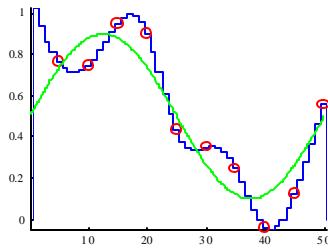
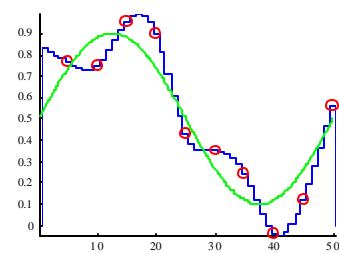
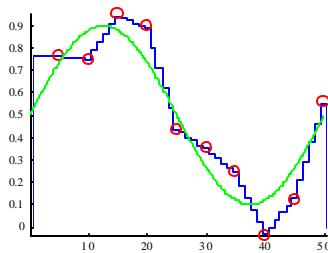
Infinite number  
of derivatives

$$\frac{\partial C(f)}{\partial f(x)} = \frac{C(f + \mathbf{d}_x) - C(f)}{|\mathbf{d}_x|} = \mathbf{d}C(x)$$

© Paul Viola 1999

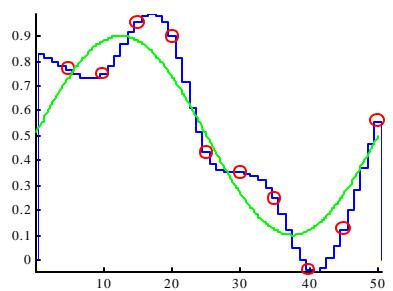
Machine Learning

## Fitting More Data



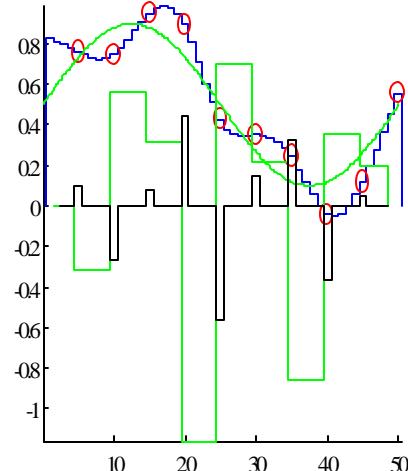
chine Lear...

## *Still Piecewise Cubic*

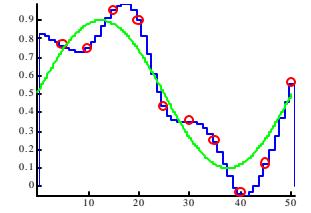


© Paul Viola 1999

Machine Learning

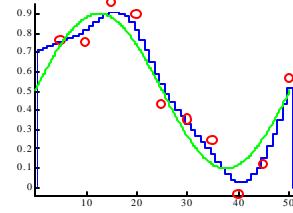
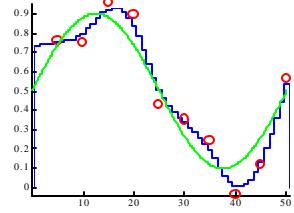


## *Smoothness is easily controlled*



© Paul Viola 1999

Machine Learning



## Regularization to RBF's

- Alternative way to motivate RBF's

$$E(y) = \text{Error}(y) + \text{Smoothness}(y)$$

$$E = \frac{1}{2} \sum_n \{y(\mathbf{x}^n) - t^n\}^2 + \frac{\nu}{2} \int |Py|^2 d\mathbf{x}$$

$$y(\mathbf{x}) = \sum_n w_n G(\|\mathbf{x} - \mathbf{x}^n\|)$$

Every Training Point

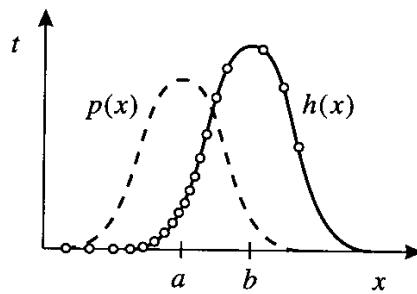
$$\int |Py|^2 d\mathbf{x} = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l!2^l} \int |D^l y(\mathbf{x})|^2 d\mathbf{x} \quad \rightarrow \quad \text{Gaussian}$$

© Paul Viola 1999

Machine Learning

## Problem: To many centers (Old solutions... )

- One per data point can be way to many...
- Choose a random subset of the points
  - Hope you don't get unlucky
- Distribute them based on the density of points
  - Perhaps EM clustering...



© Paul Viola 1999

## Too Many Centers 2

- Put them where you need them...
  - » To best approximate your function
- Compute the derivative of  $E(y)$  w.r.t. the centers
  - » This gets very hairy and does not work well
    - Too many local minima -- no small weight trick

© Paul Viola 1999

Machine Learning

## Too Many Centers 3

- Support Vector Regression...

$$Cost(f) = Error(f) + Smoothness(f)$$

$$= \sum_j |f(x^j) - y^j|_e + \int \left( \frac{d}{dx} f(\hat{x}) \right)^2 d\hat{x}$$

$$f(x) = \sum_j w_j K(x, x^j) \quad \text{Many j's are zero!!!}$$

© Paul Viola 1999

Machine Learning

# *6.891 Machine Learning and Neural Networks*

Lecture 13:  
Kernel Networks  
... on to Unsupervised Learning

© Paul Viola 1999

Machine Learning

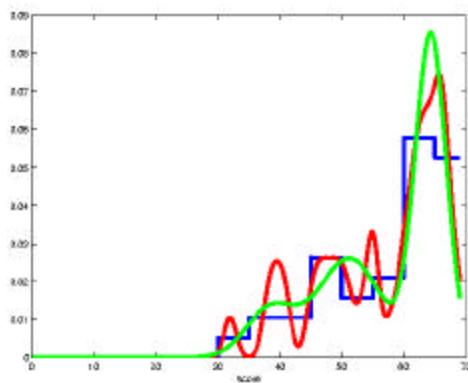
## *News*

- Quizzes are graded...
  - » Each problem has been graded.
  - » \*\* The overall score for the quiz is being determined.
    - We ran out of time last night.
- Course grading: (approximate)
  - » Psets: 35%
  - » Quiz: 20%
  - » Final: 30%
  - » Project: 10%
  - » Participation: 5%

© Paul Viola 1999

Machine Learning

## *Pset 3*

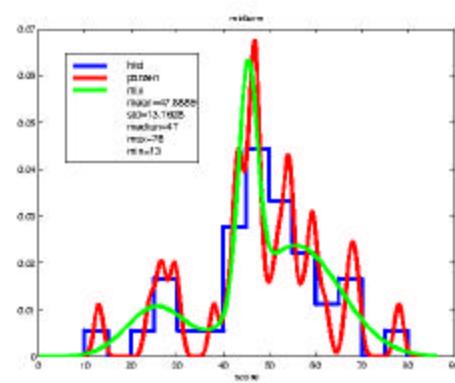


You are doing spectacularly well...

© Paul Viola 1999

Machine Learning

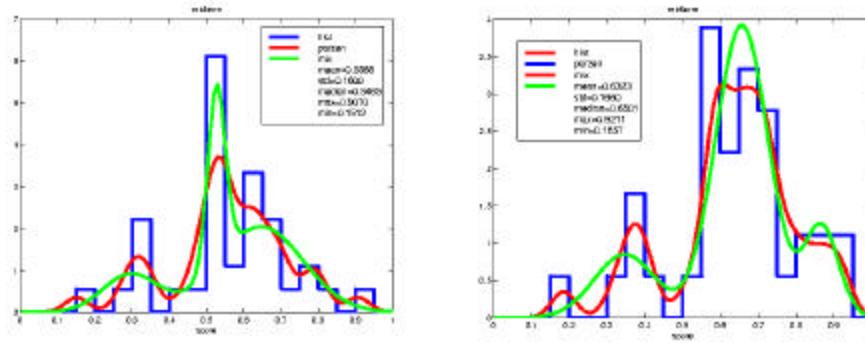
## *Exams*



© Paul Viola 1999

Machine Learning

## *Grading alternatives...*



© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 12:
  - » Trying to find smooth functions.
  - » Requiring smoothness simplifies functions:
    - 1st deriv  $\rightarrow$  piecewise linear; 2nd deriv  $\rightarrow$  cubic
- Finish off Regression
- Begin unsupervised learning.
  - » PCA, etc...

© Paul Viola 1999

Machine Learning

## Calculus of Variations

$$C(f) = \int \left( \frac{\partial f}{\partial x} \right)^2$$

$$\begin{aligned} dC(x) &= \frac{\partial C(f)}{\partial f(x)} = \frac{C(f + \mathbf{d}_x) - C(f)}{|\mathbf{d}_x|} \\ &= f''(x) \\ &= 0 \end{aligned}$$

$$f(x) = ax + b$$

$$C(f) = \sum_j (f(x^j) - y^j)^2 + I \int \left( \frac{\partial f}{\partial x} \right)^2$$

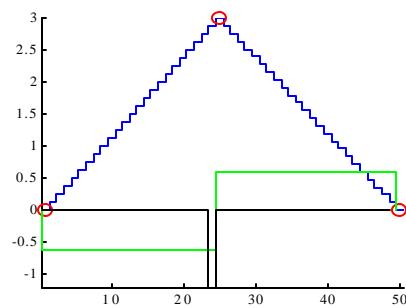
$$dC(x) = If''(x) + \sum_j 2(f(x^j) - y^j) \mathbf{d}(x - x^j) = 0$$

$$f''(x) = -\frac{1}{I} \sum_j 2(f(x^j) - y^j) \mathbf{d}(x - x^j)$$

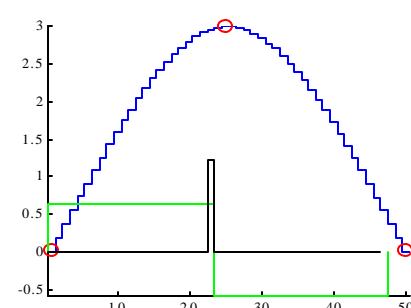
© Paul Viola 1999

Piecewise Linear

## A Closer Look



Linear + Kinks

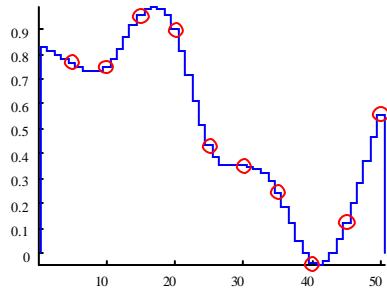


Cubics + Kinks

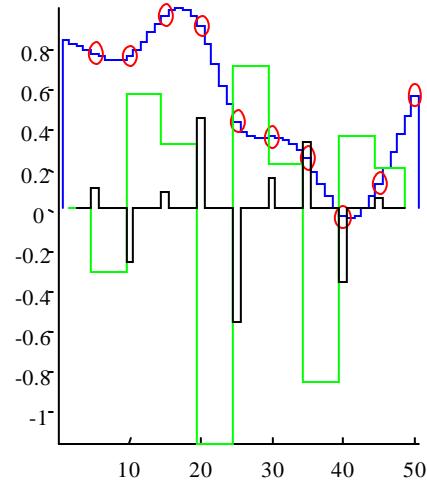
© Paul Viola 1999

Machine Learning

## Still Piecewise Cubic



© Paul Viola 1999



Machine Learning

## But where are the kernel functions??

- Recall that this was supposed to be another way to motivate kernel functions!!!

$$C(f) = \sum_j (f(x^j) - y^j)^2 + \lambda \int \left( \frac{\partial f}{\partial x} \right)^2 \quad \leftrightarrow \quad f(x) = \sum_j b_j K(x, x^j)$$

$$f''(x) = \sum_j a_j \mathbf{d}(x - x^j)$$

$$f''(x) = \sum_j b_j \frac{\partial^2 K(x, x^j)}{\partial x^2}$$

$$\frac{\partial^2 K(x, x^j)}{\partial x^2} = \mathbf{d}(x - x^j)$$

$$K(x, x^j) = |x - x^j|$$

© Paul Viola 1999

Machine Learning

*Cubics are similar...*

$$C(f) = \sum_j (f(x^j) - y^j)^2 + I \int \left( \frac{\nabla^2 f}{\|x\|^2} \right)^2$$



$$f(x) = \sum_j b_j K(x, x^j)$$

$$f''''(x) = \sum_j a_j \mathbf{d}(x - x^j)$$

$$f''''(x) = \sum_j b_j \frac{\partial^4 K(x, x^j)}{\partial x^4}$$

$$\frac{\partial^4 K(x, x^j)}{\partial x^4} = \mathbf{d}(x - x^j)$$



$$K(x, x^j) = |x - x^j|(x - x^j)^2$$

© Paul Viola 1999

Machine Learning

*Can also get gaussian kernels...*

- ... if you want them!!

$$E(y) = Error(y) + Smoothness(y)$$

$$E = \frac{1}{2} \sum_n \{y(\mathbf{x}^n) - t^n\}^2 + \frac{\nu}{2} \int |Py|^2 d\mathbf{x}$$

$$y(\mathbf{x}) = \sum_n w_n G(\|\mathbf{x} - \mathbf{x}^n\|)$$

Every Training Point

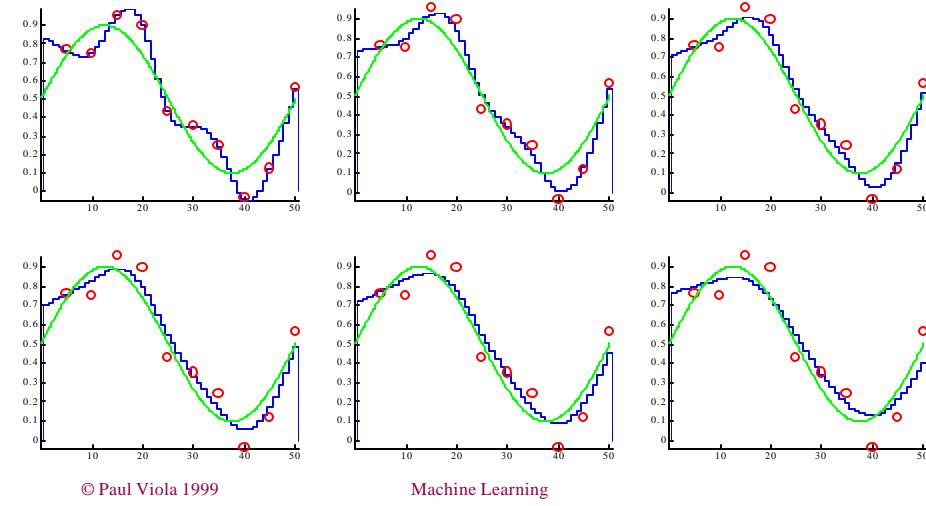
$$\int |Py|^2 d\mathbf{x} = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l! 2^l} \int |D^l y(\mathbf{x})|^2 d\mathbf{x}$$

Gaussian

© Paul Viola 1999

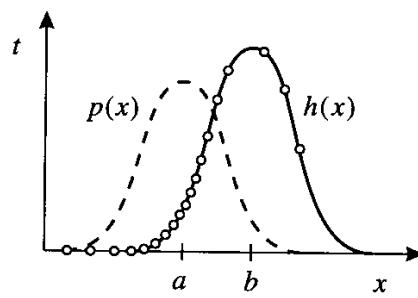
Machine Learning

## *Smoothness is easily controlled*



## *Problem: To many centers (Old solutions... )*

- One per data point can be way too many...
- Choose a random subset of the points
  - Hope you don't get unlucky
- Distribute them based on the density of points
  - Perhaps EM clustering...



© Paul Viola 1999

## Too Many Centers 2

- Put them where you need them...
  - » To best approximate your function
- Compute the derivative of  $C(f)$  w.r.t. the centers
  - » This gets very hairy and does not work well
    - Too many local minima -- no small weight trick

© Paul Viola 1999

Machine Learning

## Support Vector Regression

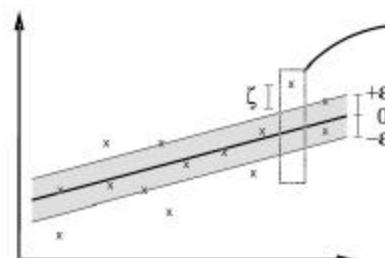
- Support Vector Regression...

$$f(x) = w^T x + b$$

$$(w^T x^j + b) - y^j \leq \epsilon$$

$$y^j - (w^T x^j + b) \leq \epsilon$$

$$\min w^T w$$

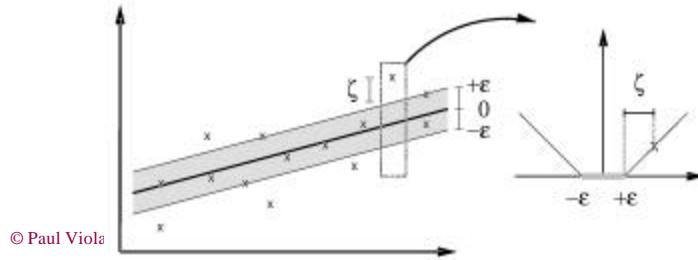


© Paul Viola 1999

## SVM Regression

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \quad |\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \end{aligned}$$

$$Cost(f) = c \sum_j \left| w^T x^j + b - y^j \right|_e + w^T w$$



*Works with smoothness as well...*

- Support Vector Regression...

$$Cost(f) = Error(f) + Smoothness(f)$$

$$= \sum_j \left| f(x^j) - y^j \right|_e + \int \left( \frac{I}{\|x\|} f(\hat{x}) \right)^2 d\hat{x}$$

$$f(x) = \sum_j w_j K(x, x^j) \quad \text{Many } w_j \text{'s are zero!!!}$$

## *New Topic: Unsupervised Learning*

- What can you do to “understand” data when you have no labels?
  - » Find unusual structure in the data.
  - » Find simplifications of the data.
- Find the clusters in the data:
  - » Fit a mixture of gaussians...
    - Been there, done that.
- Reduce the dimensionality of the data:
  - » Find a linear projection from high to low dimensions
- These all amount to density estimation
- There are many other approaches
  - » Build a tree which captures the data, etc.

© Paul Viola 1999

Machine Learning



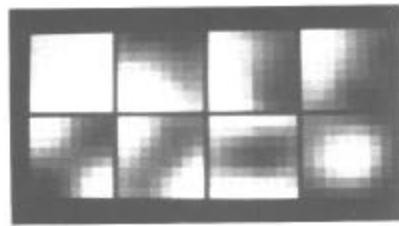
8 bits per pixel



.36 bits per pixel

© Paul Viola 1999

Machine Learning



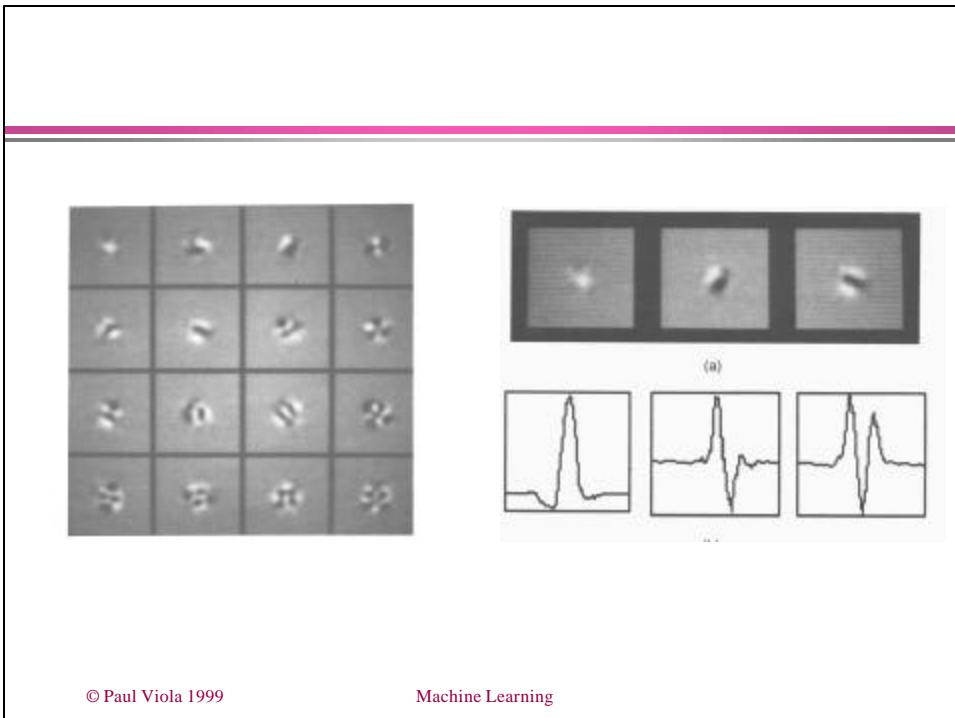
© Paul Viola 1999

Machine Learning



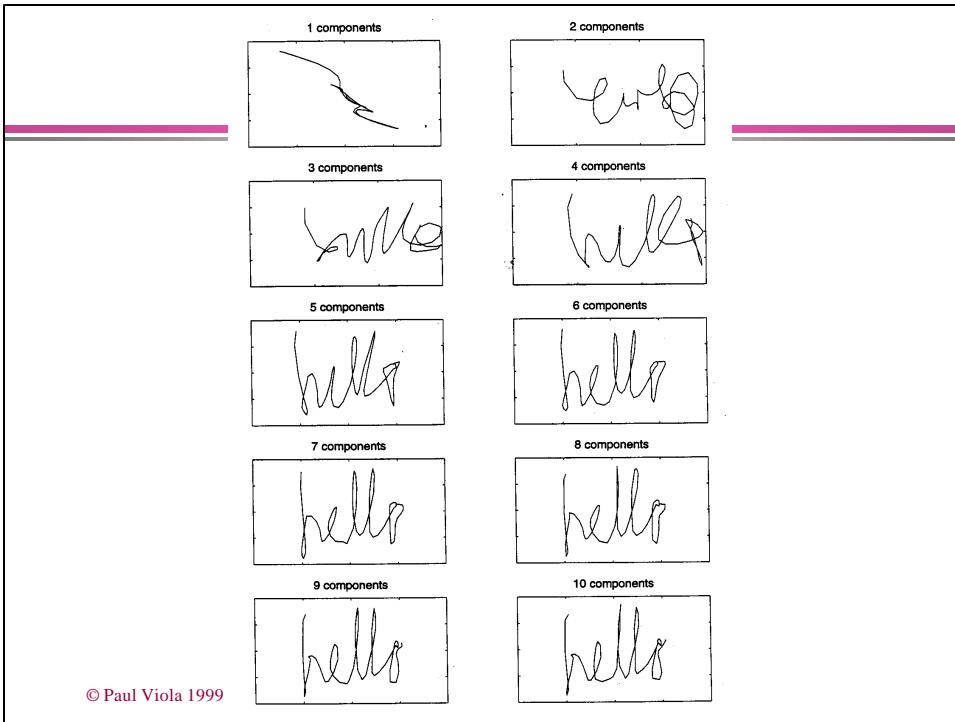
© Paul Viola 1999

Machine Learning



© Paul Viola 1999

Machine Learning



© Paul Viola 1999

# *6.891 Machine Learning and Neural Networks*

Lecture 14:  
... on to Unsupervised Learning

© Paul Viola 1999

Machine Learning

## *News*

- I will try to give you a feeling for where we are headed:
  - » Next 4 lectures
    - Bayes Nets / Graphical Models / Boltzman Machines / HMM's
  - » After that a series of topics (... from papers).

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 13:
  - » The end of regression...
- Begin unsupervised learning.
  - » PCA, etc...

© Paul Viola 1999

Machine Learning

## *New Topic: Unsupervised Learning*

- What can you do to “understand” data when you have no labels?
  - » Find unusual structure in the data.
  - » Find simplifications of the data.
- Find the clusters in the data:
  - » Fit a mixture of gaussians...
    - Been there, done that.
- Reduce the dimensionality of the data:
  - » Find a linear projection from high to low dimensions
- These all amount to density estimation
- There are many other approaches
  - » Build a tree which captures the data, etc.

© Paul Viola 1999

Machine Learning

## *Exploratory Data Analysis*

- Machine learning is simply not that smart...
- It is still very important to look at the data.
- But when there are millions of examples and thousands of dimensions you cannot look at the data.
  
- It is very important to summarize the data...
  - » Summarize the statistics
  - » Reduce dimensionality

© Paul Viola 1999

Machine Learning

## *Example: Mixture of Gaussians*

- Start out with a many training points (1000's)
  - » Distributed in a "clumpy" fashion.
- Replace with a few summary clusters (10's)
  - » One for each clump.
- Why?
  - » Better understand/summarize the data...
    - Demographics Data: growing number of stay at home dad's
      - Salary < 1000; Children > 0; Family Salary > 20,000, etc.
    - Astronomical Data: Unusual distribution of stars near galaxy
    - Extract "symbols"
      - one cluster per word (in speech), one cluster per letter (in writing)
  - » Speed learning...
    - Learn with the clusters rather than all the data... approximate

© Paul Viola 1999

Machine Learning

## *Example of Clustering*

- Can I get some examples of clustering???
- PDP??
- Andrew Moore

© Paul Viola 1999

Machine Learning

## *Speed Learning???*

- Regression & Kernel Networks 
$$f(x) = \sum_j b_j K(x, x^j)$$
  - » Learning time is cubic in the number of examples.
    - Need to solve the linear system to find weights.
  - » Performance time is linear in the number of examples
  - » 10,000 examples -> 100 clusters??
- Problem: these clusters are not optimal...
  - » Other locations may be much more effective for approximating the target function.
  - » Clusters congregate near data... not at the margin!

© Paul Viola 1999

Machine Learning

## *Support Vector Machines*

$$f(x) = \sum_j b_j K(x, x^j)$$

- Claims to choose the optimal set of examples
  - » 10,000 → 100
  - » Performance is optimal
- But, training time is still cubic in the number of examples.
- Though some new algorithms are beginning to work more quickly.

© Paul Viola 1999

Machine Learning

## *Example: Dimensionality Reduction*

- It is difficult to visualize 10, 20, 1000 dimensions
- Worse, there is reason to believe that it can be very difficult to learn in high dimensions.
- Curse of dimensionality:
  - » As the dimensionality of the data grows, the average distance between points also grows.
  - » As a result it is difficult to get an accurate local estimate for what is going on.

© Paul Viola 1999

Machine Learning

## *Curse of Dimensionality of Nearest Neighbor*

- How far is it to your nearest neighbor??
  - » Easier Question: How far do you have to look before expecting 1 neighbor.
  - » Assume points are distributed in a sphere of unit radius
$$\text{Vol}(S_k(r)) = c_k r^k$$
  - » The point in the center of the sphere should have the largest number of neighbors. How far do we have to go to find 1 neighbor on average??

© Paul Viola 1999

Machine Learning

## *Reducing Dimensionality Linearly*

$$y = Wx$$

- Where  $W$  has fewer rows than columns...
- What is the right  $W$ ??
  - » One that preserves as much information as possible.
  - » Ah, but what is information??

$$\min_W E(x - (W^{-*})Wx)^2$$

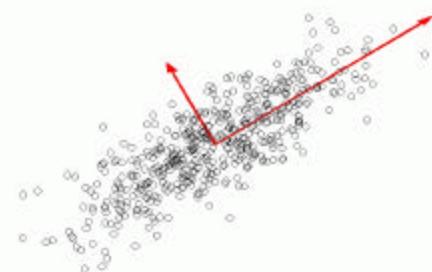


$$W = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \end{pmatrix}$$

© Paul Viola 1999

Machine Learning

*First Eigenvector preserves more info...*



PCA

© Paul Viola 1999

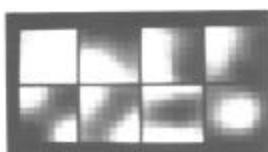
Machine Learning



256,000  
Numbers



4,000  
Numbers

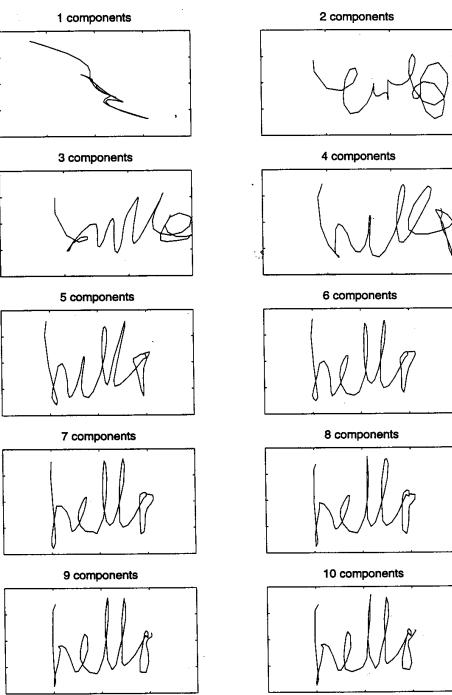


© Paul Viola 1999



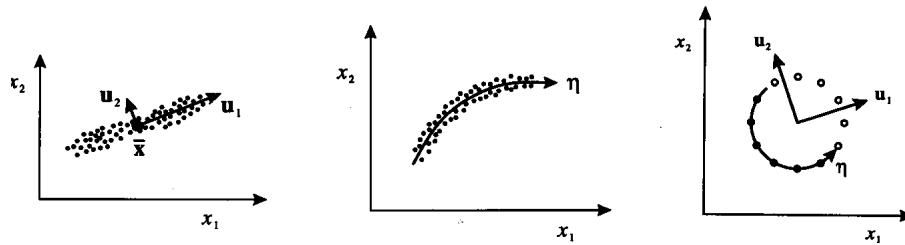
© Paul Viola 1999

Machine Learning



© Paul Viola 1999

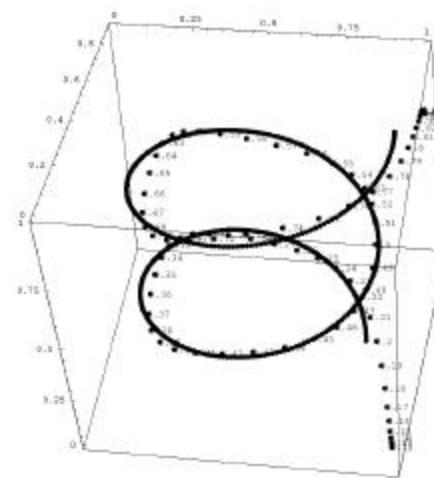
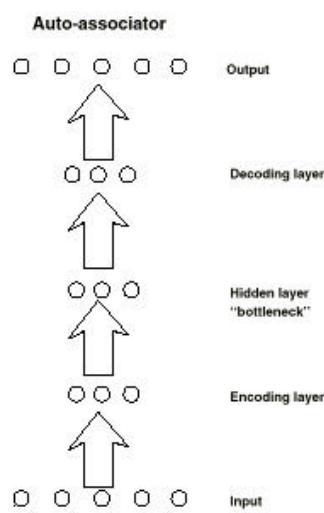
## Dimensionality Reduction



© Paul Viola 1999

Machine Learning

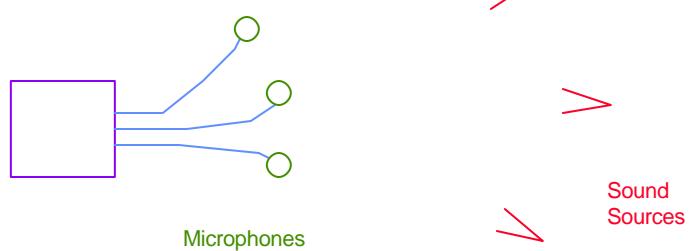
## Cottrell and Metcalfe



© Paul Viola 1999

Machine Learning

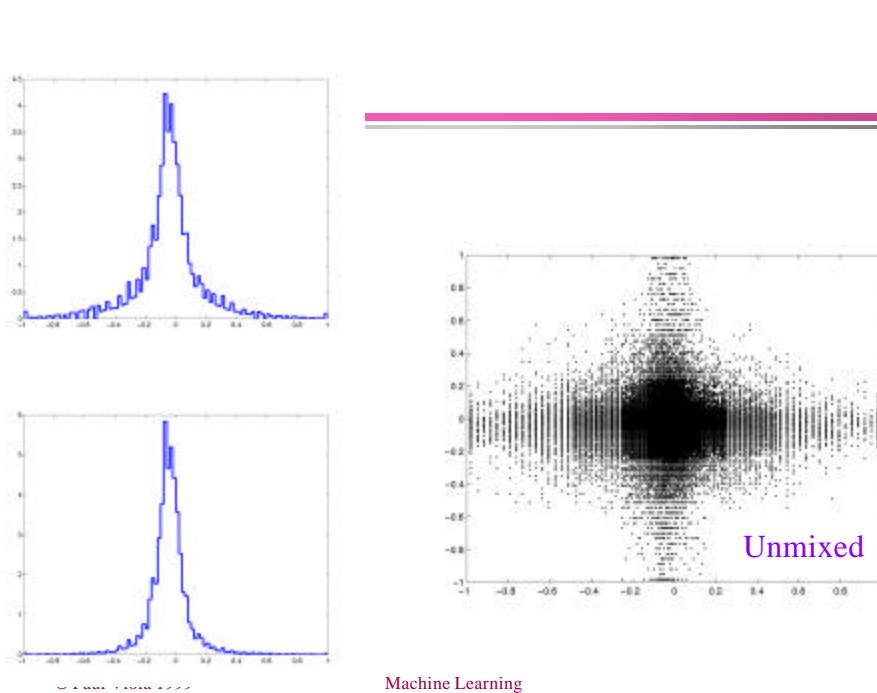
## *Information Theory for Signal Separation*

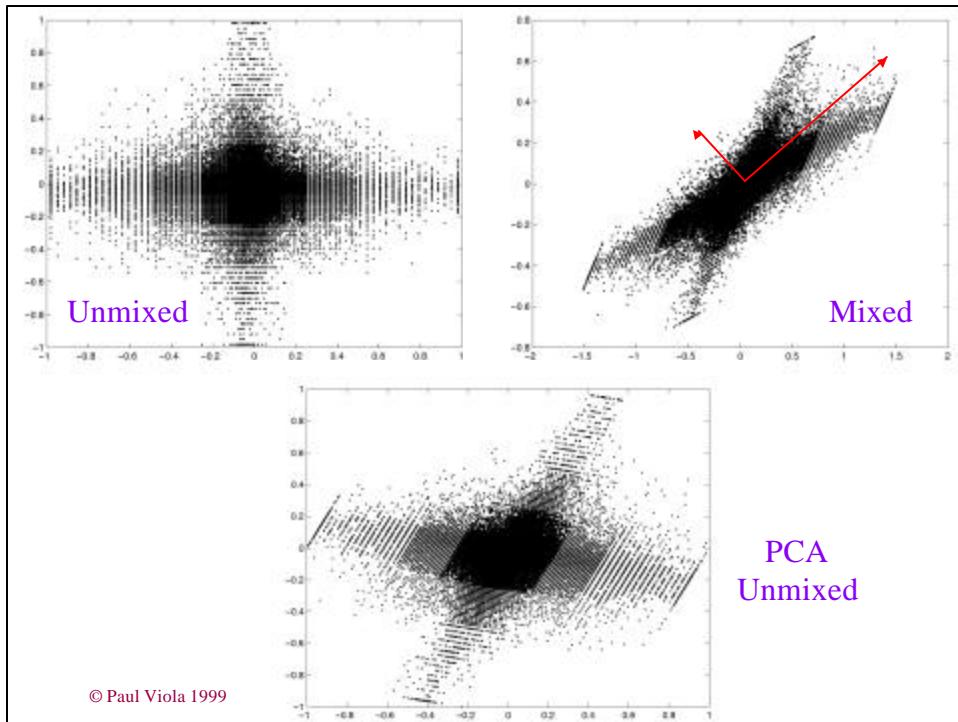


- The Cocktail Party Problem
- Many Speakers -- the signals a hopelessly mixed

© Paul Viola 1999

Machine Learning





## Mathematical Assumptions

$$S = \begin{bmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \end{bmatrix} \quad M = \begin{bmatrix} m_1(t) \\ m_2(t) \\ m_3(t) \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$M = AS$$

- Assumptions:
  - » Sounds Travels instantaneously
  - » Sound Mixes Linearly
  - » Signals are independent

## *The Unmixing Problem*

$$\hat{S} = A^{-1}AS$$

- We would like to undo the mixing...

© Paul Viola 1999

Machine Learning

## *Reducing Dimensionality Non-Linearly*

$$y = g(Wx)$$

- Where W has fewer rows than columns...
- What is the right W??
  - » One that preserves as much information as possible.
  - » Ah, but what is information??

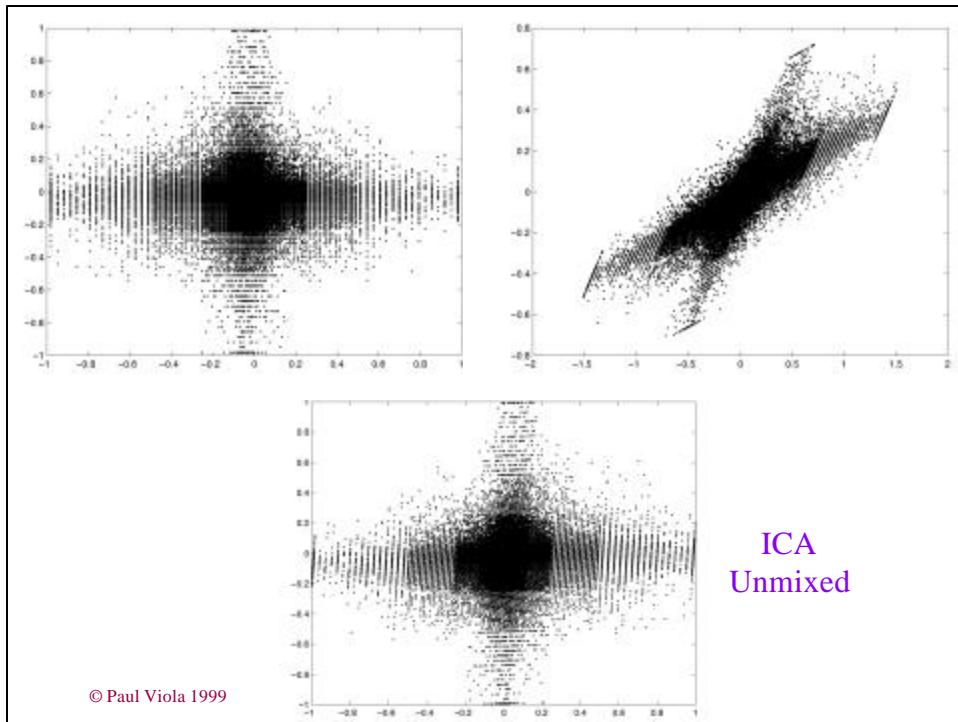
$$\max_W MI(g(Wx), x)$$



$$W = \begin{pmatrix} \text{independent} \\ \text{components} \end{pmatrix}$$

© Paul Viola 1999

Machine Learning



### *Learning Rule*

$$\Delta \mathbf{W} \propto [\mathbf{W}^T]^{-1} + (1 - 2y)\mathbf{x}^T$$

$$\begin{aligned}\Delta W &= (W^{-T} + (1 - 2y)x^T)W^TW \\ &= W + (1 - 2y)x^TW^TW \\ &= W + (1 - 2y)u^TW\end{aligned}$$

# *6.891 Machine Learning and Neural Networks*

Lecture 15:  
Reasoning and Learning on Discrete Data  
Bayes Nets

© Paul Viola 1999

Machine Learning

## *News*

- Final Problem Set will be ready tomorrow
  - » Mostly Bayes Nets
- Please begin to think about your final project

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 14:
  - » Principal Components Analysis
    - A low dimensional projection is can summarize data
  - » Independent Components Analysis
    - An alternative to PCA which can pick out the independent sources of data.
- Bayes Nets
  - » Meeting of the minds
    - Artificial Intelligence and Machine Learning
  - » Represents symbolic knowledge and reasoning
  - » Principled mechanism for inference and learning
    - Bayes Rule

© Paul Viola 1999

Machine Learning

## *Artificial Intelligence*

- Build systems that reason about the world:
  - » Diagnosis
    - "Why won't my car start?"
  - » Goal directed behavior
    - "How can I get from here to the White House?"
    - Space Probe: "How do I change orbit, take photos of Mars, and communicate with Earth in the next 5 minutes?"
    - "How can I symbolically integrate this function?"
  - » Game Playing
    - "How can I beat Kasparov?"
- Biases:
  - » Symbolic data and symbolic problems (not continuous)
  - » No representation of uncertainty or probability.

© Paul Viola 1999

Machine Learning

## *Techniques in Artificial Intelligence*

- Write down a set of rules that govern the world
  - » If I get on a plane to Wash. DC then I will end up in DC.
  - » If I take a taxi to Logan then I will end up at Logan.
  - » If my car is out of fuel then it won't start.
  - » If my starter motor is broken then it won't start.
  - » The integral of  $\sin(x)$  is  $-\cos(x)$ .
  - » The integral of  $2x$  is  $2x^2$
- Use these to either:
  - » Reason forward from the initial conditions
  - » Reason backwards from your goal (or problem).

© Paul Viola 1999

Machine Learning

## *Difficulty with Artificial Intelligence*

- No explicit representation for uncertainty
  - » Some unknown or unmodeled aspect of the world may interfere with your rules
    - Taking a plane to DC gets you to DC unless there is engine trouble, or the airport is fog bound, or the pilot gets ill, or someone on board has a heart attack, etc.
  - » Causality is probabilistic
    - The probability of vapor lock is higher in the summer.
- Early attempts to model uncertainty:
  - » Confidence factors
  - » Certainty factors
    - Not consistent with Probability...

© Paul Viola 1999

Machine Learning

## *Probabilistic Reasoning is Optimal*

- What we really want is to reason with the laws of probability:
  - » The probability that I will get to the White House is:
    - The probability of the conjunction of events
      - Get packed
      - Get to Logan
      - Catch plane
      - Arrive in DC
      - Get Taxi to White house
- Provided you can estimate and represent these probabilities!

© Paul Viola 1999

Machine Learning

## *Simple Example: Faculty Meetings*

- Faculty meetings can be argumentative sometimes.
  - » Hard to say why, but it happens about 23%
  - » When Marvin Minsky comes to the meeting
    - (Minsky only comes to 10% of the meetings)
    - Arguments happen 30%
  - » When Chomsky comes
    - (Chomsky only comes to 5% of the meetings)
    - Arguments happen 30%
  - » When Minsky and Chomsky come
    - Arguments happen 90%
- Very difficult for AI systems
  - » If Minsky than argument (false!!!)
  - » If Chomsky than argument (false!!!)

*The World  
is Fuzzy  
Logic is NOT!*

© Paul Viola 1999

Machine Learning

## A Probabilistic Approach

$$P(A, M, C)$$

Probability distribution over our 3 Events:  
Arguments, Minsky, and Chomsky

$$P(A = a, M = \bar{m}, C = c)$$

Probability of a joint event:  
Argument happens,  
Minsky does not come  
Chomsky does come

$$P(A) = \sum_{x,y} P(A, M = x, C = y) = \sum_{M,C} P(A, M, C)$$

$$P(A | M = m) = \frac{P(A, M = m)}{P(M = m)} = \frac{P(A, m)}{P(m)} = \frac{\sum_c P(A, m, C)}{\sum_{A,C} P(A, m, C)}$$

© Paul Viola 1999

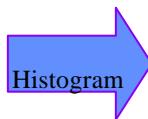
Machine Learning

## The Probabilistic Approach

- Everything is driven off  $P(A, M, C)$
- Given a set of example meetings

| A | M | C |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| ⋮ | ⋮ | ⋮ |

Observe Data



| M | C | A | Probability |
|---|---|---|-------------|
| 0 | 0 | 0 | 0.684       |
| 0 | 0 | 1 | 0.171       |
| 0 | 1 | 0 | 0.0315      |
| 0 | 1 | 1 | 0.036       |
| 1 | 0 | 0 | 0.0665      |
| 1 | 0 | 1 | 0.0285      |
| 1 | 1 | 0 | 0.00005     |
| 1 | 1 | 1 | 0.00045     |

Probability of Events

Everything should work out perfectly right?

© Paul Viola 1999

Machine Learning

## Problems with Naïve Probability

| M      | C | A | Probability |
|--------|---|---|-------------|
| 0      | 0 | 0 | 0.684       |
| 0      | 0 | 1 | 0.171       |
| 0      | 1 | 0 | 0.0315      |
| 0      | 1 | 1 | 0.036       |
| 1      | 0 | 0 | 0.0665      |
| 1      | 0 | 1 | 0.0285      |
| 1      | 1 | 0 | 0.00005     |
| 1      | 1 | 1 | 0.00045     |
| P(?) = |   |   | 0.23595     |

$$P(m) = 0.1$$

$$P(c) = 0.068$$

$$P(a) = 0.23$$

- Way too many variables:

»  $2^N$  variables (minus 1)

» Occam wouldn't like this

- Lots of computation:

»  $P(M)$  requires  $O(2^{(N-1)})$

»  $P(A|M)$  requires  $O(2^{(N-1)})$

- Optimality comes with serious penalties.

- Furthermore the table is very hard to interpret...

© Paul Viola 1999

Machine Learning

## Bayes Nets to the Rescue

- Assumptions can help a lot:

» We could assume that each property is independent

- Big mistake for most reasoning problems

» We could assume that certain variables are independent

- While others are dependent...

$$P(A, M, C) \equiv \underbrace{P(M)}_7 \underbrace{P(C | M)}_1 \underbrace{P(A | M, C)}_{\substack{2 \\ 4}} \quad \text{Tables}$$

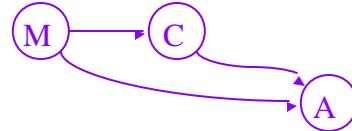
$$P(A, M, C) \approx \underbrace{P(M)}_7 \underbrace{P(C)}_1 \underbrace{P(A | M, C)}_{\substack{1 \\ 4}}$$

© Paul Viola 1999

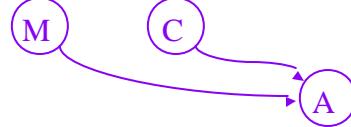
Machine Learning

## *Removing Links*

$$P(A, M, C) \equiv P(M) P(C | M) P(A | M, C)$$



$$P(A, M, C) \approx P(M) P(C) P(A | M, C)$$

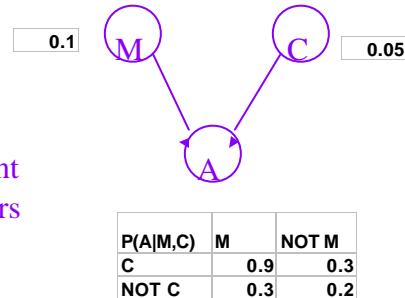


© Paul Viola 1999

Machine Learning

## *An Efficient Representation*

- Draw a directed acyclic graph
- Links imply causation
- Represent probabilities
  - Prior for every node with no parent
  - Conditional probabilities for others

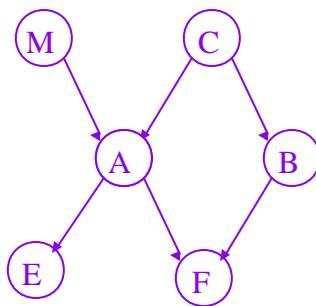


| $P(A M, C)$ | M   | NOT M |
|-------------|-----|-------|
| C           | 0.9 | 0.3   |
| NOT C       | 0.3 | 0.2   |

© Paul Viola 1999

Machine Learning

## *Much more efficient representations*



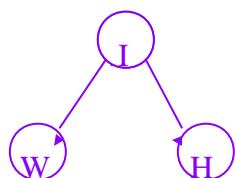
$$2^6 = 64 - 1 = 63 \text{ parameters}$$

$$\text{vs. } (2 * 1) + (2 * 2) + (2 * 4) = 13$$

© Paul Viola 1999

Machine Learning

## *Additional Example 1*



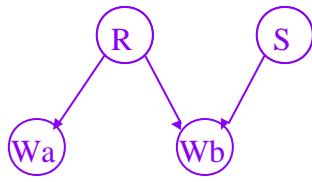
- You are waiting for an appointment with Holmes (H) and Watson (W).
- Both are very poor drivers and are likely to be avoid driving if the roads are icy (I).
- It is winter so that probability of I is high 0.5.
- H and W are dependent...
  - » Unless you know the road conditions

© Paul Viola 1999

Machine Learning

## *Additional Example 2*

### *Wet Grass*



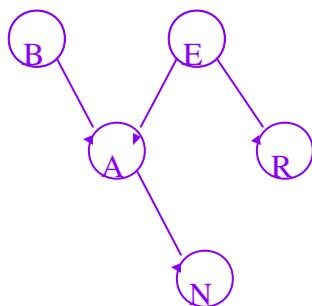
- The lawn of home A is either wet or not ( $W_a$ ).
  - » This could have been caused by rain (R) or sprinkler (S)
- The lawn of home B is either wet or not ( $W_b$ ).
  - » Home A has no sprinklers so rain is the only cause.

© Paul Viola 1999

Machine Learning

## *Additional Example 3*

### *Earthquake or Burglar*



- Home Alarm (A)
- Neighbor reports the Alarm (N)
- Burglary (B) 0.001
- Earthquake (E) 0.0000001
- Radio Report of Earthquake (R)
- You receive a call from your neighbor saying that your Alarm is going off.
- You drive home to confront the burglar... on the drive you hear a radio report of an Earthquake.

© Paul Viola 1999

Machine Learning

## Reasoning

- In some cases computation time is not changed:
  - » We have re-written the joint distribution
  - » Some reasoning still requires large summations...

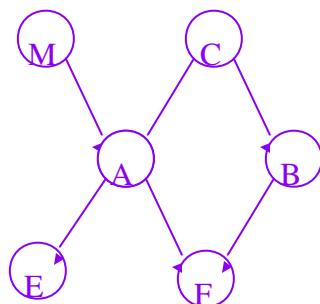
$$P(A) = \sum_{x,y} P(A, M = x, C = y) = \sum_{M,C} P(A, M, C)$$

$$P(A | M = m) = \frac{P(A, M = m)}{P(M = m)} = \frac{P(A, m)}{P(m)} = \frac{\sum_C P(A, m, C)}{\sum_{A,C} P(A, m, C)}$$

© Paul Viola 1999

Machine Learning

## Sometimes reasoning is efficient



$$P(e | a) = \frac{\sum_a P(a, B, C, D, e, F)}{\sum_a P(A, B, C, D, e, F)}$$

$$P(A, B, C, D, E, F) = P(M) P(C) P(A | M, C) P(B | C) P(E | A) P(F | A, B)$$

© Paul Viola 1999

Machine Learning

## Junction Tree Algorithm 1

- Table arithmetic:



$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$

$\forall a, b, c$

$$P(X = a, Y = b, Z = c) = P(X = a) P(Y = b | X = a) P(Z = c | Y = b)$$

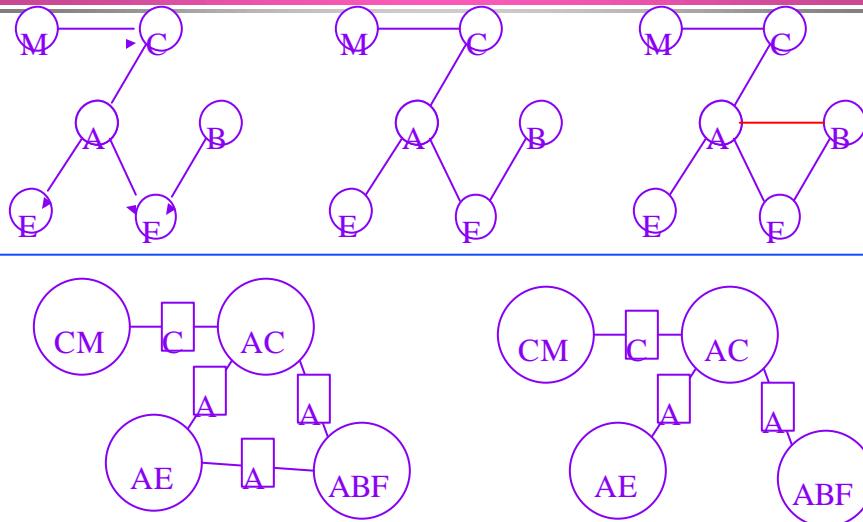
$$\begin{array}{c} \text{Z} \\ \text{X} \\ \text{Y} \end{array} = \begin{array}{c|c} \text{P}(X) & \\ \hline \text{X} & 0.4 \\ \text{NOT X} & 0.6 \end{array} * \begin{array}{c|cc} \text{P}(Y|X) & Y & \text{NOT Y} \\ \hline \text{X} & 0.9 & 0.1 \\ \text{NOT X} & 0.3 & 0.7 \end{array} * \begin{array}{c|cc} \text{P}(Z|Y) & Z & \text{NOT Z} \\ \hline \text{Y} & 0.2 & 0.8 \\ \text{NOT Y} & 0 & 1 \end{array}$$

$$T_{XYZ} = T_X \times T_{YX} \times T_{ZY}$$

© Paul Viola 1999

Machine Learning

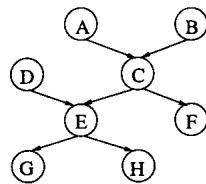
## Junction Tree Algorithm: Graph Hacking



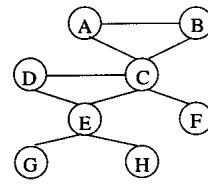
© Paul Viola 1999

Machine Learning

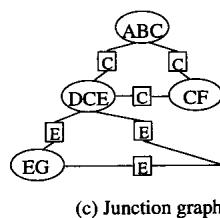
## More Junction Trees



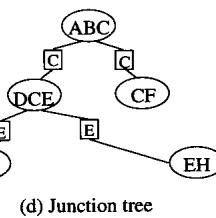
(a) DAG



(b) Moral graph



(c) Junction graph

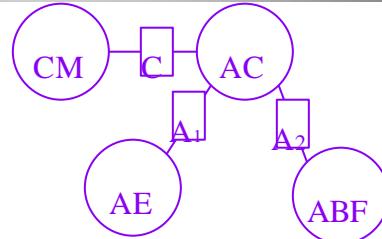
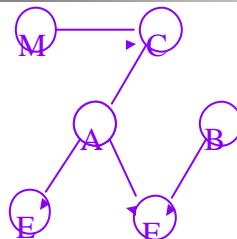


(d) Junction tree

© Paul Viola 1999

Machine Learning

## From Junction Trees to Probability



$$\begin{aligned}
 P(A, B, C, D, E, F) &= T_{ABCDEF} = \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \\
 &= P(M)P(C|M) \dots T_{CM} \\
 &\quad \times P(A|C) \dots T_{AC} \\
 &\quad \times P(E|A) \dots T_{AE} \\
 &\quad \times P(B)P(F|B) \dots T_{ABF}
 \end{aligned}$$

© Pau

# *6.891 Machine Learning and Neural Networks*

## Lecture 16: More Bayes Nets

© Paul Viola 1999

Machine Learning

## *News*

- Half of pset 5 is done
  - » and on the web.
  - » Other half will be done over the weekend.

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 15:
  - » Bayes Nets
    - Meeting of the minds
      - Artificial Intelligence and Machine Learning
    - Represents symbolic knowledge and reasoning
    - Principled mechanism for inference and learning
      - Bayes Rule
  - Reasoning with Bayes Nets
  - Efficient algorithms

© Paul Viola 1999

Machine Learning

## *Bayes Net Review*

- Knowledge of the complete probability table provides the opportunity for powerful deduction.
  - » Relate symptoms to diseases
    - Even when you have no observed every symptom
  - » Reason with partial and conflicting knowledge
- But the joint probability is difficult to model
  - »  $2^N$  -- N binary variables
  - » Reasoning is equally hard
- Given  $2^N$  numbers you can model any dependency
  - » But sometimes variables are not really dependent.

© Paul Viola 1999

Machine Learning

## Bayesian Text Classification

$\{d_k\}$  : A collection of documents

$$W_i(d_k) : \begin{cases} 1 & \text{if } d_k \text{ contains word i} \\ 0 & \text{otherwise} \end{cases}$$

$$P(F_1 = f_1, F_2 = f_2, \dots | C = c_j) \xleftarrow{\text{2^N probs}}$$

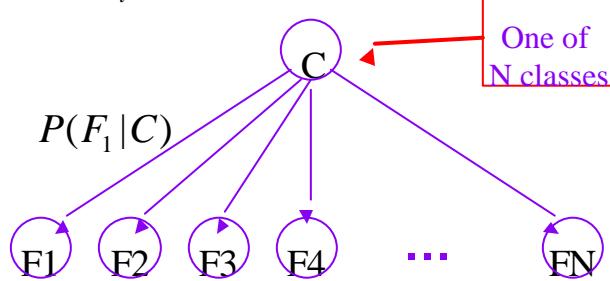
$$= P(\{f_1 - f_N\} | C = c_j)$$

$$\equiv \prod_i P(F_i = f_i | C = c_j) \xleftarrow{\text{Assume Independence}}$$

$$P(F_i = 1 | C = c_j) = p_{ij} \quad \text{Probability of word i appearing in a Doc from Class j}$$

## Bayes Nets Show Dependencies

$$P(\{f_i\} | C_j) = \prod_i P(F_i = f_i | C_j)$$

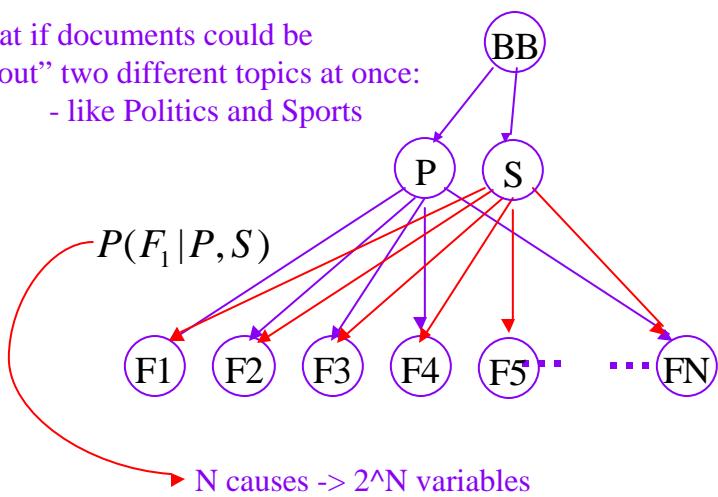


$$P(c_j | \{f_i\}) = \frac{P(c_j) \prod_i P(f_i | c_j)}{\prod_i P(f_i)}$$

© Paul Viola 1999

## *More Complex Models are “Easy”*

What if documents could be  
“about” two different topics at once:  
- like Politics and Sports

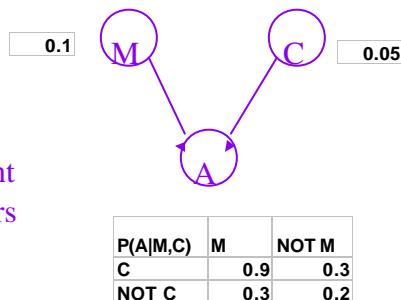


© Paul Viola 1999

Machine Learning

## *An Efficient Representation*

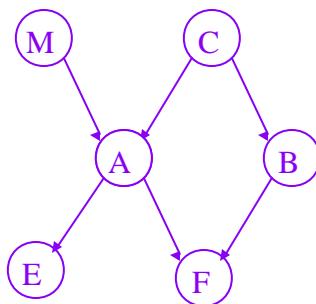
- Draw a directed acyclic graph
- Links imply causation
- Represent probabilities
  - Prior for every node with no parent
  - Conditional probabilities for others



© Paul Viola 1999

Machine Learning

## *Much more efficient representations*



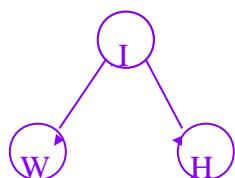
$$2^6 = 64 - 1 = 63 \text{ parameters}$$

$$\text{vs. } (2 * 1) + (2 * 2) + (2 * 4) = 13$$

© Paul Viola 1999

Machine Learning

## *Additional Example 1*



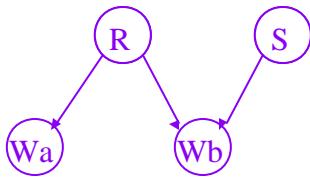
- You are waiting for an appointment with Holmes (H) and Watson (W).
- Both are very poor drivers and are likely to be avoid driving if the roads are icy (I).
- It is winter so that probability of I is high 0.5.
- H and W are dependent...
  - » Unless you know the road conditions

© Paul Viola 1999

Machine Learning

## *Additional Example 2*

### *Wet Grass*



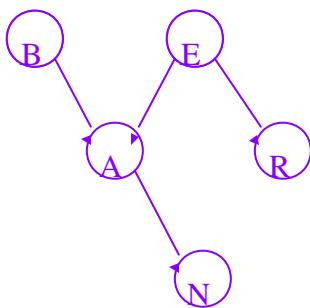
- The lawn of home A is either wet or not ( $W_a$ ).
  - » This could have been caused by rain (R) or sprinkler (S)
- The lawn of home B is either wet or not ( $W_b$ ).
  - » Home A has no sprinklers so rain is the only cause.

© Paul Viola 1999

Machine Learning

## *Additional Example 3*

### *Earthquake or Burglar*



- Home Alarm (A)
- Neighbor reports the Alarm (N)
- Burglary (B) 0.001
- Earthquake (E) 0.0000001
- Radio Report of Earthquake (R)
- You receive a call from your neighbor saying that your Alarm is going off.
- You drive home to confront the burglar... on the drive you hear a radio report of an Earthquake.

© Paul Viola 1999

Machine Learning

## Reasoning

- In some cases computation time is not changed:
  - » We have re-written the joint distribution
  - » Some reasoning still requires large summations...

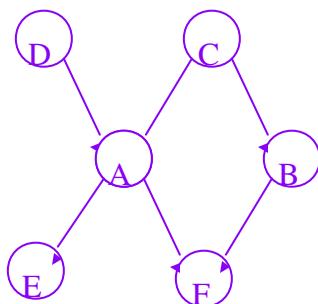
$$P(A) = \sum_{x,y} P(A, M = x, C = y) = \sum_{M,C} P(A, M, C)$$

$$P(A | M = m) = \frac{P(A, M = m)}{P(M = m)} = \frac{P(A, m)}{P(m)} = \frac{\sum_C P(A, m, C)}{\sum_{A,C} P(A, m, C)}$$

© Paul Viola 1999

Machine Learning

## Sometimes reasoning is efficient



$$P(e | a) = \frac{\sum_a P(a, B, C, D, e, F)}{\sum_a P(A, B, C, D, e, F)}$$

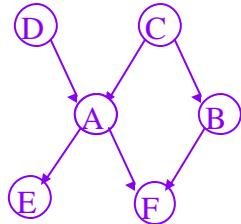
$$P(A, B, C, D, E, F) = P(D) P(C) P(A | D, C) P(B | C) P(E | A) P(F | A, B)$$

By the way: this is called adding the evidence (or observation)  
that  $A=a$ .

© Paul Viola 1999

Machine Learning

## Sometimes reasoning is more efficient



$$P(e | c) = \frac{\sum P(A, B, c, D, e, F)}{\sum P(A, B, c, D, E, F)}$$

Add the evidence that  $C=c$ .  
Observe the marginal of  $E$ .

$$\begin{aligned} & \sum_{a,b,d,e,f} P(A, B, c, D, E, F) \\ &= \sum_{a,b,d,e,f} P(D) P(c) P(A | D, c) P(B | c) P(E | A) P(F | A, B) \\ &= P(c) \sum_{a,e} P(E | A) \sum_d P(D) P(A | D, c) \sum_b P(B | c) \sum_f P(F | A, B) \end{aligned}$$

© Paul Viola 1999

Machine Learning

## Saving Work

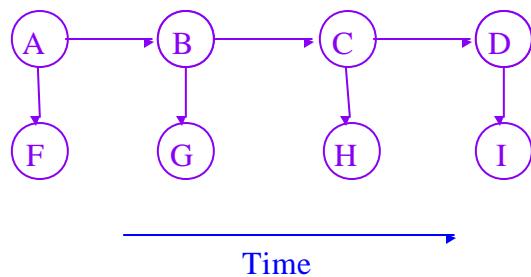
$$\begin{aligned} & \sum_{a,b,d,e,f} P(A, B, c, D, E, F) \\ &= P(c) \sum_{a,e} P(E | A) \sum_d P(D) P(A | D, c) \sum_b P(B | c) \sum_f P(F | A, B) \\ &= P(c) \sum_{a,e} P(E | A) \sum_d P(D) P(A | D, c) \sum_b T_B^2 T_{AB}^1 \\ &= P(c) \sum_{a,e} P(E | A) \sum_d T_D^5 T_{AD}^4 T_A^3 \\ &= P(c) \sum_{a,e} P(E | A) T_A^3 \sum_d T_D^5 T_{AD}^4 \\ &= P(c) \sum_{a,e} T_{AE}^7 T_A^3 T_A^6 \\ &= P(c) \sum_e T_e^8 \end{aligned}$$

$$\sum_{a,b,d,f} P(A, B, c, D, e, F) = P(c) T_E^8(e)$$

© Paul Viola 1999

Machine Learning

## Hidden Markov Model



$$\begin{aligned}
 P(A, B, C, D, F, G, H, I) \\
 = P(A) P(F | A) P(B | A) P(G | B) P(C | B) P(H | C) P(D | C) P(I | D)
 \end{aligned}$$

© Paul Viola 1999

Machine Learning

## Junction Tree Algorithm 1

- Table arithmetic:



$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$

$\forall a, b, c$

$$P(X = a, Y = b, Z = c) = P(X = a) P(Y = b | X = a) P(Z = c | Y = b)$$

|        | $=$ | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(X)</th> <th></th> </tr> <tr> <td>X</td> <td>0.4</td> </tr> <tr> <td>NOT X</td> <td>0.6</td> </tr> </table> $\ast$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(Y X)</th> <th>Y</th> <th>NOT Y</th> </tr> <tr> <td>X</td> <td>0.9</td> <td>0.1</td> </tr> <tr> <td>NOT X</td> <td>0.3</td> <td>0.7</td> </tr> </table> $\ast$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(Z Y)</th> <th>Z</th> <th>NOT Z</th> </tr> <tr> <td>Y</td> <td>0.2</td> <td>0.8</td> </tr> <tr> <td>NOT Y</td> <td>0</td> <td>1</td> </tr> </table> | P(X) |  | X | 0.4 | NOT X | 0.6 | P(Y X) | Y | NOT Y | X | 0.9 | 0.1 | NOT X | 0.3 | 0.7 | P(Z Y) | Z | NOT Z | Y | 0.2 | 0.8 | NOT Y | 0 | 1 |
|--------|-----|---|------|--|---|-----|-------|-----|--------|---|-------|---|-----|-----|-------|-----|-----|--------|---|-------|---|-----|-----|-------|---|---|
| P(X)   |     |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| X      | 0.4 |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT X  | 0.6 |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| P(Y X) | Y   | NOT Y   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| X      | 0.9 | 0.1   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT X  | 0.3 | 0.7   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| P(Z Y) | Z   | NOT Z   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| Y      | 0.2 | 0.8   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT Y  | 0   | 1   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |

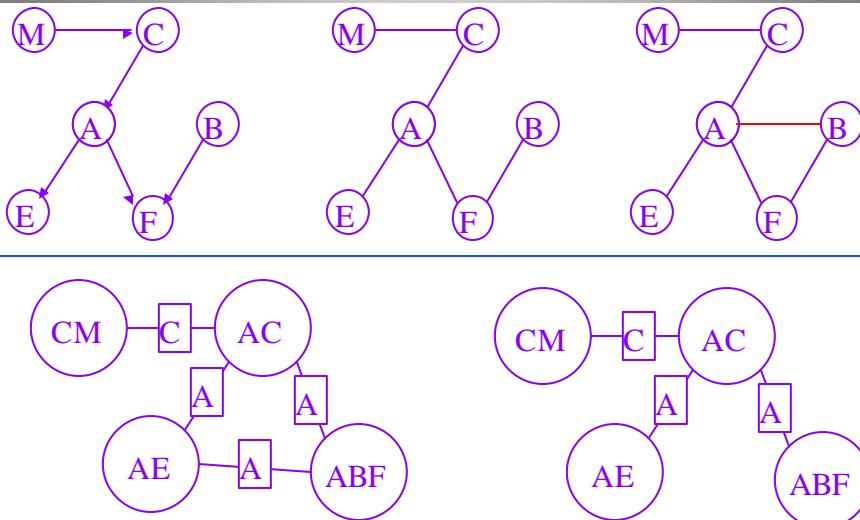
  

|           |     |                                   |
|-----------|-----|-----------------------------------|
| $T_{XYZ}$ | $=$ | $T_X \times T_{YX} \times T_{ZY}$ |
|-----------|-----|-----------------------------------|

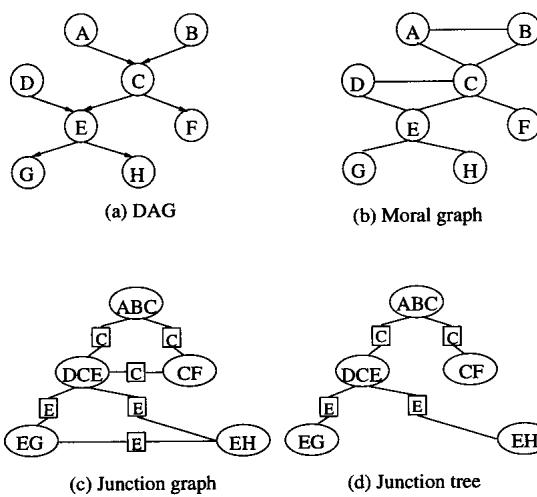
© Paul Viola 1999

Machine Learning

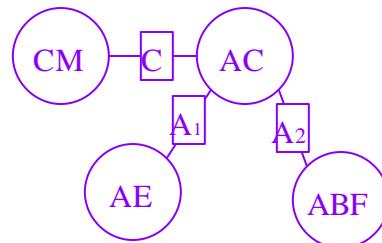
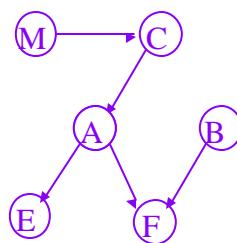
## Junction Tree Algorithm: Graph Hacking



## More Junction Trees



## From Junction Trees to Probability



$$\begin{aligned}
 P(A, B, C, D, E, F) &= T_{ABCDEF} = \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \\
 &= P(M)P(C | M) \quad \dots T_{CM} \\
 &\quad \times P(A | C) \quad \dots T_{AC} \\
 &\quad \times P(E | A) \quad \dots T_{AE} \\
 &\quad \times P(B)P(F | B) \quad \dots T_{ABF}
 \end{aligned}$$

© Pau

# *6.891 Machine Learning and Neural Networks*

## Lecture 17: Hidden Markov Models & Other Bayes Nets

© Paul Viola 1999

Machine Learning

## *News*

- Problem Set 5 complete on Monday
- Remember to keep thinking about your final projects!
  - » Please send us some email which describes your project
  - » 1-2 paragraphs
  - » We will render our “expert” opinion
    - Not too hard!!

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lecture 16:

- » Bayes Nets

- An Efficient way to represent joint probability distributions
    - Allow reasoning about subtle and conflicting evidence
    - Allow reasoning with partial information

- » Structure implies Reasoning Efficiency

- Dependence structure allows for more efficient reasoning
    - Dynamic programming

- Markov Processes

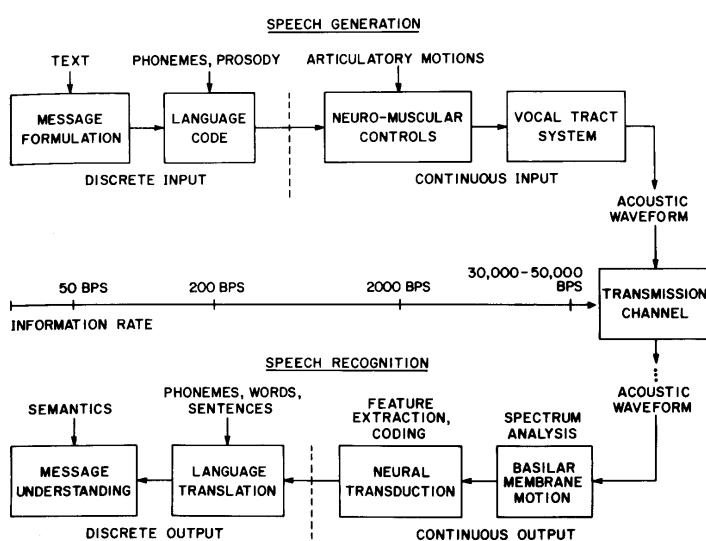
- Hidden Markov Models

- » Speech

© Paul Viola 1999

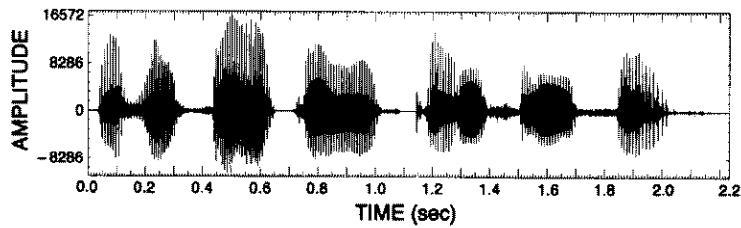
Machine Learning

## *A brief overview of speech recognition*



© I

## *A brief overview of speech recognition*

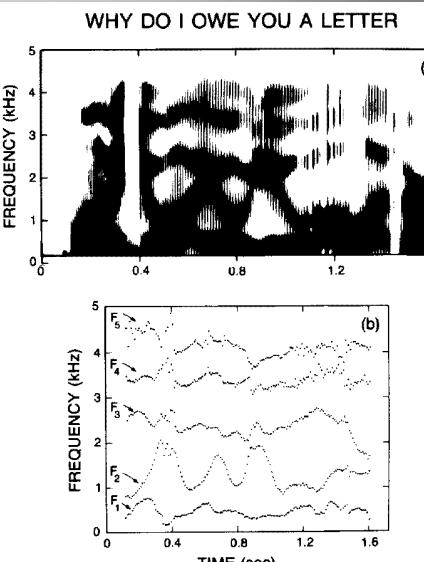
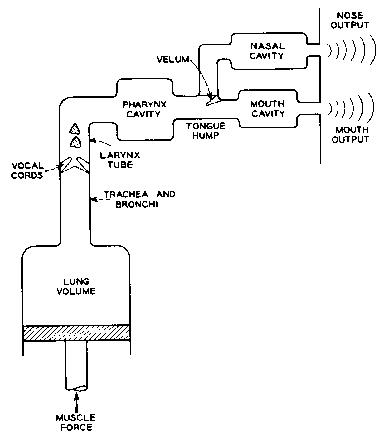


The signal is very high dimensional...  
10000 samples / second

© Paul Viola 1999

Machine Learning

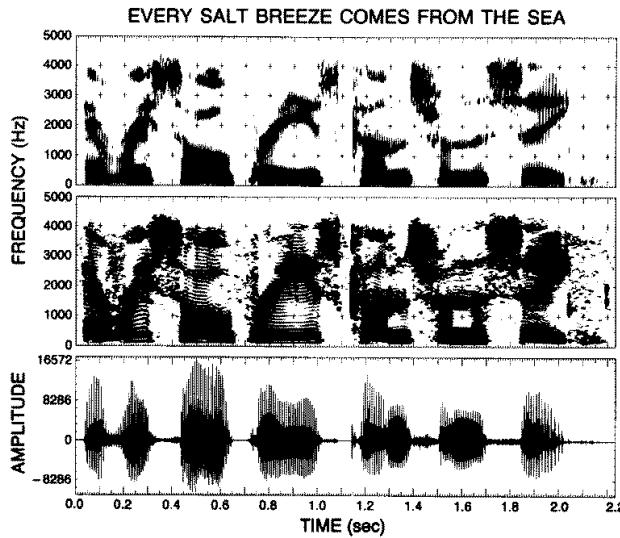
## *The production Process*



© Paul Viola 1999

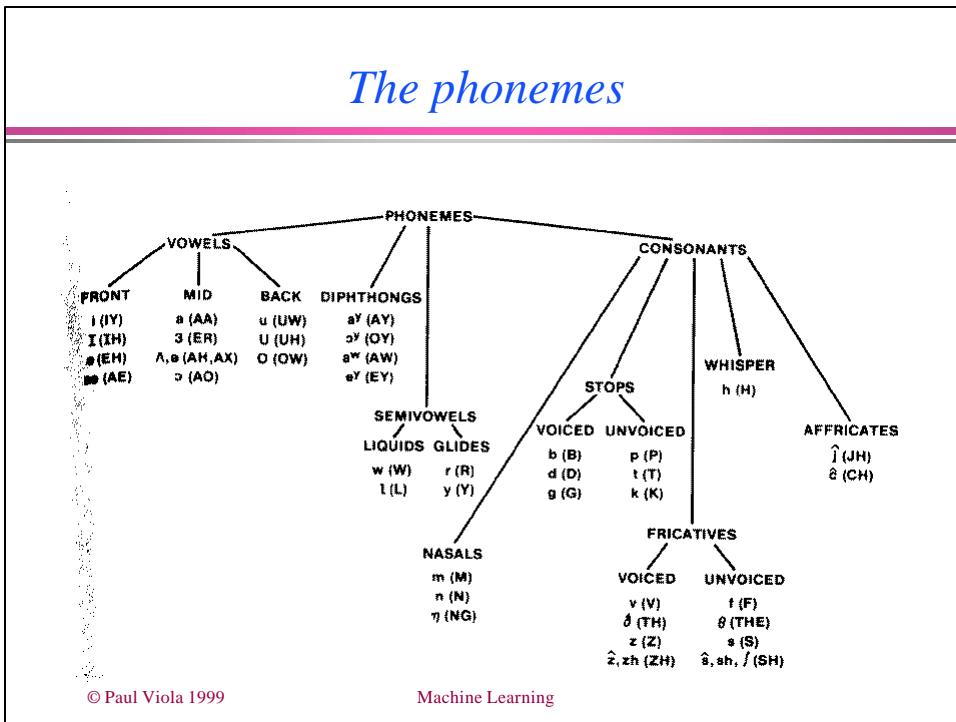
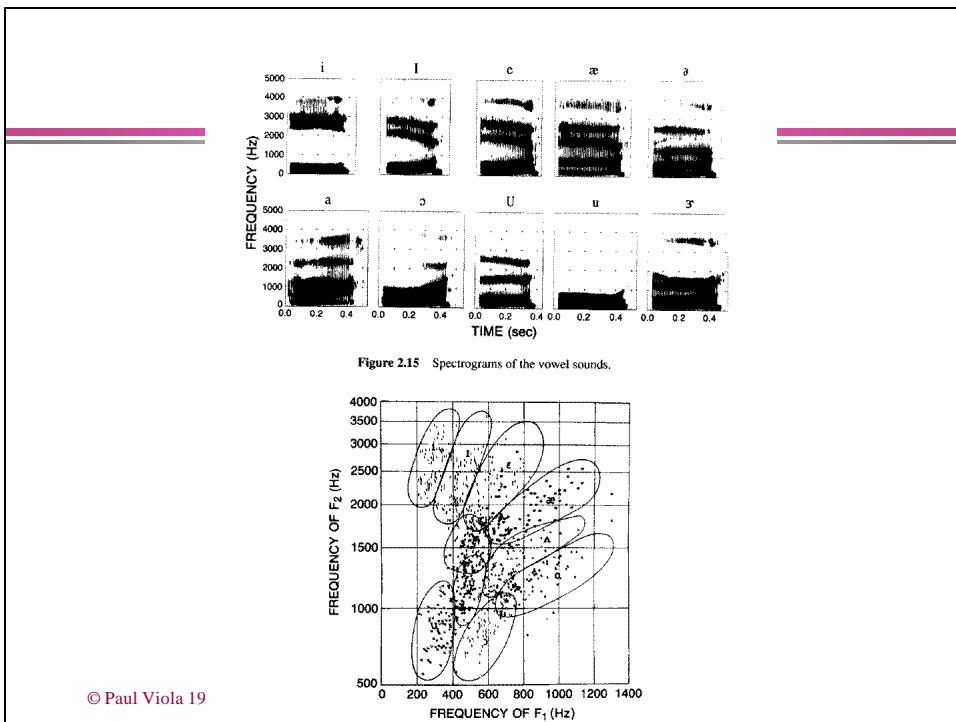
Ma

## *Differing representations*

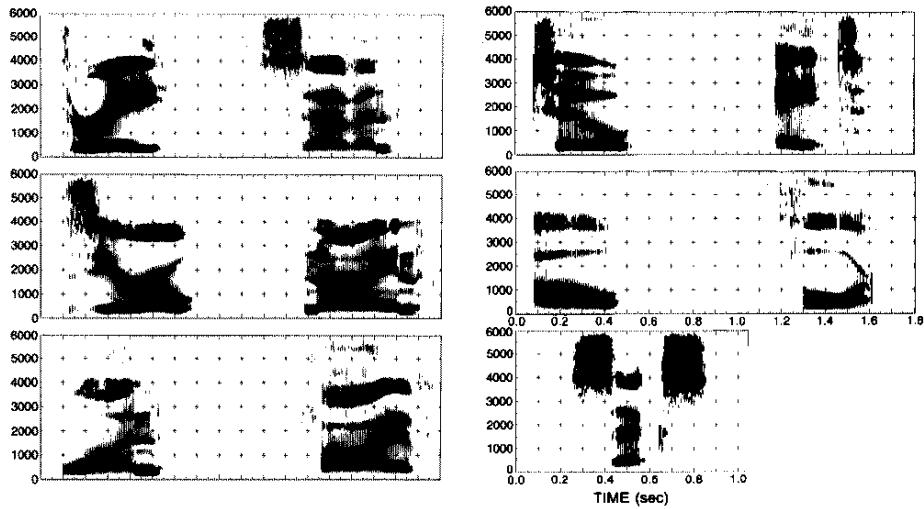


## *Spectrogram provides better information*

- The vocal tract produces sound by combining the output of multiple oscillators...
  - » Each vowel has several formants -- pure tones
- The spectrum of speech helps to distinguish vowels..
- The consonants are very different...
  - » Broad band and very brief
  - » Consonant are much harder for speech recognition systems



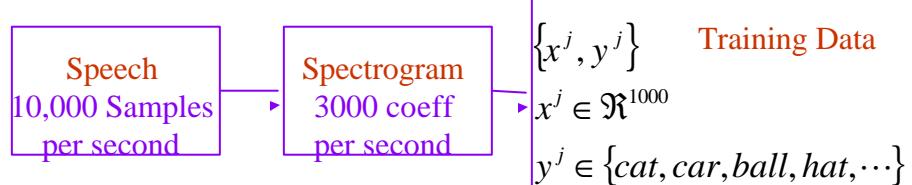
## The digits



© Paul Viola 1999

Machine Learning

## Speech using Pattern Recogniton



Note: This does not work... for many reasons!!!

© Paul Viola 1999

Machine Learning

## *Speech Difficulties*

- Rate of speech
  - » Words are spoken at different rates -- factor of 2 or 3.
- Continuous speech
  - » Where are the boundaries between words??
    - Is this your cat?  
0.2 - 0.3 - 0.6 - 0.2
    - When is your train?  
0.2 - 0.2 - 0.6 - 0.3
  - Can't build sentence recognizers  
Is this your cat?  
0.1 - 0.2 - 0.4 - 0.2
  - Other difficulties: Pitch variation, Accent, Prosody

© Paul Viola 1999

Machine Learning

## *Decompose the construction of words*

- Words are constructed from letters
  - » In written english
  - » There are 26 letters
- Words are constructed from phonemes
  - » In spoken english
  - » There are XX phonemes in English

Cat -> ‘c’ - ‘ah’ - ‘t’  
0.03 - 0.15 - 0.02

fat -> ‘f’ - ‘ah’ - ‘t’  
0.1 - 0.1 - 0.02

© Paul Viola 1999

Machine Learning

## *Implications of Decomposition*

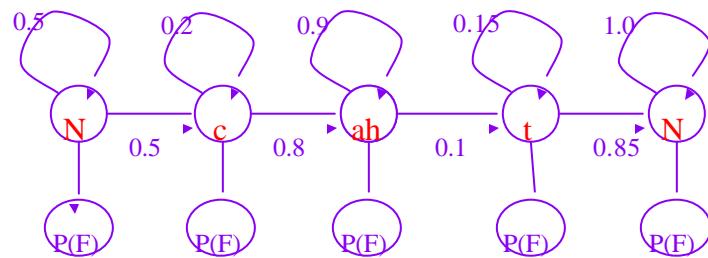
- The parts of words can be reused
  - » Words are built from XX phoneme models
  - » Perhaps we can train the phoneme recognizers separately??
  - » (Sometimes... co-articulation can make this harder)
- But, even the parts vary in length
- It can be very hard to find the beginnings and endings of phonemes
- Time is our enemy!!!

© Paul Viola 1999

Machine Learning

## *Probabilistic Models of Time*

- We've always built probabilistic models in class...
  - » Salmon vs. Bass
  - » 2's vs. 3's
  - » etc.

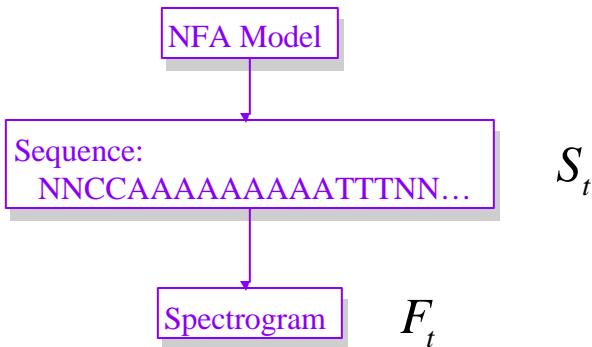


Non-deterministic Finite State Automata

© Paul Viola 1999

Machine Learning

## Phoneme Sequences



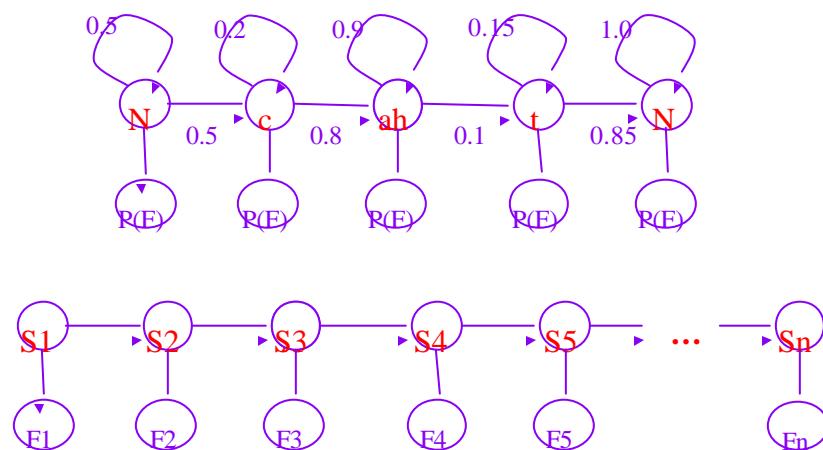
$$P(F, S \mid \text{Model}) \\ = P(F \mid S)P(S \mid \text{Model})$$

NFA model for 'cat'  
assigns a probability  
to each spectrogram

© Paul Viola 1999

Machine Learning

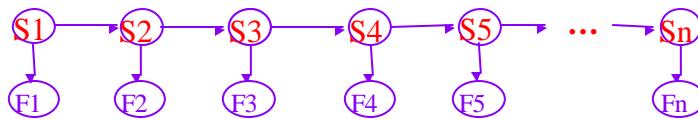
## Non-deterministic FSM -> Bayes Net



© Paul Viola 1999

Machine Learning

## The Details



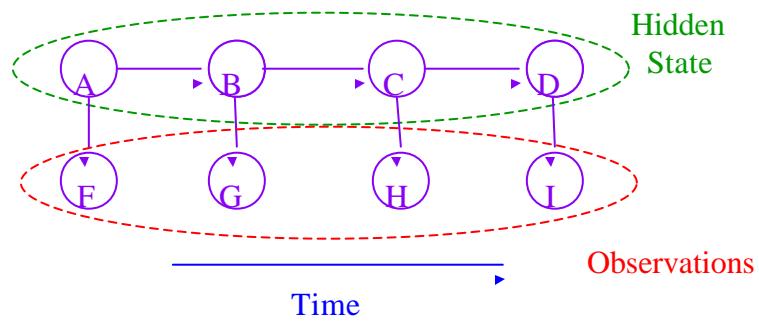
$$S_i \in \{N_B, C, A, T, N_A\}$$

$$P(S_1) = \{0.5, 0.5, 0.0, 0.0, 0.0\}$$

$$P(F_i | S_i = k) = G(F_i, \mathbf{m}_k, \Sigma_k)$$

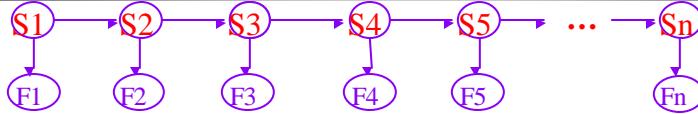
$$P(S_{i+1} | S_i) = \begin{bmatrix} & N_B & C & A & T & N_A \\ N_B & 0.5 & 0.5 & & & \\ C & & 0.2 & 0.8 & & \\ A & & & 0.9 & 0.1 & \\ T & & & & 0.15 & 0.85 \\ N_A & & & & & 1.0 \end{bmatrix}$$

## Hidden Markov Model



$$P(A, B, C, D, F, G, H, I) = P(A) P(F | A) P(B | A) P(G | B) P(C | B) P(H | C) P(D | C) P(I | D)$$

## Using Dynamic Programming...



$$\begin{aligned}
 P(F | Model) &= \sum_S P(F | S)P(S | Model) \\
 &= \sum_{\hat{S}=\{s_1, s_2, s_3, s_4, \dots\}} P(F | S = \hat{S})P(S = \hat{S} | Model) \\
 &= \sum_{\hat{S}=\{s_1, s_2, s_3, s_4, \dots\}} \left[ \prod_j P(F_j | S_j = s_j) \right] P(S = \hat{S} | Model) \\
 &= \sum_{s_1} P(F_1 = f_1 | S = s_1) P(S_1 = s_1) \sum_{s_2} P(F_2 = f_2 | S = s_2) P(S_2 = s_2 | S_1 = s_1) \sum_{s_3} \dots
 \end{aligned}$$

© Paul Viola 1999

Machine Learning

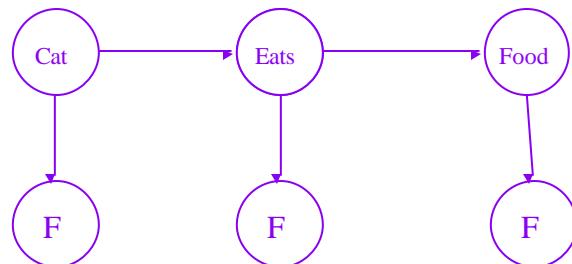
## Some standard notation...

$$\begin{aligned}
 P(F | Model) &= \sum_{s_1} P(F_1 = f_1 | S = s_1) P(S_1 = s_1) \sum_{s_2} P(F_2 = f_2 | S = s_2) P(S_2 = s_2 | S_1 = s_1) \sum_{s_3} \dots \\
 &= \sum_{s_1} T_{S_1} \sum_{s_2} T_{S_1 S_2} \sum_{s_3} T_{S_2 S_3} \sum_{s_4} T_{S_3 S_4} \sum_{s_5} T_{S_4 S_5} \\
 &= \sum_{s_1} T_{S_1} \sum_{s_2} T_{S_1 S_2} \sum_{s_3} T_{S_2 S_3} \sum_{s_4} T_{S_3 S_4} \mathbf{b}_{S_4}^4 \\
 &= \sum_{s_1} T_{S_1} \sum_{s_2} T_{S_1 S_2} \sum_{s_3} T_{S_2 S_3} \mathbf{b}_{S_3}^3 \\
 &= \sum_{s_1} T_{S_1} \sum_{s_2} T_{S_1 S_2} \mathbf{b}_{S_2}^2 \\
 &= \sum_{s_1} T_{S_1} \mathbf{b}_{S_1}^1
 \end{aligned}$$

© Paul Viola 1999

Machine Learning

## *Stringing Words Together*



© Paul Viola 1999

Machine Learning

## *Limitations of HMM for Speech*

- Observed spectrograms are independent given phonemes
  - » Does not model pronunciation or accent
- Spend most of their effort on vowels...
  - » Fat vs. Far.

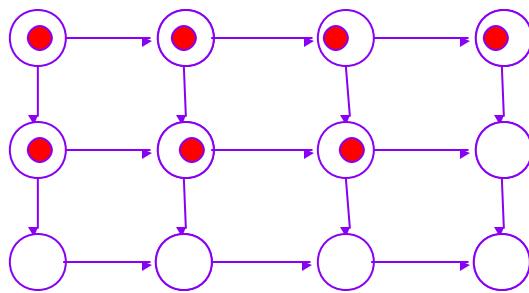
© Paul Viola 1999

Machine Learning

## Markov Processes

- Markov Processes are in fact very general...
  - » Loosely, they are processes in which there is a great deal of conditional independence.
  - Like most Bayes Nets.

$$P(A | B, C, D, F, G, H, I) \\ = P(A | p(A))$$



Note: up 'til now we have seen only directed models... the notion of Markov for undirected models is a bit more complex...

© Paul Viola 1999

Machine Learning

## Junction Tree Algorithm 1

- Table arithmetic:



$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$

$\forall a, b, c$

$$P(X = a, Y = b, Z = c) = P(X = a) P(Y = b | X = a) P(Z = c | Y = b)$$

|        | $=$ | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(X)</th> <th></th> </tr> <tr> <td>X</td> <td>0.4</td> </tr> <tr> <td>NOT X</td> <td>0.6</td> </tr> </table> $*$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(Y X)</th> <th>Y</th> <th>NOT Y</th> </tr> <tr> <td>X</td> <td>0.9</td> <td>0.1</td> </tr> <tr> <td>NOT X</td> <td>0.3</td> <td>0.7</td> </tr> </table> $*$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>P(Z Y)</th> <th>Z</th> <th>NOT Z</th> </tr> <tr> <td>Y</td> <td>0.2</td> <td>0.8</td> </tr> <tr> <td>NOT Y</td> <td>0</td> <td>1</td> </tr> </table> | P(X) |  | X | 0.4 | NOT X | 0.6 | P(Y X) | Y | NOT Y | X | 0.9 | 0.1 | NOT X | 0.3 | 0.7 | P(Z Y) | Z | NOT Z | Y | 0.2 | 0.8 | NOT Y | 0 | 1 |
|--------|-----|---|------|--|---|-----|-------|-----|--------|---|-------|---|-----|-----|-------|-----|-----|--------|---|-------|---|-----|-----|-------|---|---|
| P(X)   |     |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| X      | 0.4 |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT X  | 0.6 |   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| P(Y X) | Y   | NOT Y   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| X      | 0.9 | 0.1   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT X  | 0.3 | 0.7   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| P(Z Y) | Z   | NOT Z   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| Y      | 0.2 | 0.8   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |
| NOT Y  | 0   | 1   |      |  |   |     |       |     |        |   |       |   |     |     |       |     |     |        |   |       |   |     |     |       |   |   |

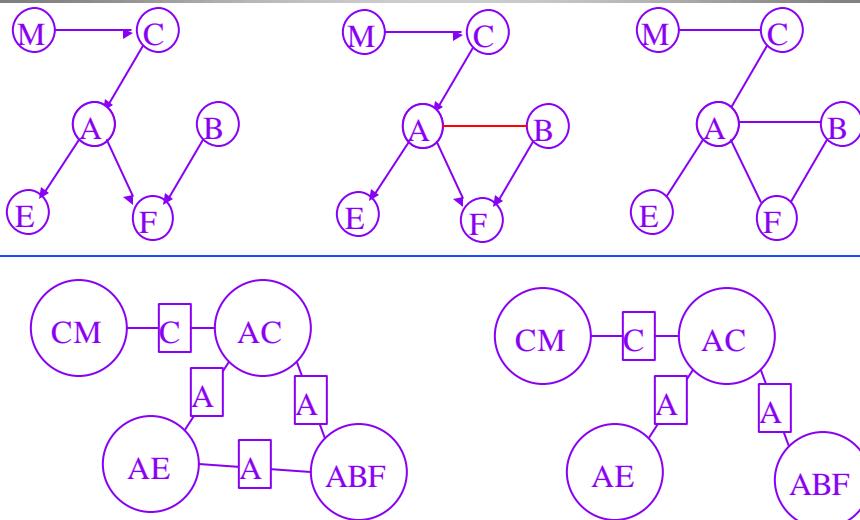
  

|           |     |                                   |
|-----------|-----|-----------------------------------|
| $T_{XYZ}$ | $=$ | $T_X \times T_{YX} \times T_{ZY}$ |
|-----------|-----|-----------------------------------|

© Paul Viola 1999

Machine Learning

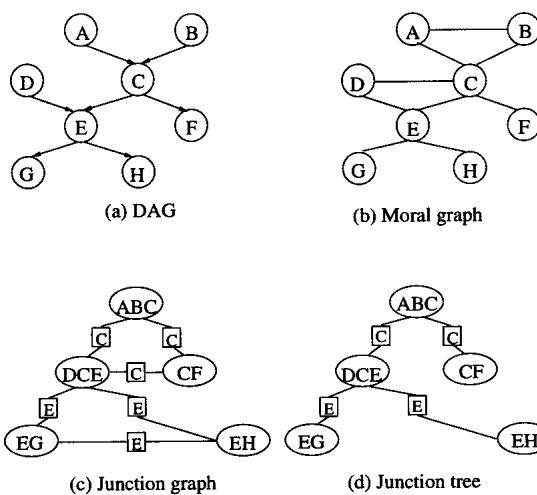
## Junction Tree Algorithm: Graph Hacking



© Paul Viola 1999

Machine Learning

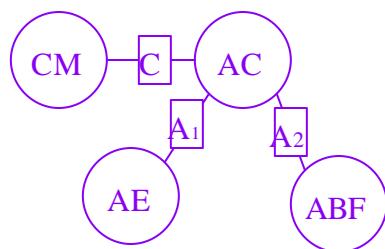
## More Junction Trees



© Paul Viola 1999

Machine Learning

## Junction Trees and Tables



$$T_{ABCDEF} = \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \quad ? = P(A, B, C, D, E, F)$$

© Paul Viola 1999

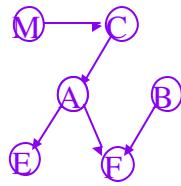
Machine Learning

## Rules for Junction Tree Initialization

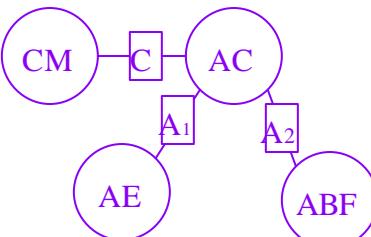
© Paul Viola 1999

Machine Learning

## From Junction Trees to Probability



$$\begin{aligned}
 P(A, B, C, D, E, F) \\
 &= P(M)P(C | M) P(A | C) \\
 &\quad \times P(E | A)P(B | F)P(F | B)
 \end{aligned}$$



$$\begin{aligned}
 T_{ABCDEF} &= \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \\
 &= P(M)P(C | M) \dots T_{CM} \\
 &\quad \times P(A | C) \dots T_{AC} \\
 &\quad \times P(E | A) \dots T_{AE} \\
 &\quad \times P(B)P(F | B) \dots T_{ABF}
 \end{aligned}$$

© Paul Viola 1999

Machine Learning

# *6.891 Machine Learning and Neural Networks*

Lecture 18:  
Finish Hidden Markov Models  
& Finish Bayes Nets

© Paul Viola 1999

Machine Learning

## *News*

- Remember to keep thinking about your final projects!
- The reading for this class is now mostly the supplemental material on the **related-info** page.
  - » You need to at scan the materials there.

© Paul Viola 1999

Machine Learning

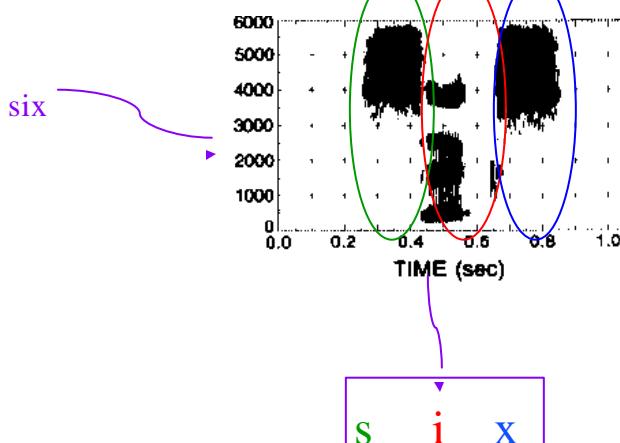
## *Review & Overview*

- Lecture 17:
  - » Hidden Markov Models for Speech
    - Speech is complex...
      - Many words / Length of words varies
    - Speech is best represented as a spectrogram
    - Variable timing of speech can be modeled as a NFA.
    - An HMM is a Bayes Net which is equivalent to an NFA
      - We can build an HMM for each word out of phoneme models
    - Can sum over the unknown states to recognize words
- More HMM examples
- Finding the most likely state sequence

© Paul Viola 1999

Machine Learning

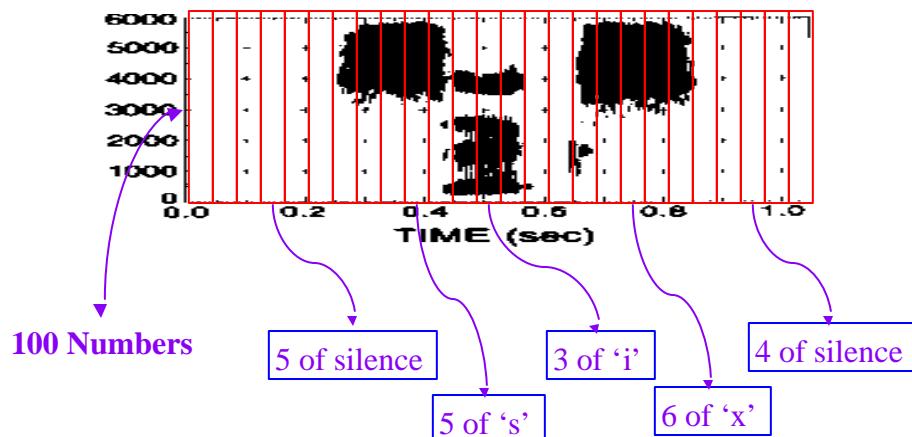
## *Speech in a Nutshell*



© Paul Viola 1999

Machine Learning

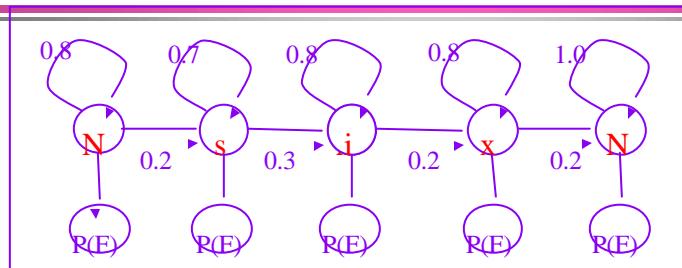
## Closer Examination



© Paul Viola 1999

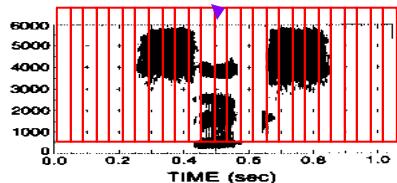
Machine Learning

*Build a Probabilistic Model  
which generates the data...*



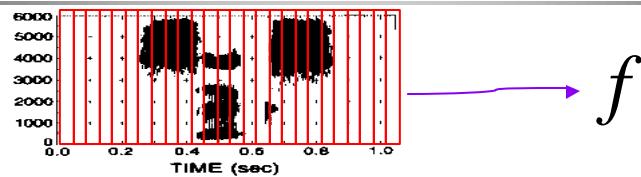
$$P(F, S | \text{Model})$$

$$= P(F | S)P(S | \text{Model})$$



Machine Learning

## Use Bayes Law



$$P(F = f | 'six') \\ = \sum_s P(F = f, S = s | 'six')$$

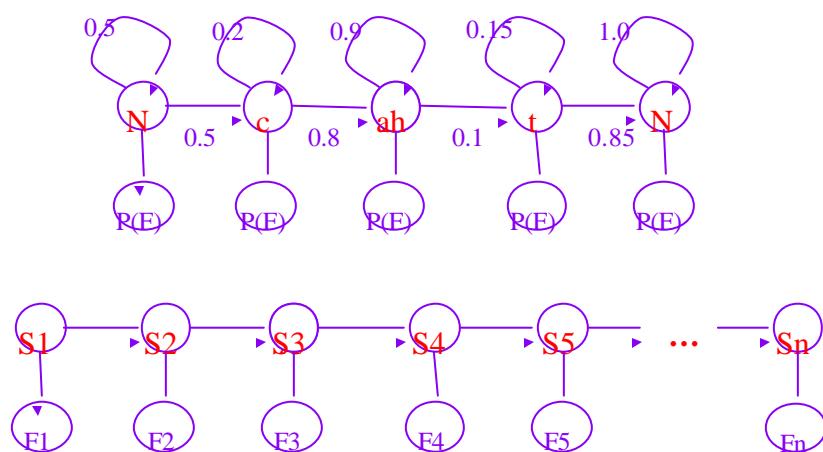
vs.

$$P(F = f | 'five') \\ = \sum_s P(F = f, S = s | 'five')$$

© Paul Viola 1999

Machine Learning

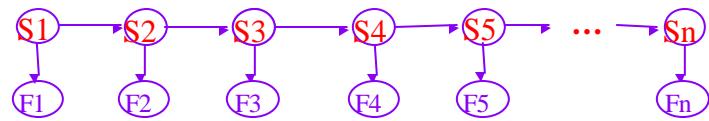
## Non-deterministic FSM -> Bayes Net



© Paul Viola 1999

Machine Learning

## A concrete example



$$S_i \in \{1, 2\}$$

$$P(S_1) = \{0.5, 0.5\}$$

$$P(S_{i+1} | S_i) = \begin{bmatrix} 1 & 2 \\ 1 & 0.9 & 0.1 \\ 2 & 0.1 & 0.9 \end{bmatrix}$$

$$P(F_i = f | S_i = 1) = G(f, 1.0, 0.1)$$

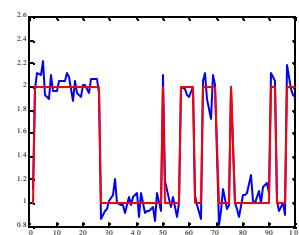
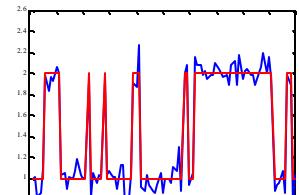
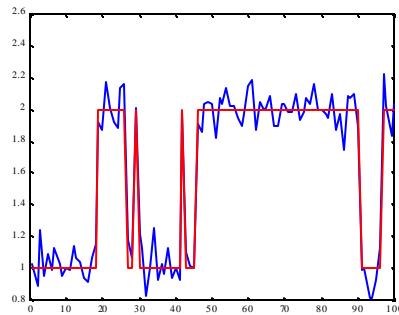
$$P(F_i = f | S_i = 2) = G(f, 2.0, 0.1)$$

$$P(F, S | Model)$$

© Paul Viola 1999

Machine Learning

## Some Samples



© Paul Viola 1999

Machine Learning

*Code is very simple...*

```
function [states, obs] = hmm_draw(n, initial, transition, obs_models)

% function [states, obs] = hmm_draw(n, initial, transition, obs_models)
%
% Draw a sample of the HMM running for N steps

% Setup the initial space
states = zeros(n, 1);
obs = zeros(n, length(hmm_observe(2, obs_models)));

states(1) = hmm_draw_state(initial);

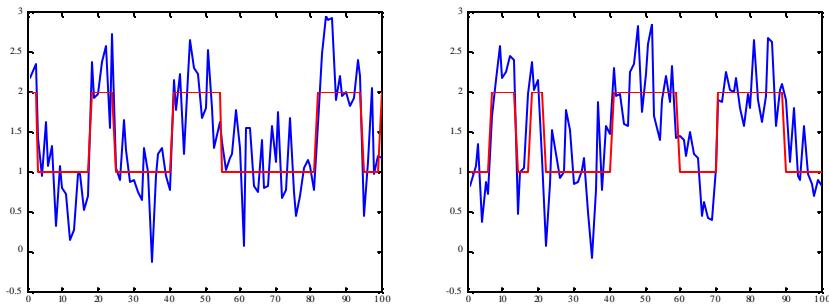
for i = 2:n
    % transition(states(i-1), :)
    states(i) = hmm_draw_state(transition(states(i-1), :));
end

obs = hmm_observe(states, obs_models);
```

© Paul Viola 1999

Machine Learning

*Add more noise...*



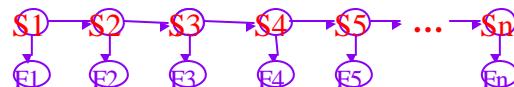
$$P(F_i = f \mid S_i = 1) = G(f, 1.0, 0.4)$$

$$P(F_i = f \mid S_i = 2) = G(f, 2.0, 0.4)$$

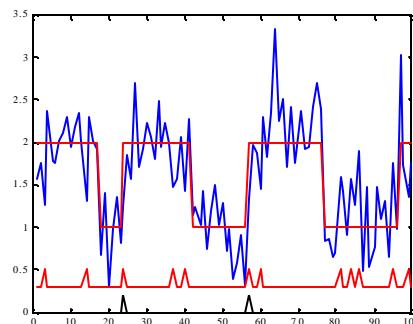
© Paul Viola 1999

Machine Learning

*But we have a detailed model*

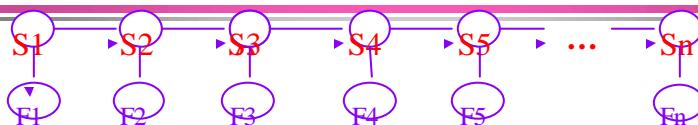


$$\arg \max_s P(F = f, S = s | Model)$$



© Paul Viola 19

*Using Dynamic Programming...*



$$\arg \max_s P(F = f, S = s | Model)$$

$$= \arg \max_s P(s_1)P(f_1 | s_1)P(s_2 | s_1)P(f_2 | s_2) \dots P(s_n | s_{n-1})P(f_n | s_n)$$

© Paul Viola 1999

Machine Learning

```

% Propagate the maximum state forward in time from the beginning
maxes = state_like;
maxes(1,:) = maxes(1,:)/sum(maxes(1,:));

for i = 2:nTimes
    for j = 1:nStates
        % For each new time, check each of the past states and to determine
        % the best state given the transition costs.
        for k = 1:nStates
            vals(j,k) = maxes(i-1,k) * trans(k,j) * state_like(i,j);
        end
        maxes(i,j) = max(vals(j,:));
    end
    maxes(i,:) = maxes(i,:)/sum(maxes(i,:));
end

vals = zeros(nStates,1);
shat = zeros(size(maxes));

[v ind] = max(maxes(nTimes, :));
shat(nTimes, ind) = 1;

for i = nTimes-1:-1:1
    for j = 1:nStates
        vals(j) = trans(ind,j) * maxes(i,j);
    end
    [v ind] = max(vals);
    shat(i, ind) = 1;
end

```

## What about distinguishing two models??

$$S_i \in \{1, 2\}$$

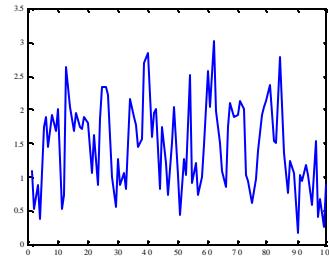
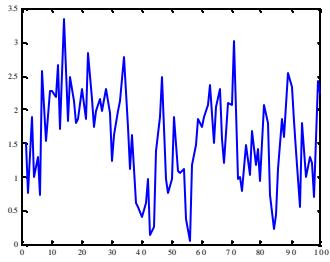
$$P(S_1) = \{0.5, 0.5\}$$

$$P(S_{i+1} | S_i) = \begin{bmatrix} & 1 & 2 \\ 1 & 0.9 & 0.1 \\ 2 & 0.1 & 0.9 \end{bmatrix}$$

$$S_i \in \{1, 2\}$$

$$P(S_1) = \{0.5, 0.5\}$$

$$P(S_{i+1} | S_i) = \begin{bmatrix} & 1 & 2 \\ 1 & 0.8 & 0.2 \\ 2 & 0.2 & 0.8 \end{bmatrix}$$



## *Code for model likelihood...*

```
function like = hmm_model_likelihood(f, initial, trans, obs_models)

% function shat = hmm_model_likelihood(f, initial, trans, obs_models)

% First compute the likelihood of every state given every observation
state_like = hmm_obs_likelihood(f, obs_models);

% initialize some variables
ntimes = size(state_like, 1);
nstates = size(state_like, 2);

mfactor = 100;

beta = mfactor .* state_like(ntimes,:);

for i = ntimes-1:-1:1
    beta = mfactor .* (state_like(i,:) .* (trans * beta')');
end

like = log10(sum(beta))-(log10(mfactor) * ntimes);
```

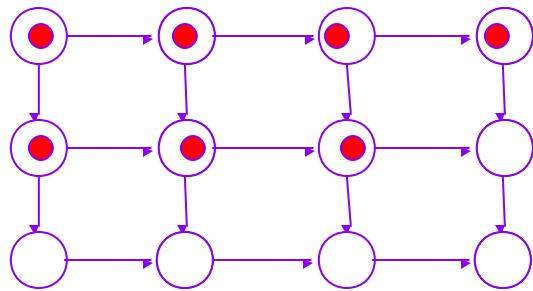
## *Limitations of HMM for Speech*

- Observed spectrograms are independent given phonemes
  - » Does not model pronunciation or accent
  - » Does not model inter word dependencies
- Spend most of their effort on vowels...
  - » mount vs. won't

## *Markov Processes*

- Markov Processes are in fact very general...
  - » Loosely, they are processes in which there is a great deal of conditional independence.
    - Like most Bayes Nets.

$$P(A | B, C, D, F, G, H, I) \\ = P(A | p(A))$$



Note: up 'til now we have seen only directed models... the notion of Markov for undirected models is a bit more complex...

© Paul Viola 1999

Machine Learning

## *Segue*

- We have seen several applications of Bayesian Networks...
  - » Expert Systems
  - » Diagnosis
  - » Speech Recognition
- Are there other algorithms for reasoning on Bayes Nets...
  - » Junction Trees
  - » Propagation algorithms
  - » Makes it easy to measure marginals...

© Paul Viola 1999

Machine Learning

## Junction Tree Algorithm 1

- Table arithmetic:



$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$

$\forall a, b, c$

$$P(X = a, Y = b, Z = c) = P(X = a) P(Y = b | X = a) P(Z = c | Y = b)$$

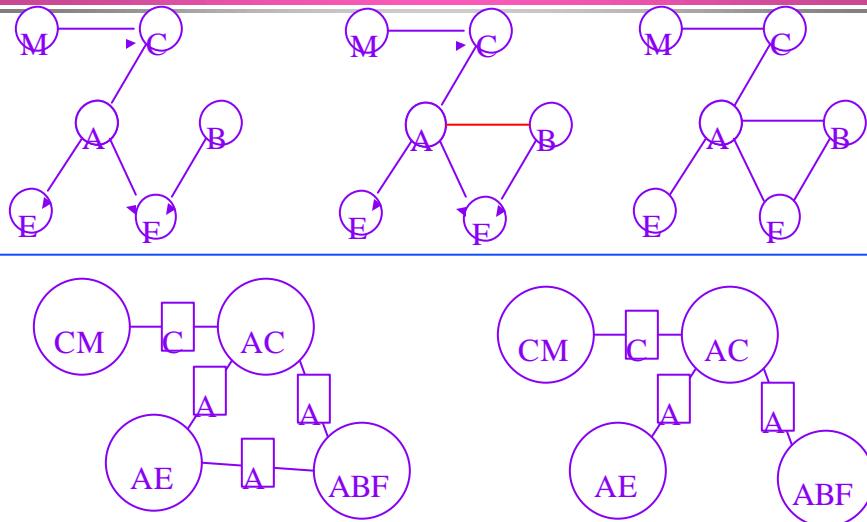
$$\begin{array}{c} \text{Z} \\ \text{X} \\ \text{Y} \end{array} = \begin{array}{c|c} \text{P}(X) & \\ \hline \text{X} & 0.4 \\ \text{NOT X} & 0.6 \end{array} * \begin{array}{c|cc} \text{P}(Y|X) & Y & \text{NOT Y} \\ \hline \text{X} & 0.9 & 0.1 \\ \text{NOT X} & 0.3 & 0.7 \end{array} * \begin{array}{c|cc} \text{P}(Z|Y) & Z & \text{NOT Z} \\ \hline \text{Y} & 0.2 & 0.8 \\ \text{NOT Y} & 0 & 1 \end{array}$$

$$T_{XYZ} = T_X \times T_{YX} \times T_{ZY}$$

© Paul Viola 1999

Machine Learning

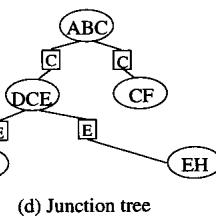
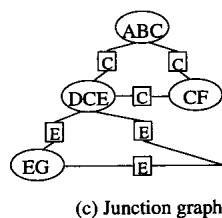
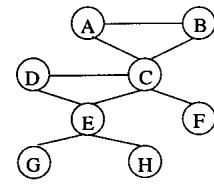
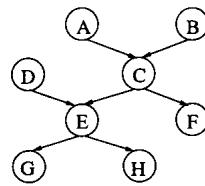
## Junction Tree Algorithm: Graph Hacking



© Paul Viola 1999

Machine Learning

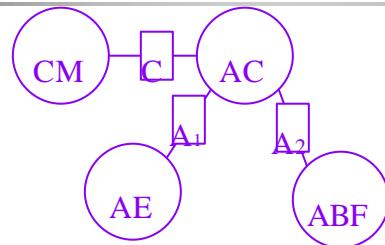
## More Junction Trees



© Paul Viola 1999

Machine Learning

## Junction Trees and Tables



$$T_{ABCDEF} = \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \quad ? = P(A, B, C, D, E, F)$$

© Paul Viola 1999

Machine Learning

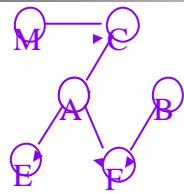
## *Rules for Junction Tree Initialization*

- For each conditional distribution in the Bayes Net
  - » Find a node in the Jtree which contains all those vars
  - » Multiply that nodes table by the conditional dist

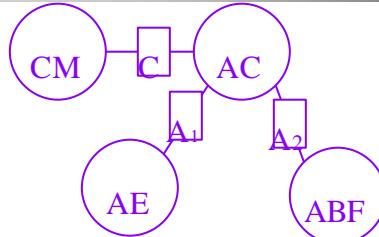
© Paul Viola 1999

Machine Learning

## *From Junction Trees to Probability*



$$\begin{aligned} P(A, B, C, D, E, F) \\ = P(M)P(C | M) P(A | C) \\ \times P(E | A)P(B)P(F | AB) \end{aligned}$$

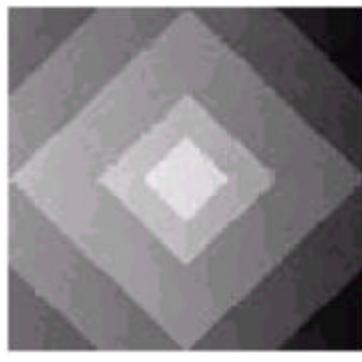
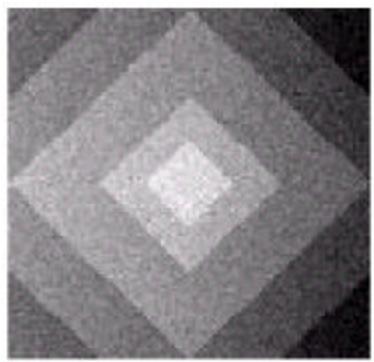


$$\begin{aligned} T_{ABCDEF} &= \frac{T_{CM} \times T_{AC} \times T_{AE} \times T_{ABF}}{S_C \times S_{A1} \times S_{A2}} \\ &= P(M)P(C | M) \dots T_{CM} \\ &\quad \times P(A | C) \dots T_{AC} \\ &\quad \times P(E | A) \dots T_{AE} \\ &\quad \times P(B)P(F | AB) \dots T_{ABF} \end{aligned}$$

© Paul Viola 1999

Machine Learning

## *Image Markov Models*



© Paul Viola 1999

Machine Learning

# Multi-scale Statistical Models: Images, People, Movement

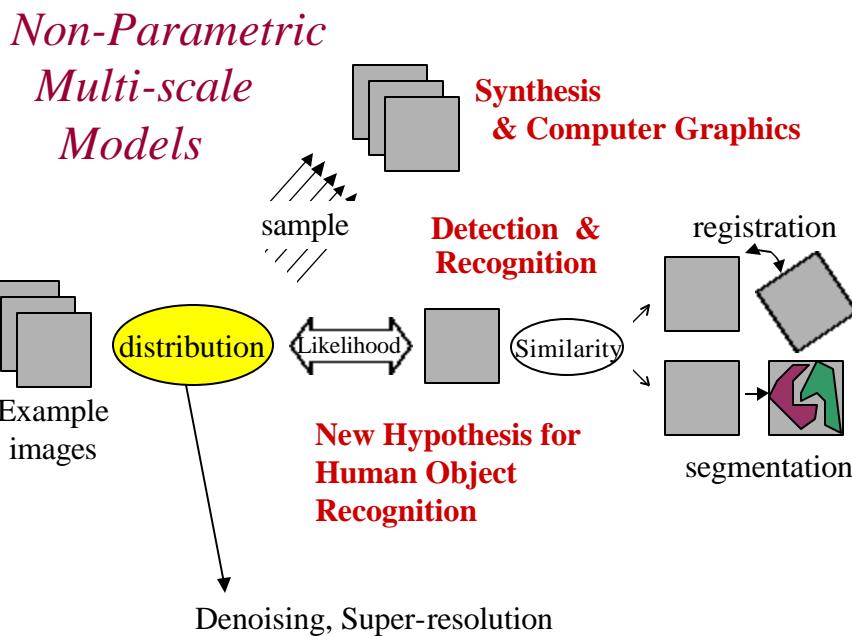
Paul Viola

Collaborators:     Jeremy De Bonet,  
                     John Fisher, Andrew Kim  
                     Tom Rikert, Mike Jones,

<http://www.ai.mit.edu/projects/lv>

*Paul Viola*

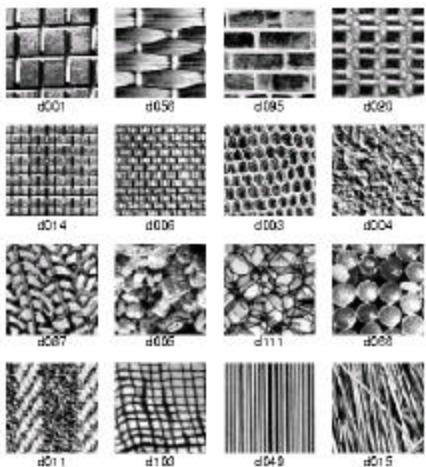
*MIT AI Lab*



*Paul Viola*

*MIT AI Lab*

## *Visual Texture: a testing ground*



*Paul Viola*

- Texture
  - Random Repeating Process
  - No two patches are identical

Good statistical  
model for images

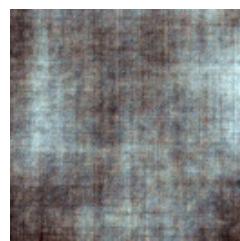
Good model  
for visual texture

*MIT AI Lab*

## *Generation a critical test*



Input  
Texture



Gaussian



Independent



Non-parametric  
Multi-scale

*Paul Viola*

## *Simple Statistical Model 1: Independent pixels*



- Statistical Model 1
  - Each pixel is independent and identically distributed

$$P(I) = \prod_{x,y} P(I_{xy})$$

*Paul Viola*

*MIT AI Lab*

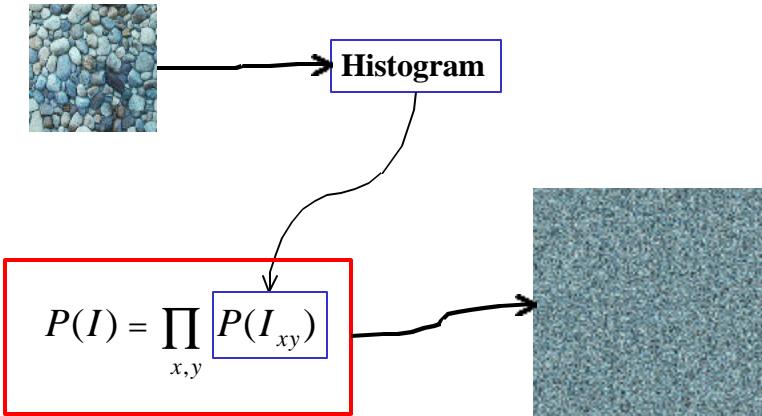
## *Technical Point: Texture is Ergodic/Stationary*

- A texture image is assumed to be many samples of a single process
  - Each sample is almost certainly dependent on the other samples
  - But actual location of the samples does not matter
  - (Space invariant process).

*Paul Viola*

*MIT AI Lab*

## *Simple Statistical Models* *Independent pixels*



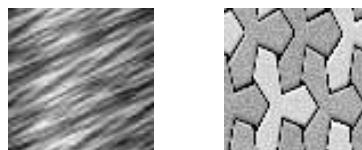
*Paul Viola*

*MIT AI Lab*

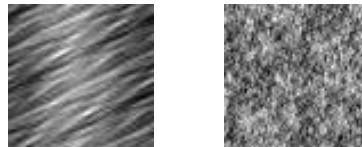
## *Statistical Model 2:* *Gaussian Distribution*

$$P(I) = N(I, \mu, \Sigma)$$
$$\propto e^{-\frac{1}{2} (I - \mu)^T \Sigma^{-1} (I - \mu)}$$

Original



Generated



*Paul Viola*

*MIT AI Lab*

## What else are probabilistic image models good for??

- Denoising:

– If we have a model for:  $P(I)$

– And we observe an image plus noise:  $\hat{I} = I + \eta$

$$P(\hat{I}) = \int P(I = \hat{I} - \eta, \eta) d\eta = \int P(I = \hat{I} - \eta) P(\eta) d\eta$$

$$P(I | \hat{I}) = \frac{P(\hat{I} | I)P(I)}{P(\hat{I})} = \frac{P(\eta = \hat{I} - I)P(I)}{P(\hat{I})}$$

$$E[I | \hat{I}] = \int \frac{I P(\eta = \hat{I} - I)P(I)}{P(\hat{I})} dI$$

*Paul Viola*

*MIT AI Lab*

## What if $I$ were a scalar?

And the both signal and noise were Gaussian

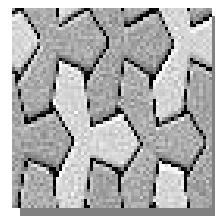
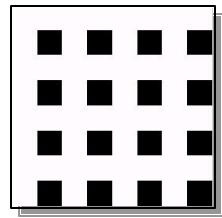
$$\begin{aligned} E[I | \hat{I}] &= \int \frac{I P(\eta = \hat{I} - I)P(I)}{P(\hat{I})} dI \\ &= \int \frac{I e^{(\hat{I}-I)^2} e^{(I-\mu)^2}}{c} dI \end{aligned}$$

Same thing as estimating the mean of a gaussian from one example and there is a prior...  
the expected value is between the observation and prior

*Paul Viola*

*MIT AI Lab*

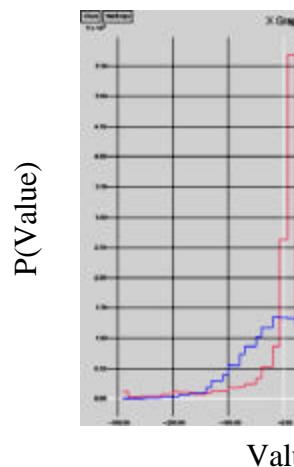
*Gaussian are not quite right...*



*Paul Viola*

*MIT AI Lab*

*Gaussian model fails other tests also...*



Derivative  
Gaussian Fit

Every linear projection  
of a Gaussian must be  
Gaussian...  
**yet the derivatives in  
images are far from  
Gaussian**

*Paul Viola*

*MIT AI Lab*

## Statistical Model 3: Independent Wavelet Models

- Donoho, Adelson, Simoncelli, etc.
- Very efficient (linear time)
  - Estimation, Sampling, Inference

$$P(I) \propto \prod_j P_j([WI]_j)$$

- $P(I)$  is defined implicitly
  - As a distribution over the features present in an image
- $W$  is a Wavelet or tight frame operator
  - Invertibility is key...

*Paul Viola*

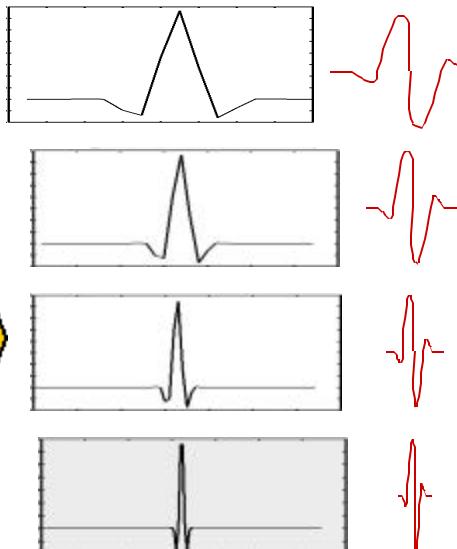
*MIT AI Lab*

### 1D Wavelet Transform



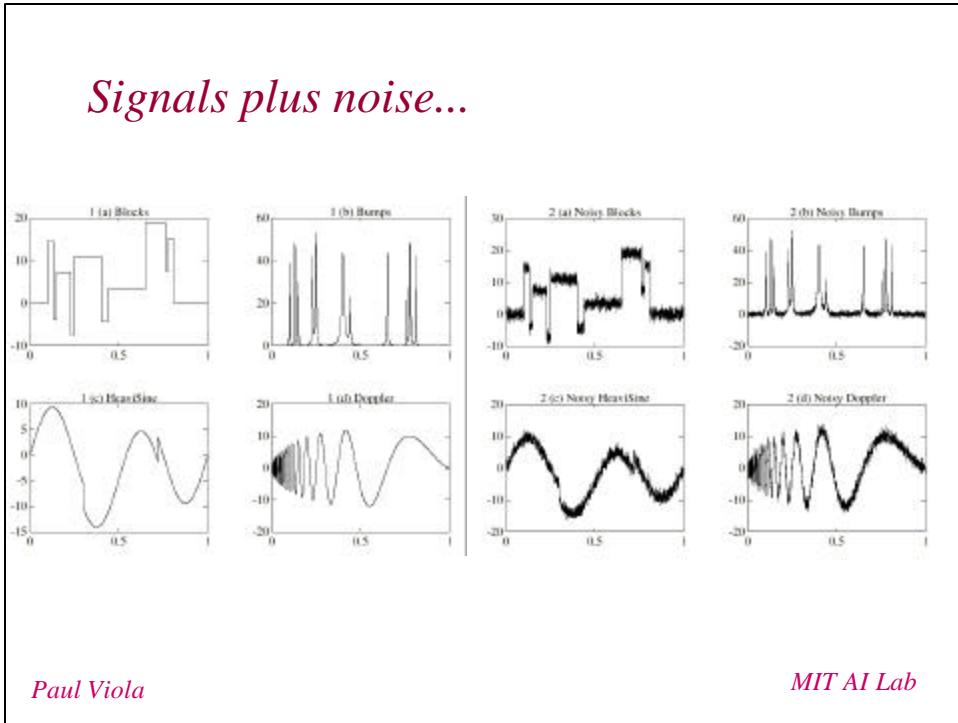
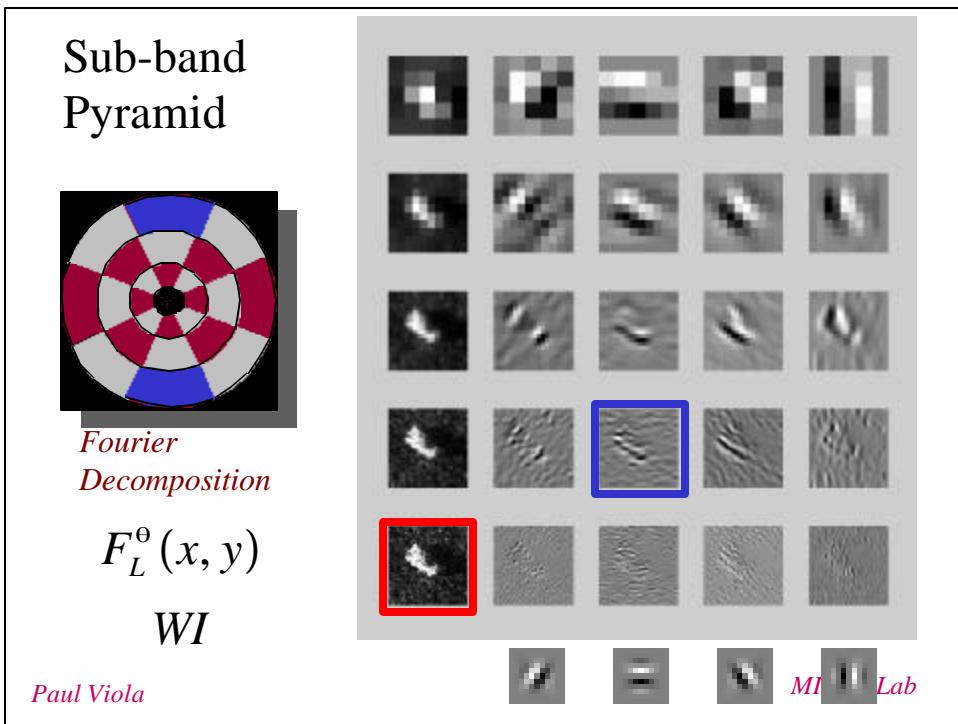
Simple Input  
Texture

Wavelet  
Transform

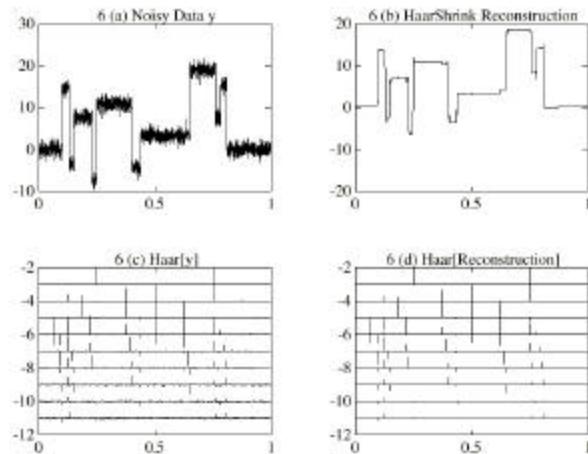


*MIT Filters*

*Paul Viola*



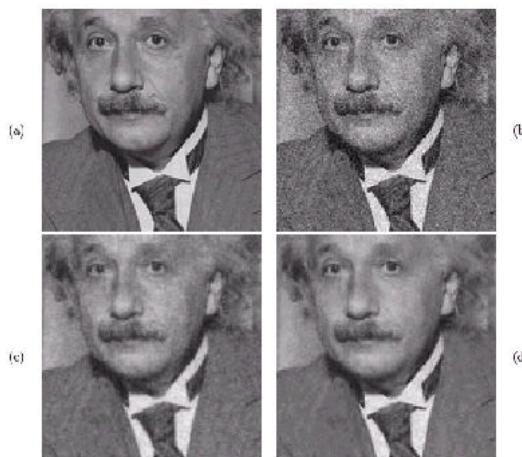
## Noise removal through shrinkage



Paul Viola

MIT AI Lab

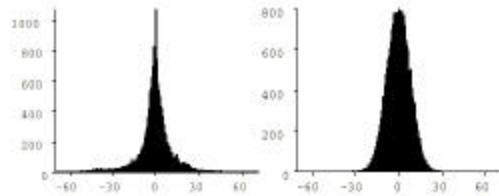
## Removing noise from images



Paul Viola

MIT AI Lab

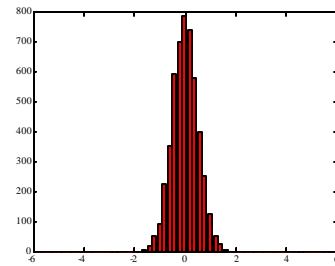
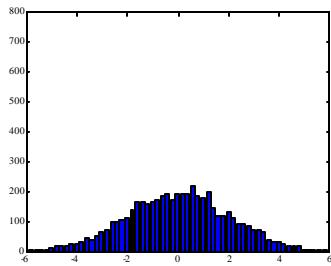
*Inside the guts...*



*Paul Viola*

*MIT AI Lab*

*Noise + Signal: Two Gaussian Case*



*Paul*

*b*

$$E[I | \hat{I}] = \int \frac{I P(\eta = \hat{I} - I)P(I)}{P(\hat{I})} dI$$

$$= \int \frac{I e^{\frac{-(\hat{I}-I)^2}{2n^2}} e^{\frac{-(I)^2}{2s^2}}}{c} dI$$

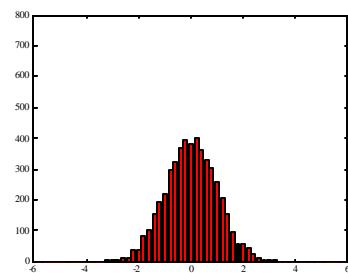
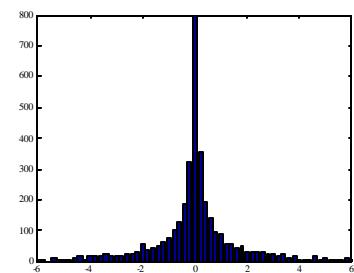
$$\frac{-(\hat{I}-I)^2}{2n^2} - \frac{(I)^2}{2s^2} = - \left[ \frac{s^2(I^2 - 2I\hat{I} + \hat{I}^2) + n^2I^2}{2n^2s^2} \right]$$

$$= - \left[ \frac{(s^2 + n^2)I^2 - 2s^2I\hat{I} + s^2\hat{I}^2}{2n^2s^2} \right]$$

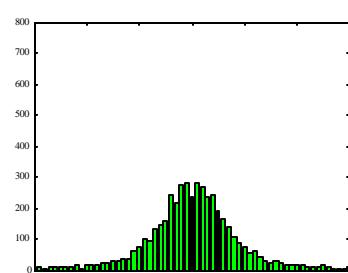
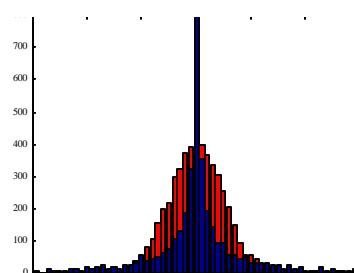
$$= - \left[ \frac{I^2 - \frac{2s^2I\hat{I}}{(s^2 + n^2)} + \frac{s^2\hat{I}^2}{(s^2 + n^2)}}{2n^2s^2} \right]$$

*Paul Viola*

### Noise vs. Signal: The details



*i*



*ab*

## *Independent Wavelet Synthesis Model*

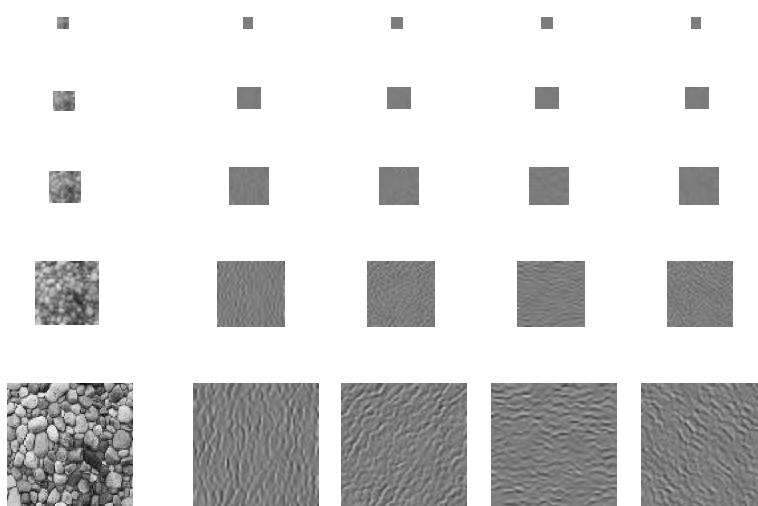
$$\begin{aligned} P(I) &\approx \prod_{l,\theta,x,y} P_{l,\theta,x,y}(F_l^{\theta}(x,y)) \\ &\approx \prod_{l,\theta,x,y} P_{l,\theta}(F_l^{\theta}(x,y)) \end{aligned}$$

*Given :*  $I, W$   
*Observe :*  $O_{l,\theta} = \{F_l^{\theta}(x,y)\}$   
*Model :*  $P_{l,\theta}(\cdot)$

*Paul Viola*

*MIT AI Lab*

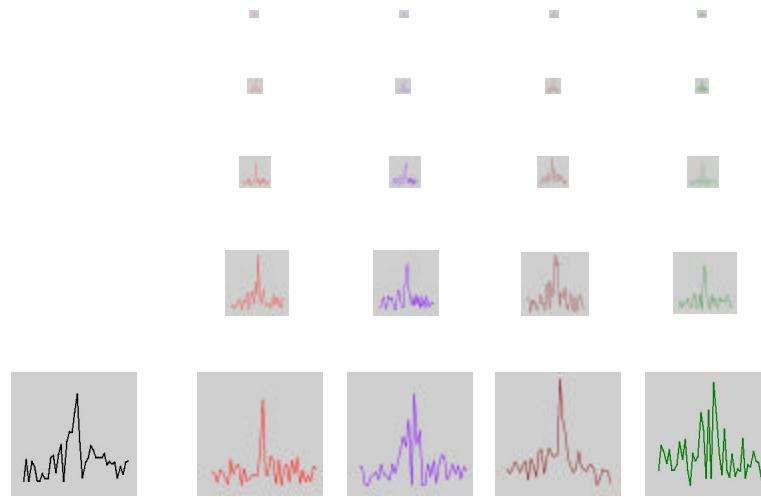
## Observe Coefficients



*Paul Viola*

*MIT AI Lab*

## Compute Histograms



*Paul Viola*

*MIT AI Lab*

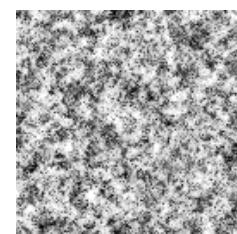
## Multi-scale Histograms

original  
texture patch



Sampling Procedure

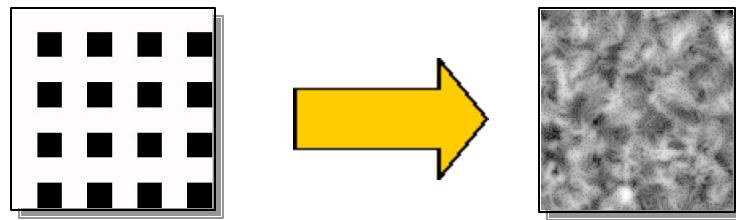
## Multi-scale Sampling



*Paul Viola*

*MIT AI Lab*

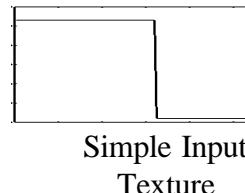
*Not quite right...*



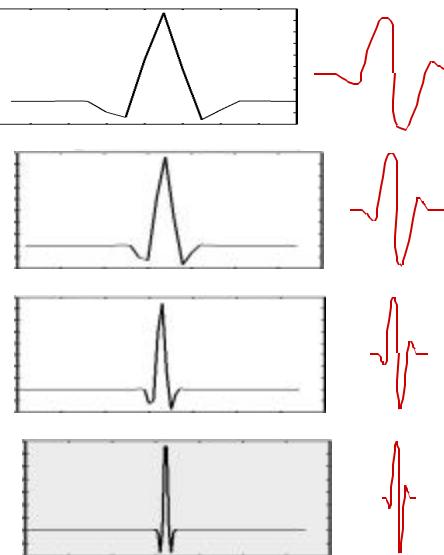
*Paul Viola*

*MIT AI Lab*

*Edges lead to aligned  
coefficients*



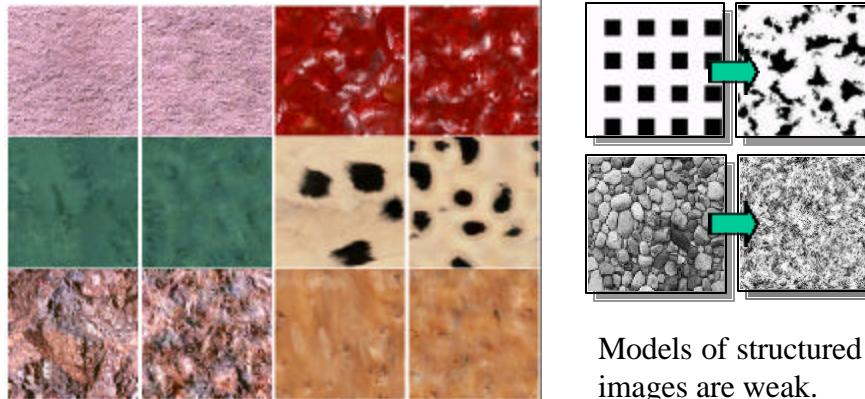
Wavelet  
Transform



*Paul Viola*

*MIT Filters*

## *Heeger and Bergen: Constrain the pixel histogram*



Models of structured images are weak.

*Paul Viola*

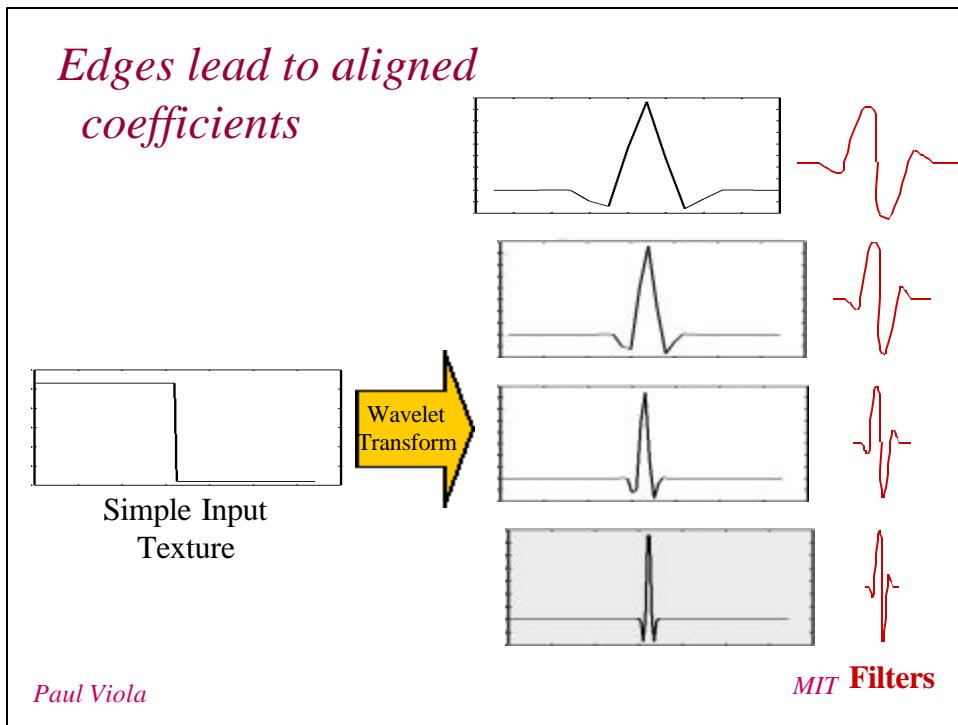
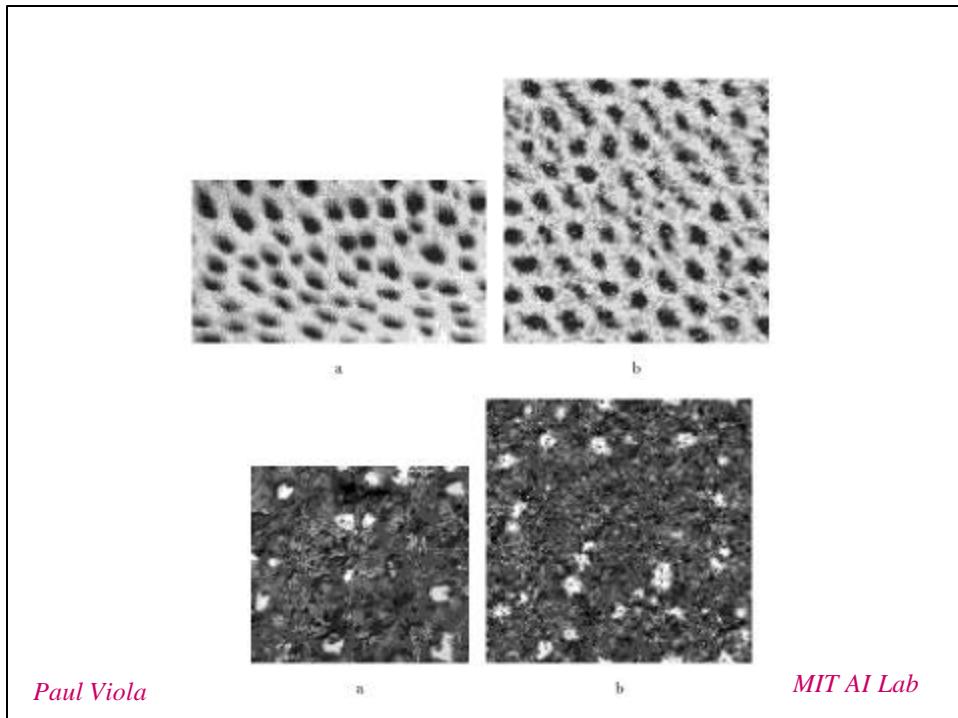
*MIT AI Lab*

## *FRAME: a generalization of B&H (Zu, Wu and Mumford)*

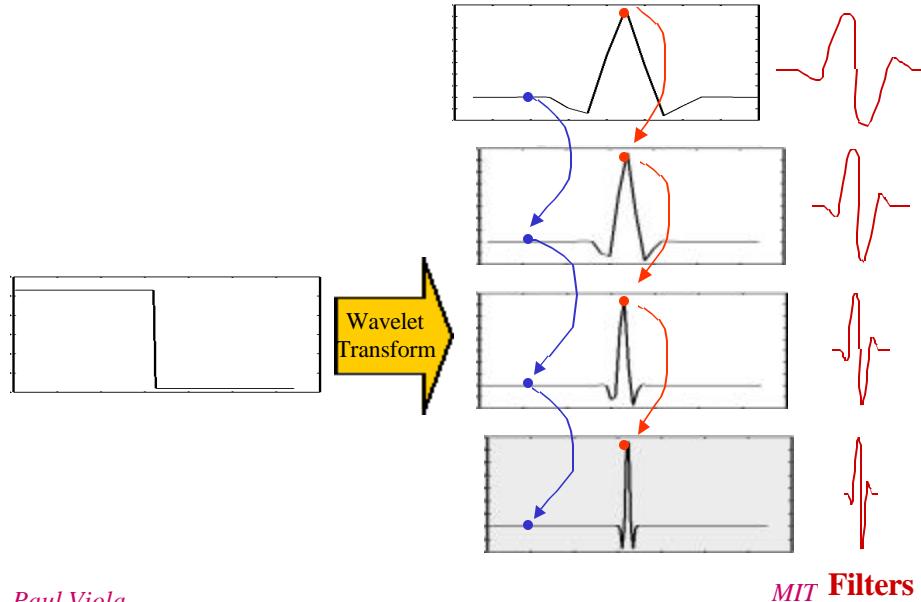
- Specify a set of filters
  - Not necessarily orthogonal or even linear.
- Measure the histogram of these filters
  - Type of statistic
- Construct a Boltzmann/Gibbs distribution which generates these statistics
  - Maximum Entropy
- Resulting algorithm is currently intractible
  - Days to generate a single image

*Paul Viola*

*MIT AI Lab*



## *Preserving Cross Scale Alignment*

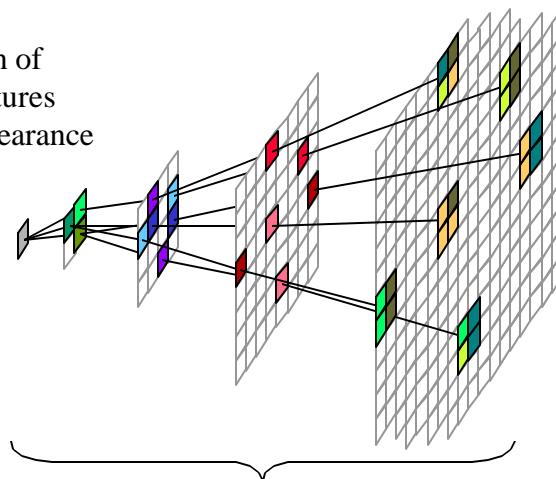


*Paul Viola*

*MIT Filters*

## *Statistical Distribution of Multi-scale Features*

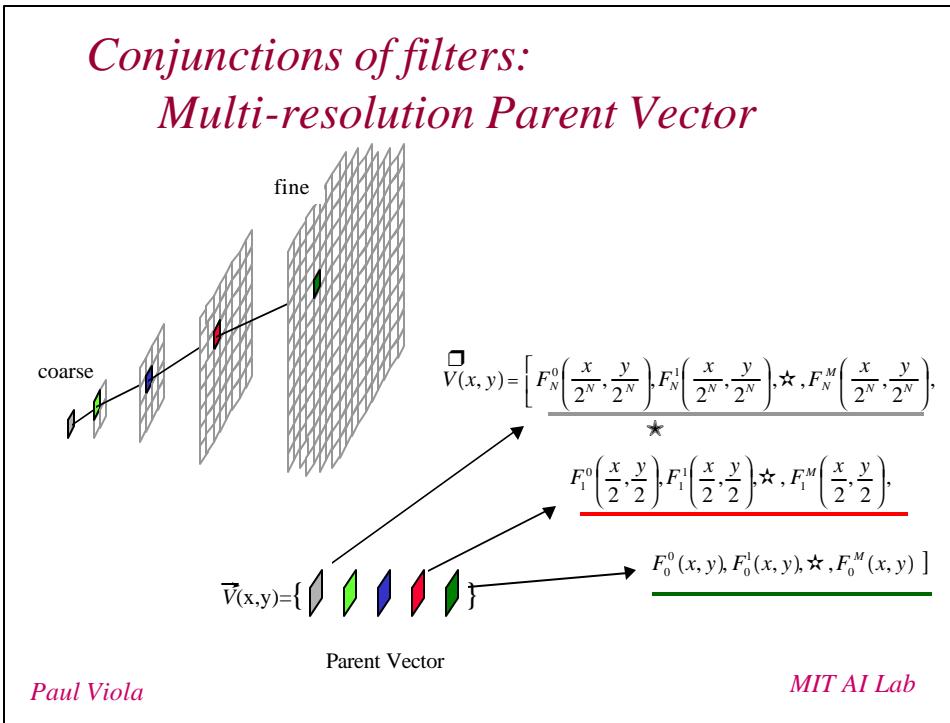
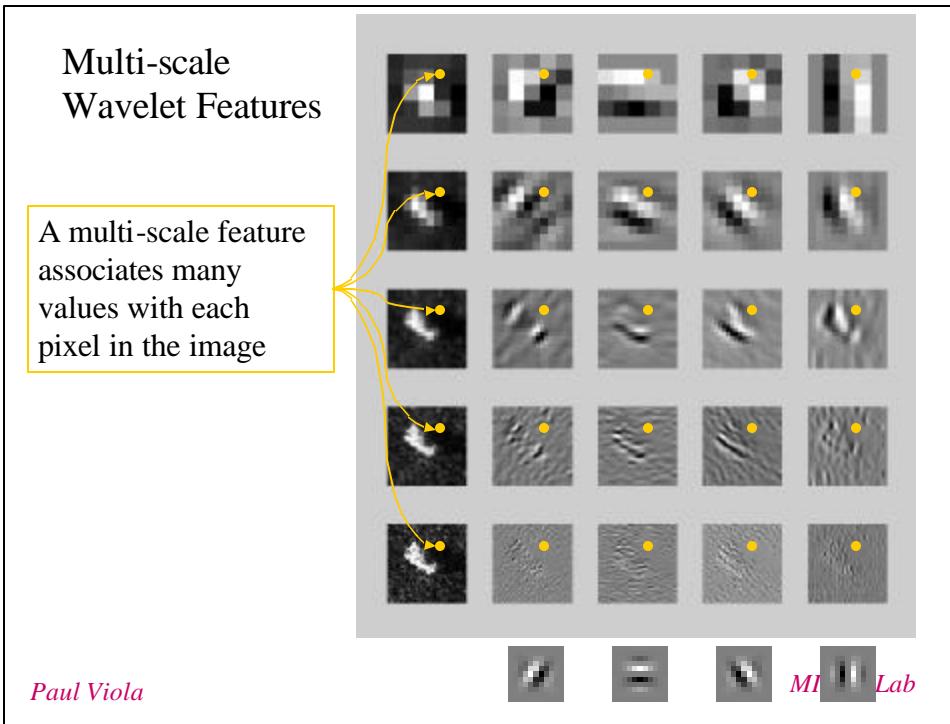
The distribution of  
multi-scale features  
determines appearance



*Paul Viola*

*Wavelet Pyramid*

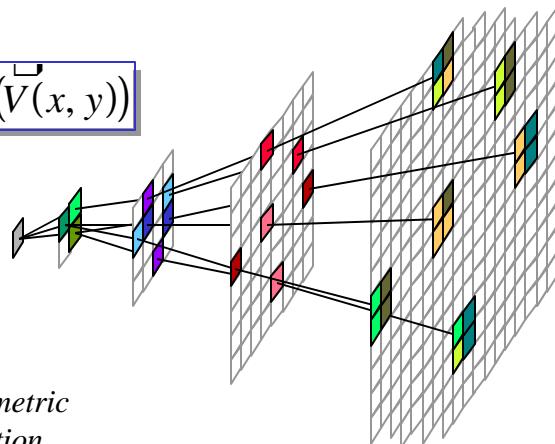
*MIT AI Lab*



## *Build a Model for Observed Distribution*

$$P(I) = P(V(x, y))$$

*Non-parametric  
Distribution*

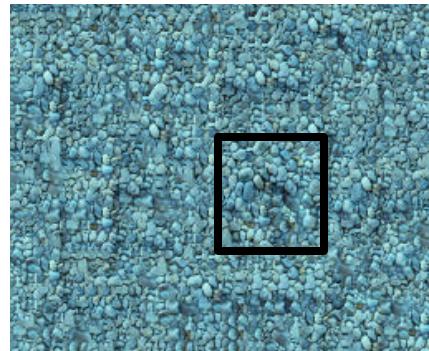


*Paul Viola*

**Related to the MAR models of Willsky et. al.** *MIT AI Lab*



Original  
Texture

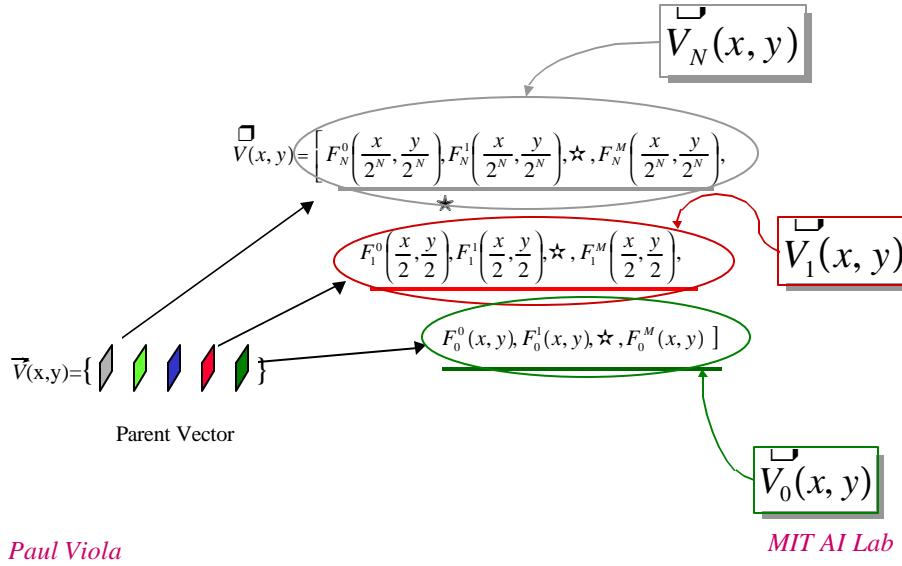


Synthesis Results

*Paul Viola*

*MIT AI Lab*

## Multi-resolution Parent Vector



## Probabilistic Model

$$P(\vec{V}(x,y))$$

Markov

$$\begin{aligned} P(V_l(x,y) | \{WI\} - V_l(x,y)) \\ = P(V_l(x,y) | V_{l+1}(x,y), V_{l+2}(x,y) \dots) \end{aligned}$$

Conditionally  
Independent

$$P(V_l) = \prod_{x,y} P(V_l(x,y) | V_{l+1}(x,y), V_{l+2}(x,y) \dots)$$

Successive  
Conditioning

$$\begin{aligned} P(I) = P(WI) &= P(V_M) \times P(V_{M-1} | V_M) \\ &\quad \times P(V_{M-2} | V_M, V_{M-1}) \\ &\quad \times P(V_{M-3} | V_M, V_{M-1}, V_{M-2}) \dots \end{aligned}$$

*Paul Viola*

*MIT AI Lab*

## *Estimating Conditional Distributions*

- Non-parametrically

$$P^*(x) = \sum_i R(x - x_i)$$

$$\begin{aligned} & P(V_l(x, y) | V_{l+1}(x, y), V_{l+2}(x, y) \dots ) \\ &= \frac{P(V_l(x, y), V_{l+1}(x, y), V_{l+2}(x, y) \dots )}{P(V_{l+1}(x, y), V_{l+2}(x, y) \dots )} \\ &\cong \frac{P^*(V_l(x, y), V_{l+1}(x, y), V_{l+2}(x, y) \dots )}{P^*(V_{l+1}(x, y), V_{l+2}(x, y) \dots )} \end{aligned}$$

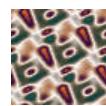
*Paul Viola*

*MIT AI Lab*

## *Shannon Resampling on a Tree*

*Step 1: Build analysis pyramid*

64x64



2x2



Input  
Image

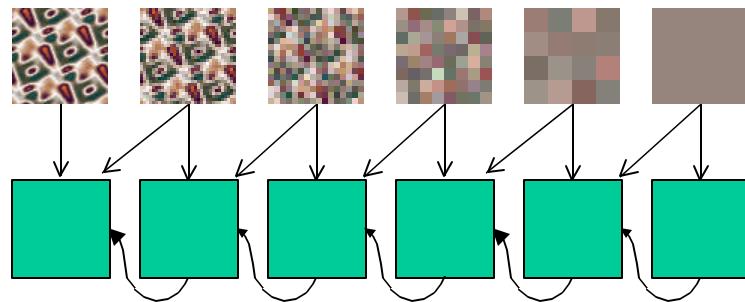
Note: We are using only the Gaussian pyramid here!

*Paul Viola*

Normally we use an oriented pyramid...

*MIT AI Lab*

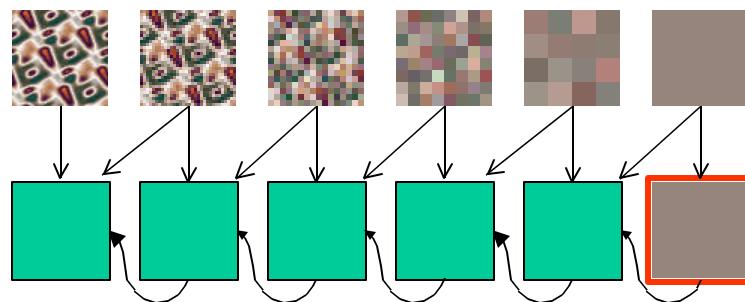
*Shannon Resampling*  
*Step 2: Build synthesis pyramid*



*Paul Viola*

*MIT AI Lab*

*Shannon Resampling*  
*Step 2a: Fill in the top...*

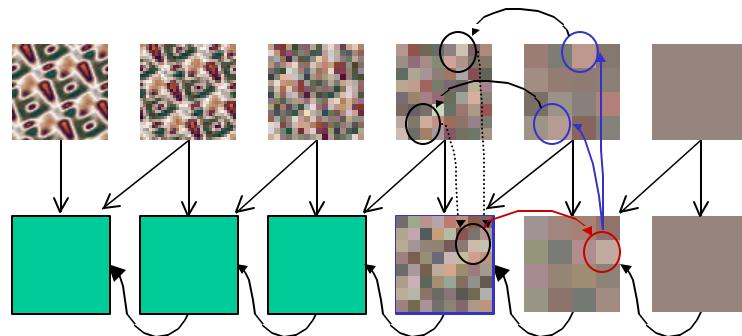


Pixels are generated by sampling  
from the analysis pyramid.

*Paul Viola*

*MIT AI Lab*

*Shannon Resampling*  
*Step 2b: Fill in subsequent levels*

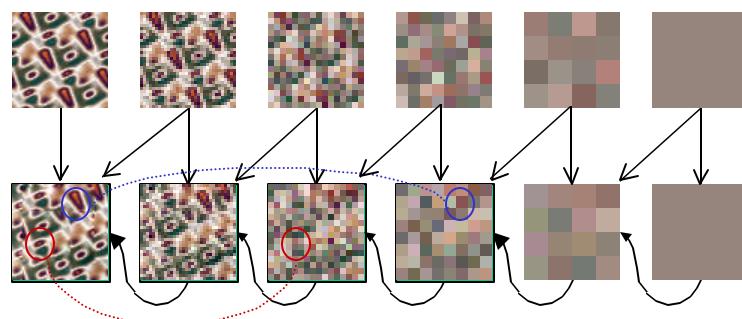


*Paul Viola*

Pixels are generated by  
*conditional sampling*  
(dependent on the parent).

*MIT AI Lab*

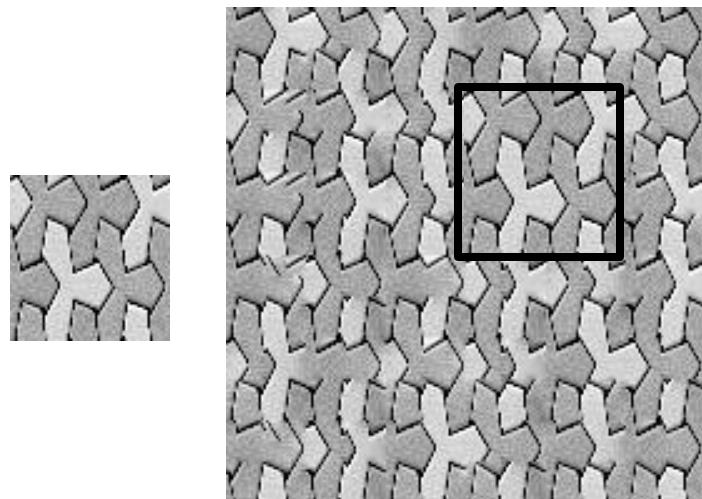
*Shannon Resampling*  
*Finish the pyramid*



*Paul Viola*

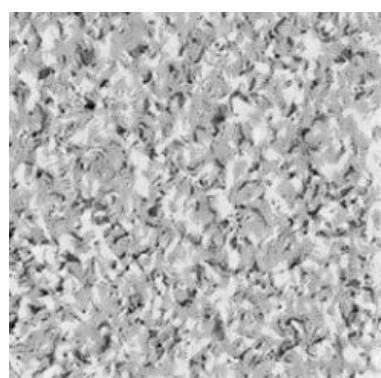
Decisions made at low resolutions  
generate discrete features in the final image.

*MIT AI Lab*



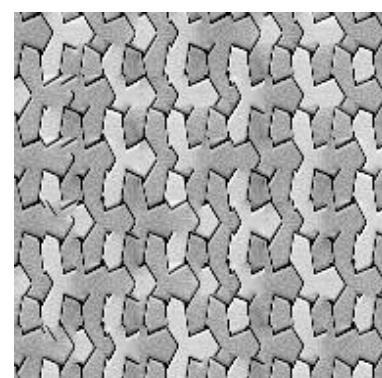
*Paul Viola*

*MIT AI Lab*



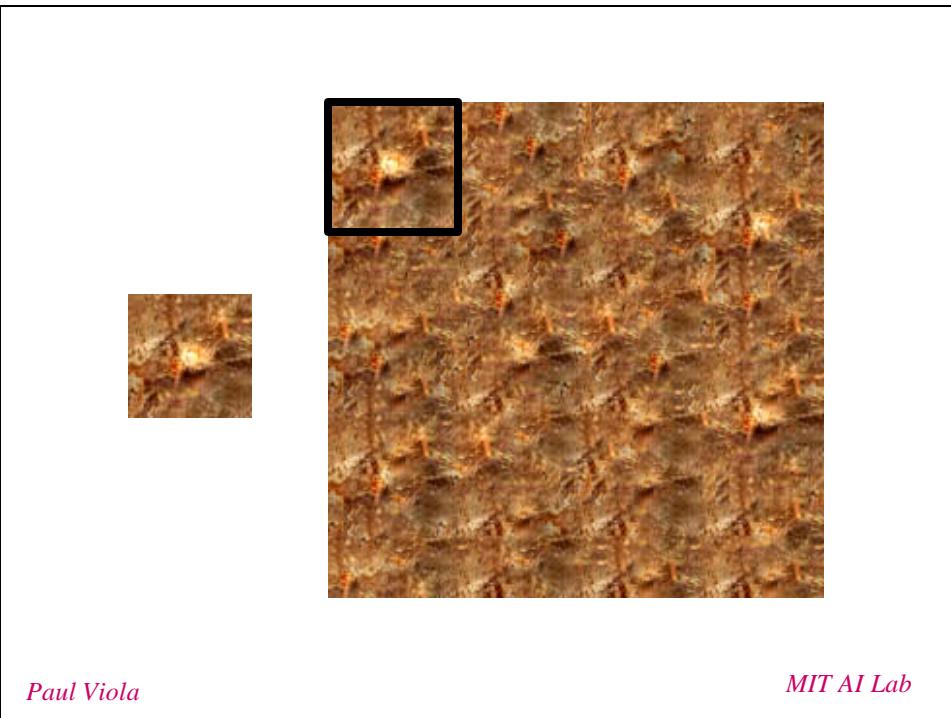
**B & H**

*Paul Viola*



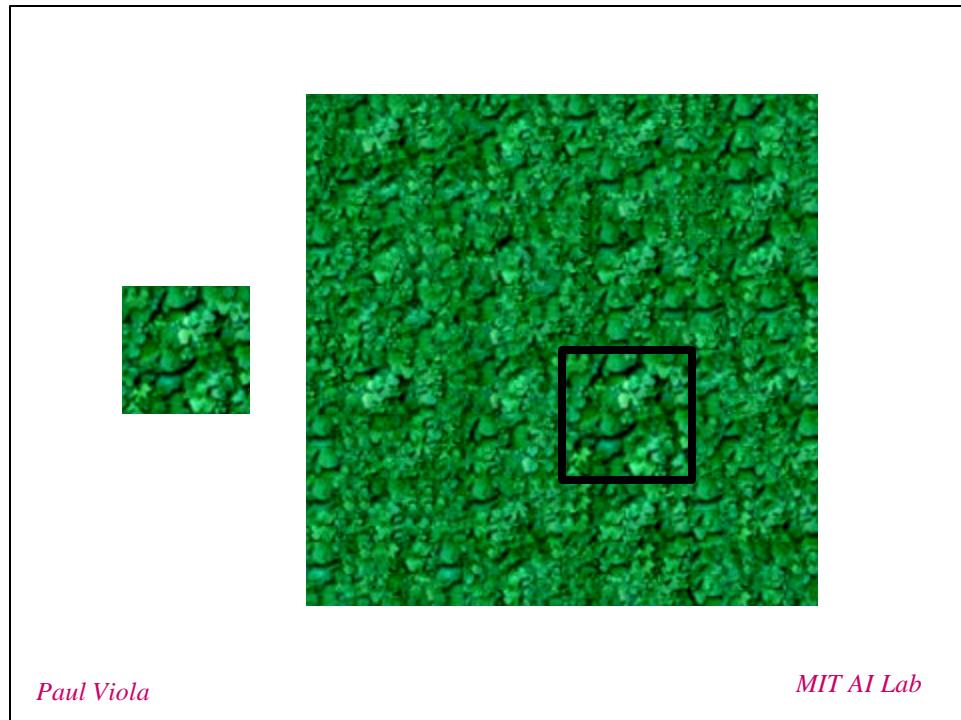
**D & V**

*MIT AI Lab*



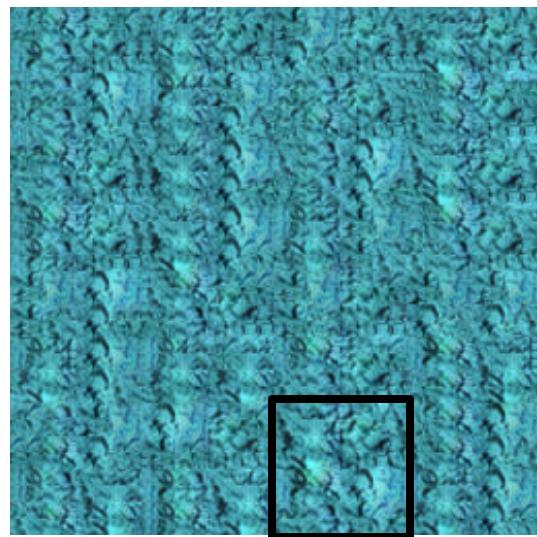
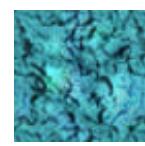
*Paul Viola*

*MIT AI Lab*



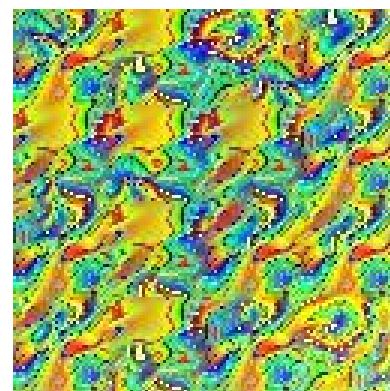
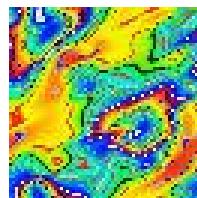
*Paul Viola*

*MIT AI Lab*



*Paul Viola*

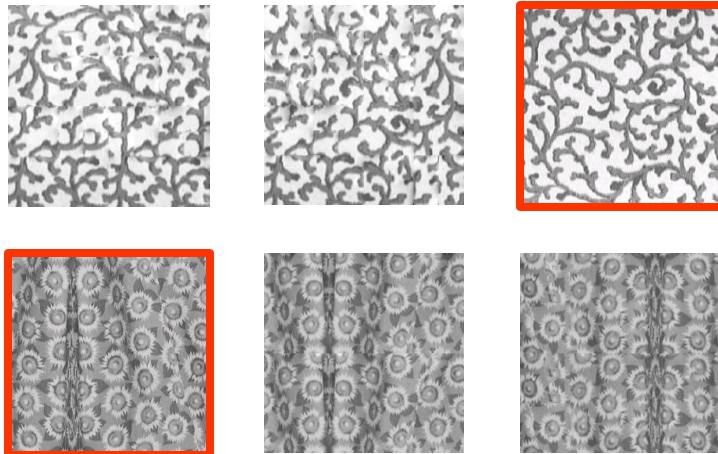
*MIT AI Lab*



*Paul Viola*

*MIT AI Lab*

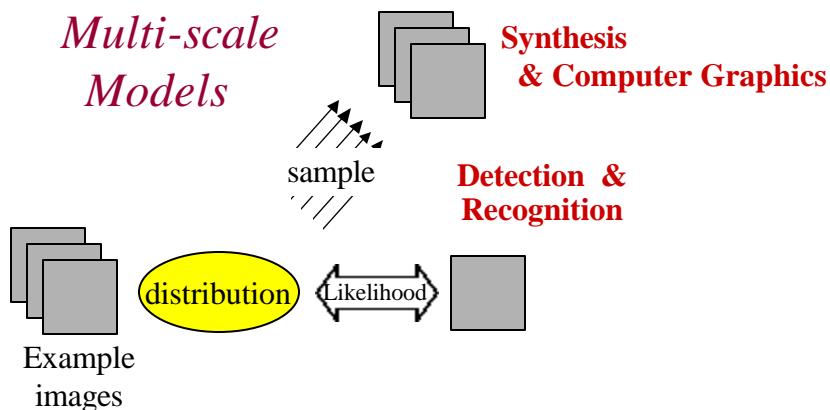
## *FRAME: Challenge*



*Paul Viola*

*MIT AI Lab*

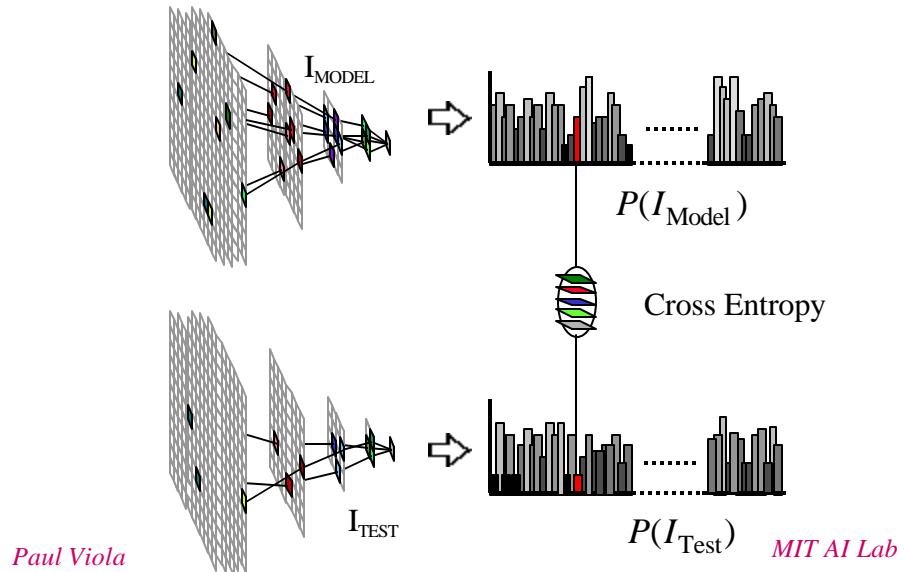
## *Non-Parametric Multi-scale Models*



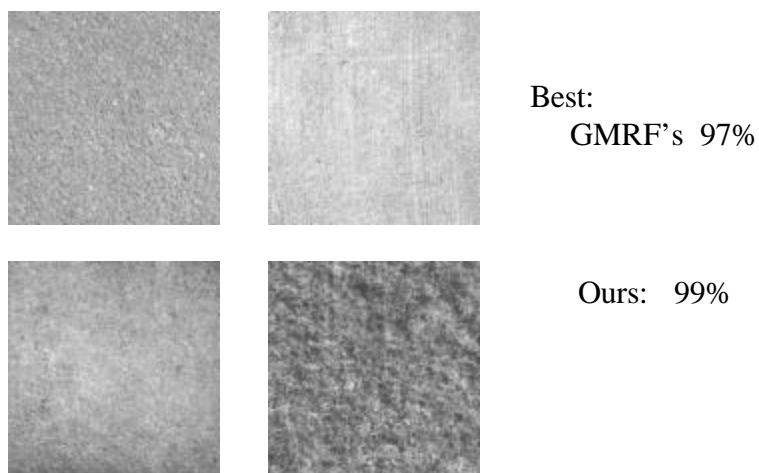
*Paul Viola*

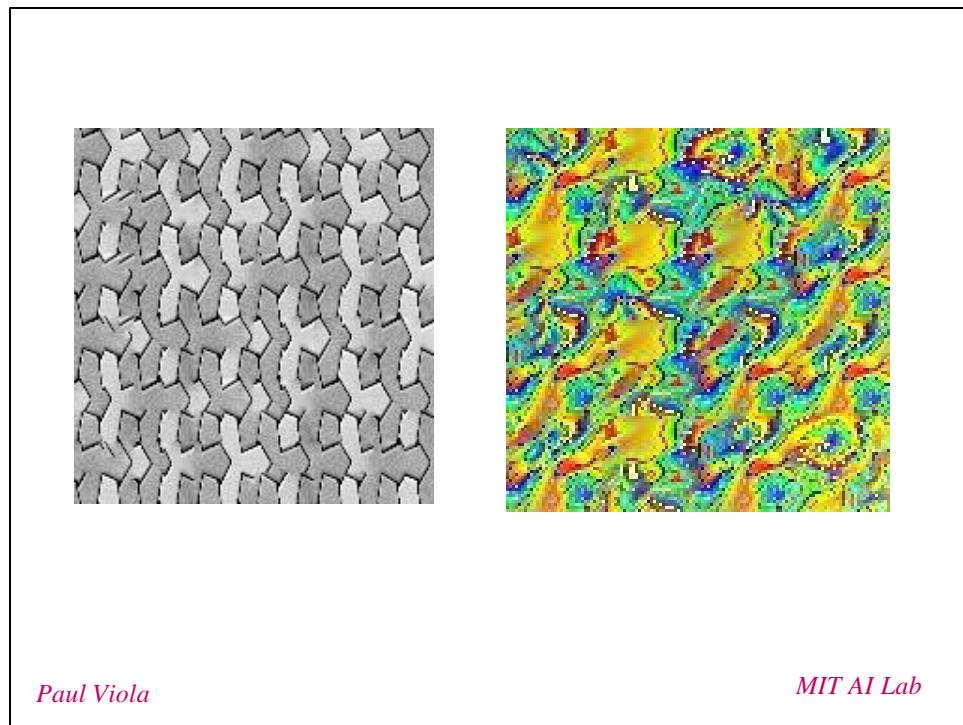
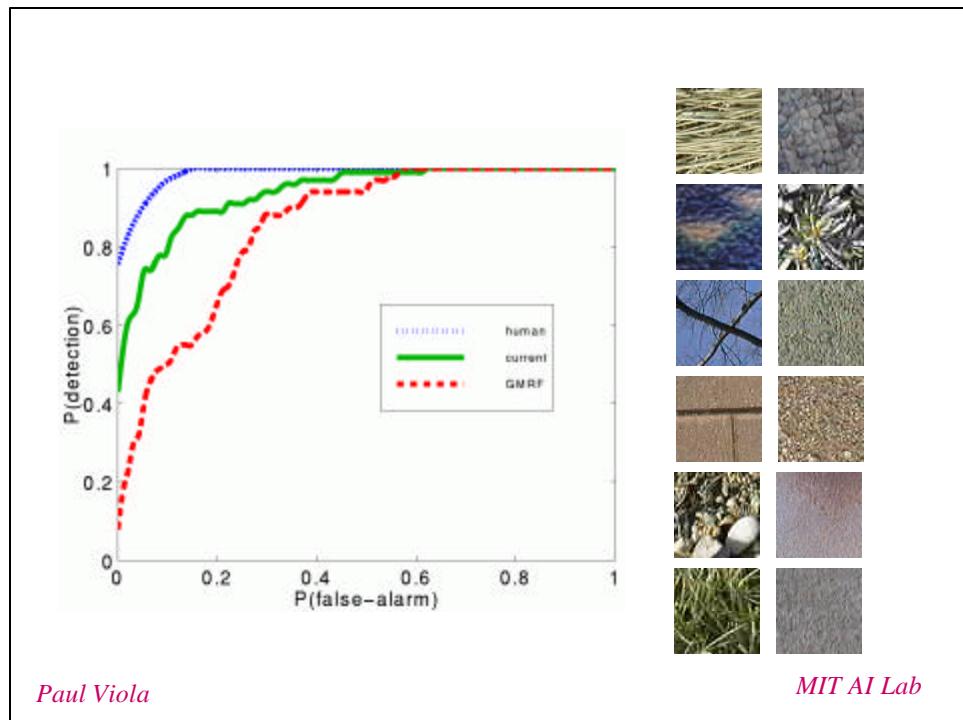
*MIT AI Lab*

## *Discrimination via Cross Entropy*



## *Meastex: Texture Classification*





## *Where is the boundary between texture and objects?*

- Our model can synthesize and recognize complex and structured textures.
  - Far beyond previous older definitions of texture.
- Where is the boundary between these complex textures and other patterns in images
  - Like faces, human forms, automobiles, etc?
- Only experiments can tell...

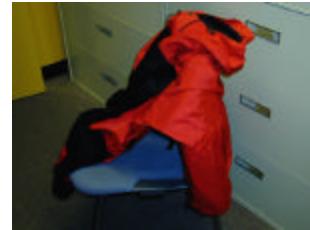
*Paul Viola*

*MIT AI Lab*

## *The Jacket Hypothesis*



*Paul Viola*



*AI Lab*

## *What about face detection?*

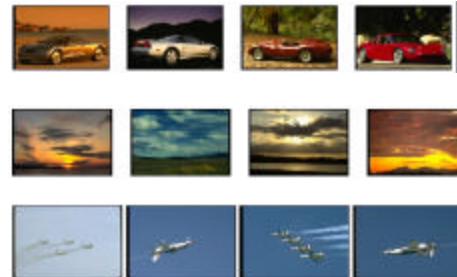


- Synthesis is convincing
- Train a texture model to detect faces

*Paul Viola*

Tom Rikert & *MIT AI Lab*

## *Detecting Objects*



- *Key Difficulties:*
  - Variation in Pose, Deformation, & variation across class
- Most Object Recognition approaches are either:
  - Very dependent on precise shape and size
  - Entirely dependent on simple features (... color, edge histograms)
- *Hypothesis:*
  - Object recognition is closely related to texture perception

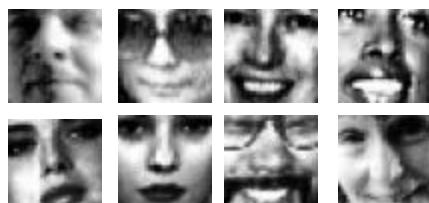
*Paul Viola*

*MIT AI Lab*

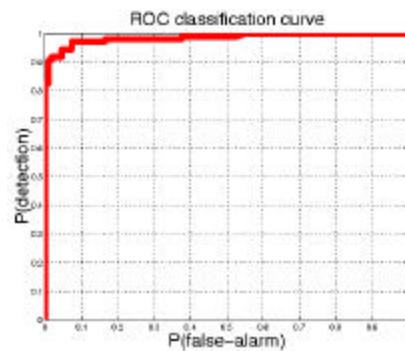
## *Detection Results*



Non-face test images

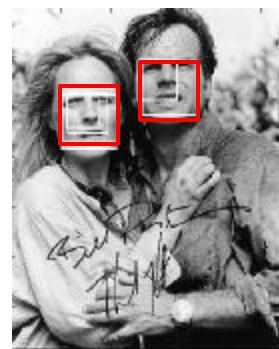
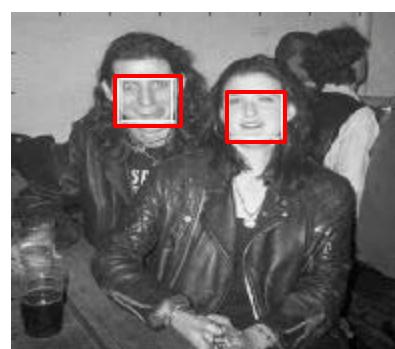


*Paul Viola*  
Web face test images



MIT AI Lab

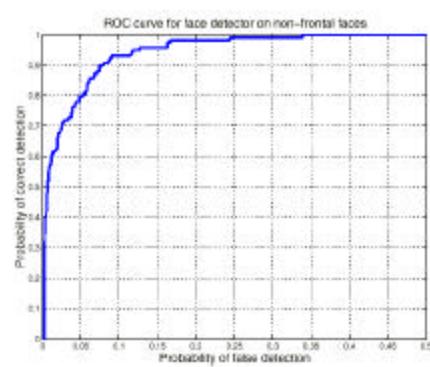
## *Detection Results:*



*Paul Viola*

MIT AI Lab

## *Non-frontal faces*

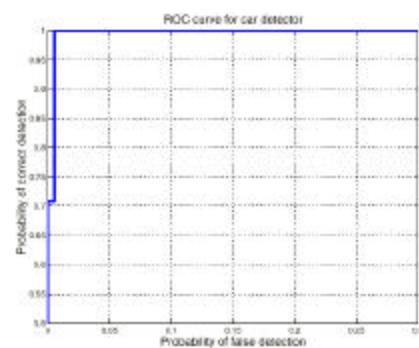


But naïve detection  
is expensive

*Paul Viola*

*MIT AI Lab*

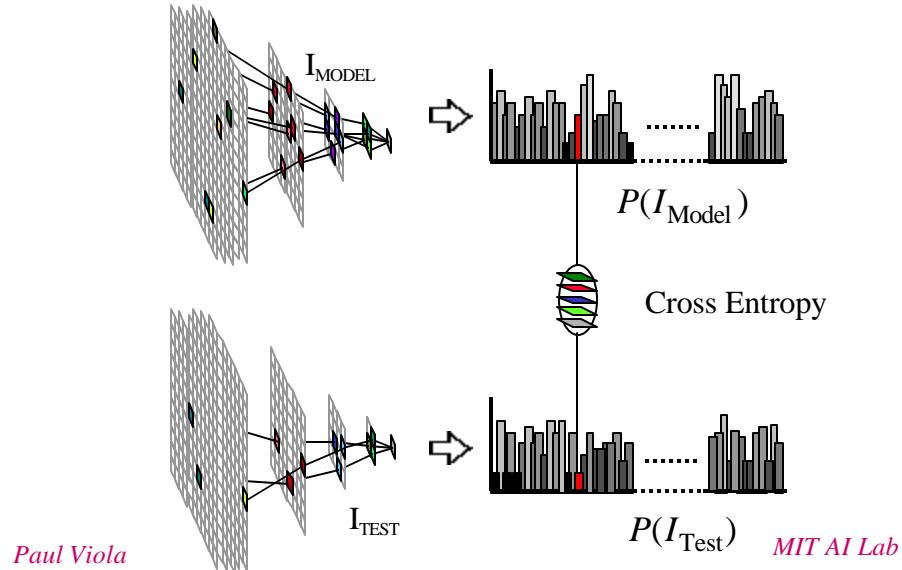
## *Car Images*



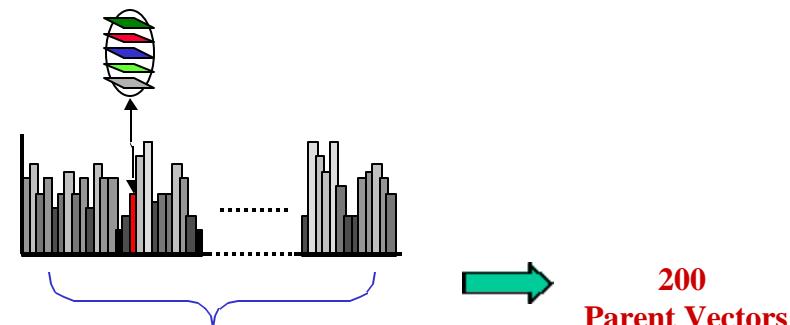
*Paul Viola*

*MIT AI Lab*

## *Texture recognition via Cross Entropy*



## *Pruning the density estimator*

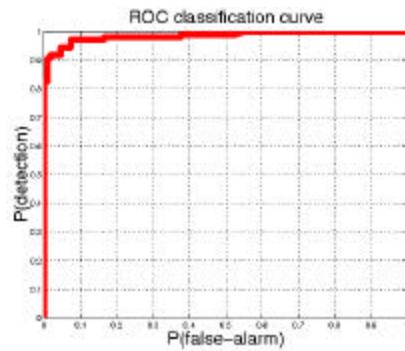


**Result:** Detection/Classification is **faster** than template correlation

*Paul Viola*

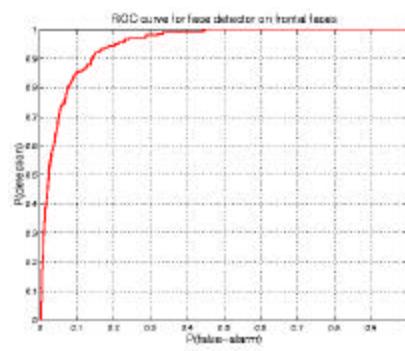
*MIT AI Lab*

*ROC using 200 vectors...*



2000 Vector Model

*Paul Viola*



200 Vector Model

*MIT AI Lab*

*Scanning results:*



Time: 9 secs



*Paul Viola*

*MIT AI Lab*

## *Key facial features*

- determined automatically
- located automatically



Multi-scale features which are come  
from the face model can be automatically  
detected for many individuals

*Paul Viola*

*MIT AI Lab*



*Paul Viola*

*b*



*Pa... 1996*

*b*



*Pa... 1996*

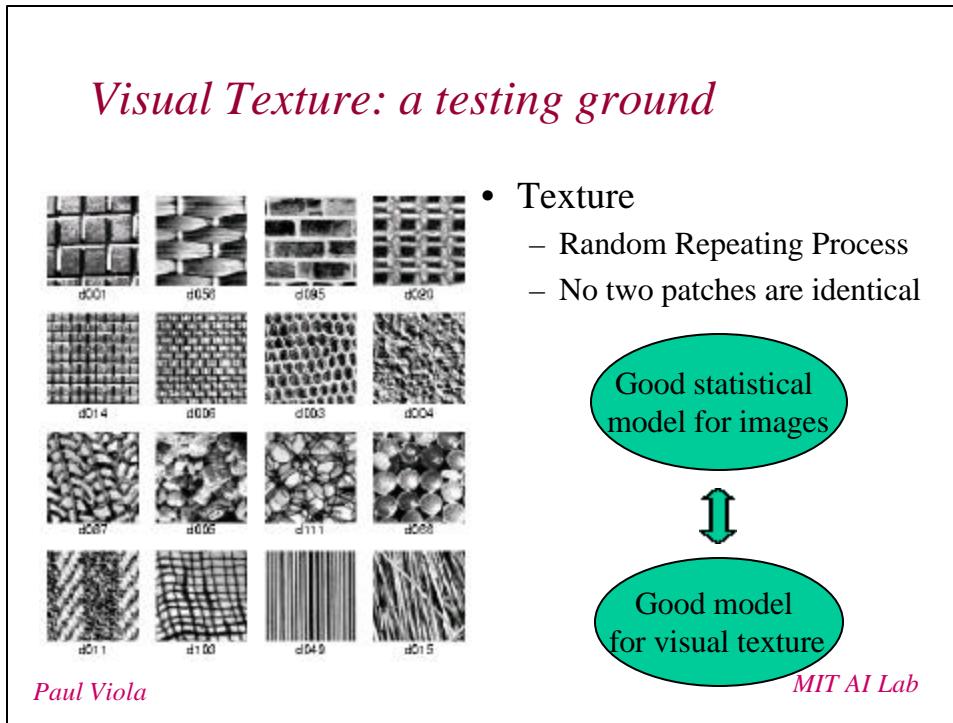
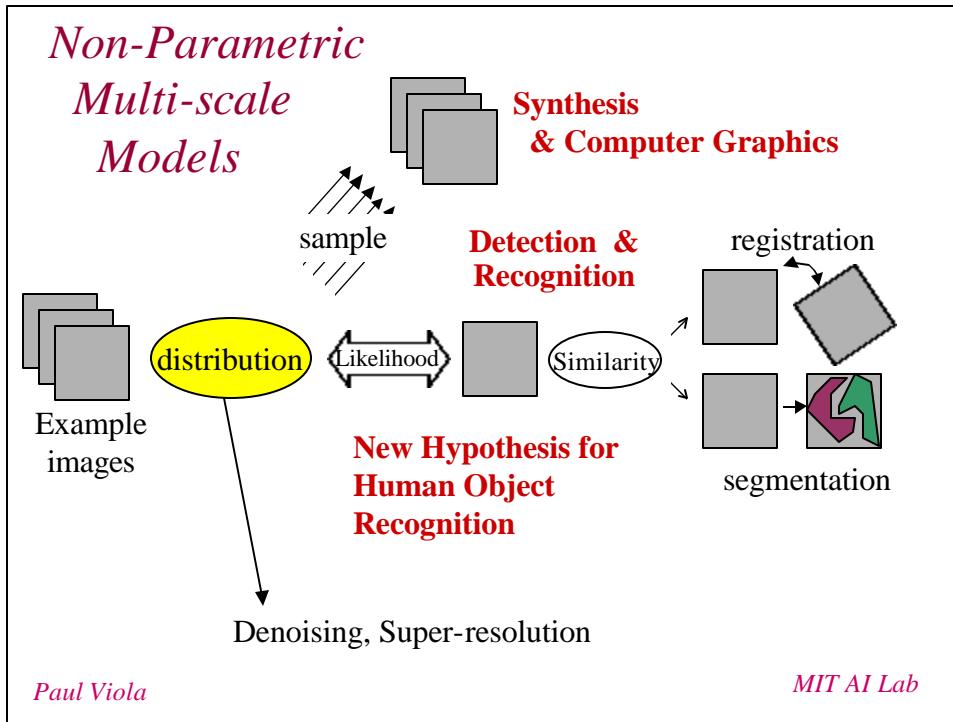
*b*

*Future Work:*  
*New Face Recognition Algorithm*

- Facial identity depends both on the types of features and their location.

*Paul Viola*

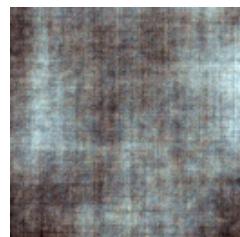
*MIT AI Lab*



*Generation a critical test*



Input  
Texture



Gaussian



Independent



Non-parametric  
Multi-scale

*MIT AI Lab*

*Paul Viola*

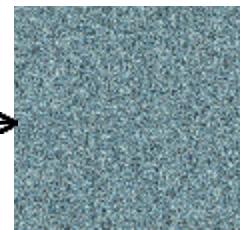
*Simple Statistical Models*

*Independent pixels*



Histogram

$$P(I) = \prod_{x,y} P(I_{xy})$$



*Paul Viola*

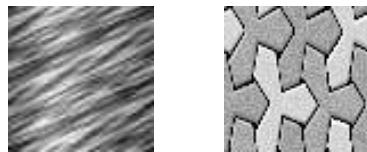
*MIT AI Lab*

## Statistical Model 2: Gaussian Distribution

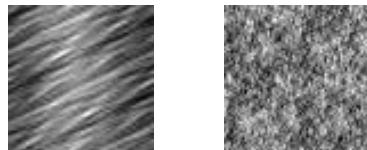
$$P(I) = N(I, \mathbf{m}, \Sigma)$$

$$\propto e^{-\frac{1}{2}(\mathbf{I}-\mathbf{m})^T \Sigma^{-1} (\mathbf{I}-\mathbf{m})}$$

Original



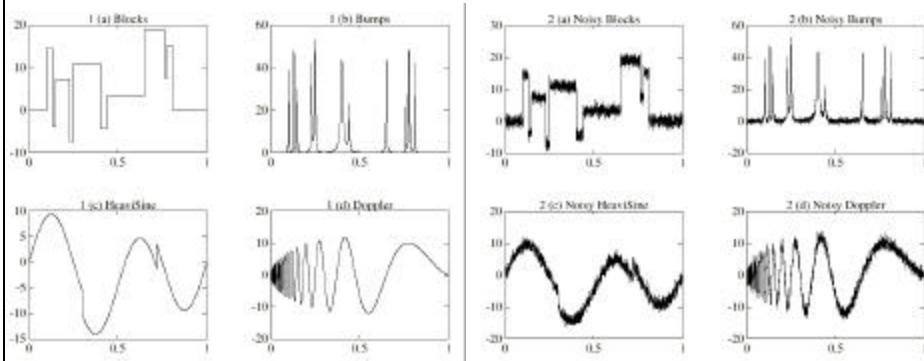
Generated



*Paul Viola*

*MIT AI Lab*

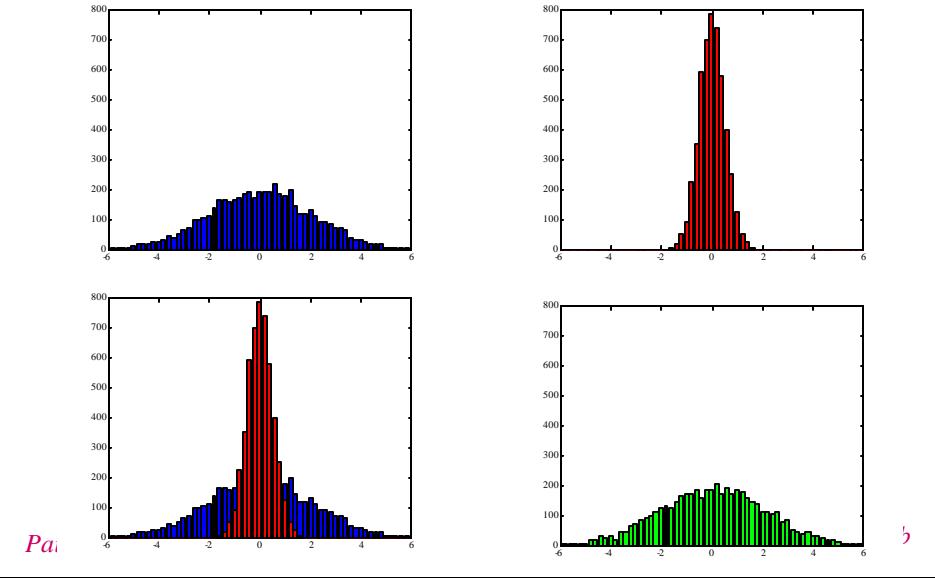
## Signals plus noise...



*Paul Viola*

*MIT AI Lab*

## Noise + Signal: Two Gaussian Case



$$E[I | \hat{I}] = \int \frac{I P(\mathbf{h} = \hat{I} - I) P(I)}{P(\hat{I})} dI$$

$$= \int \frac{I e^{\frac{-(\hat{I}-I)^2}{2n^2}} e^{\frac{-(I)^2}{2s^2}}}{c} dI$$

$$E[I | \hat{I}] = \frac{s^2}{n^2 + s^2} \hat{I}$$

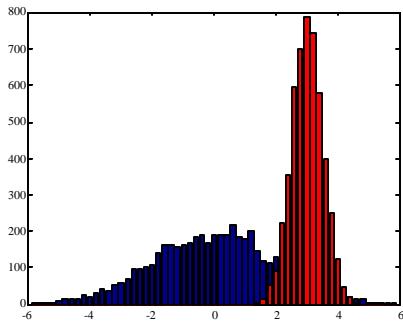
$$\frac{-(\hat{I}-I)^2}{2n^2} - \frac{(I)^2}{2s^2} = - \left[ \frac{s^2(I^2 - 2I\hat{I} + \hat{I}^2) + n^2I^2}{2n^2s^2} \right]$$

$$= - \left[ \frac{(s^2 + n^2)I^2 - 2s^2I\hat{I} + s^2\hat{I}^2}{2n^2s^2} \right]$$

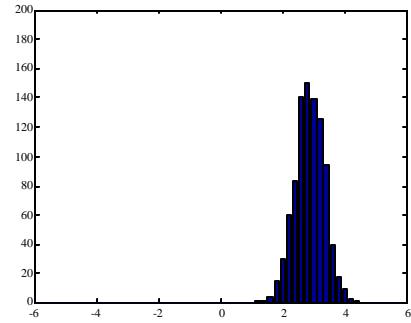
$$= - \left[ \frac{I^2 - \frac{2s^2I\hat{I}}{(s^2 + n^2)} + \frac{s^2\hat{I}^2}{(s^2 + n^2)}}{2n^2s^2} \right]$$

Paul Viola

*In pictures...*



**Observation = 3.0**



**Mean = 2.8**

*Paul Viola*

*MIT AI Lab*

### *Statistical Model 3: Independent Wavelet Models*

- Donoho, Adelson, Simoncelli, etc.
- Very efficient (linear time)
  - Estimation, Sampling, Inference

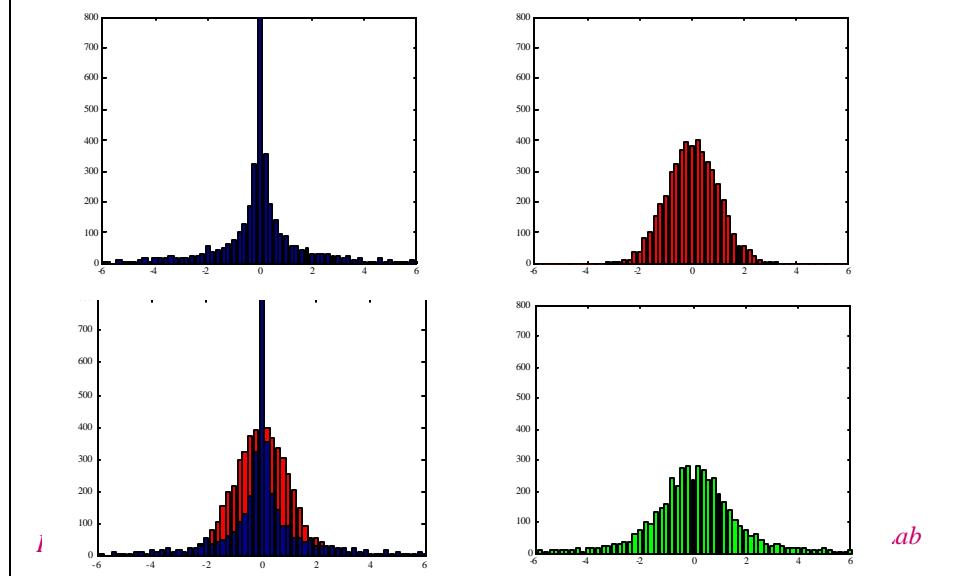
$$P(I) \propto \prod_j P_j([WI]_j)$$

- $P(I)$  is defined implicitly
  - As a distribution over the features present in an image
- $W$  is a Wavelet or tight frame operator
  - Invertibility is key...

*Paul Viola*

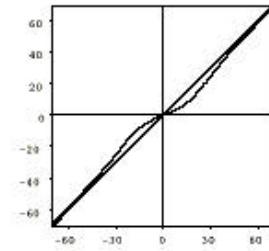
*MIT AI Lab*

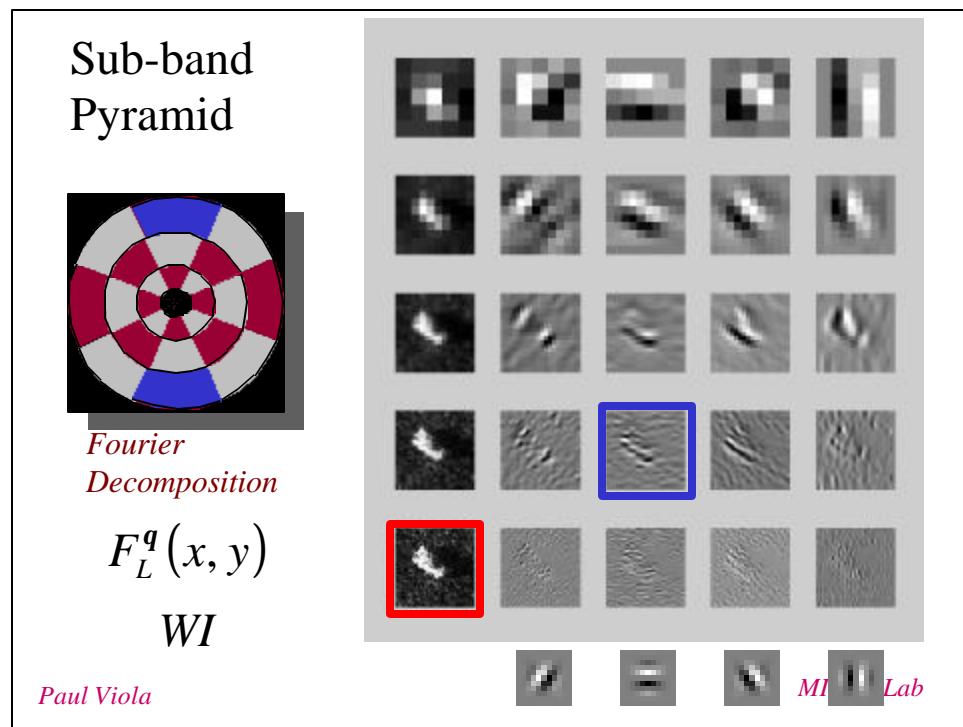
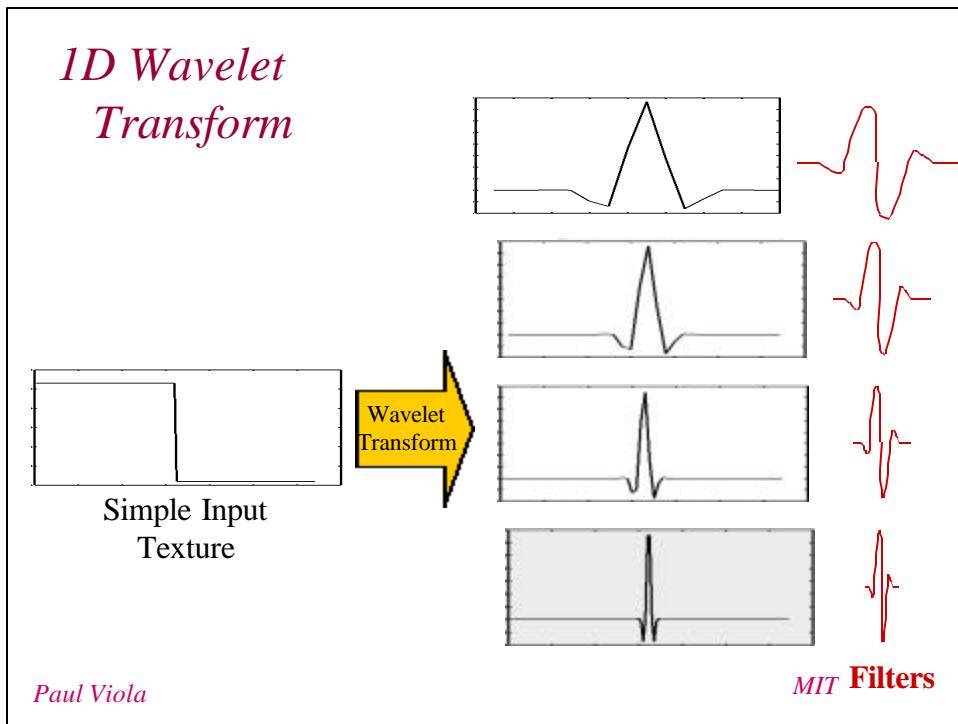
## Noise vs. Signal: The details



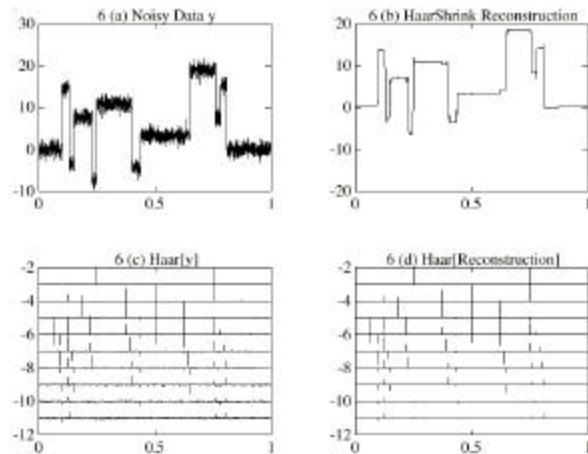
*Non-gaussian:  
Integral is evaluated numerically*

$$E[I | \hat{I}] = \int \frac{I P(\mathbf{h} = \hat{I} - I) P(I)}{P(\hat{I})} dI$$





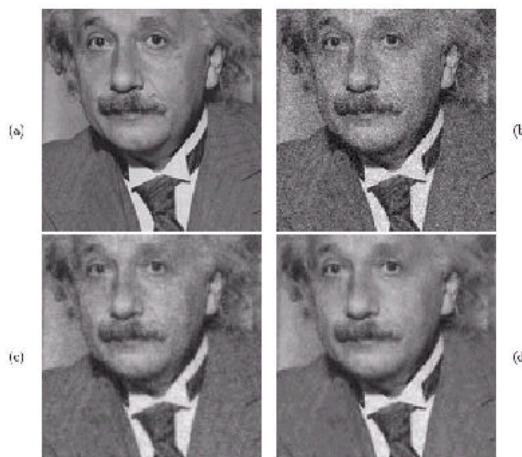
## Noise removal through shrinkage



Paul Viola

MIT AI Lab

## Removing noise from images



Paul Viola

MIT AI Lab

## *Independent Wavelet Synthesis Model*

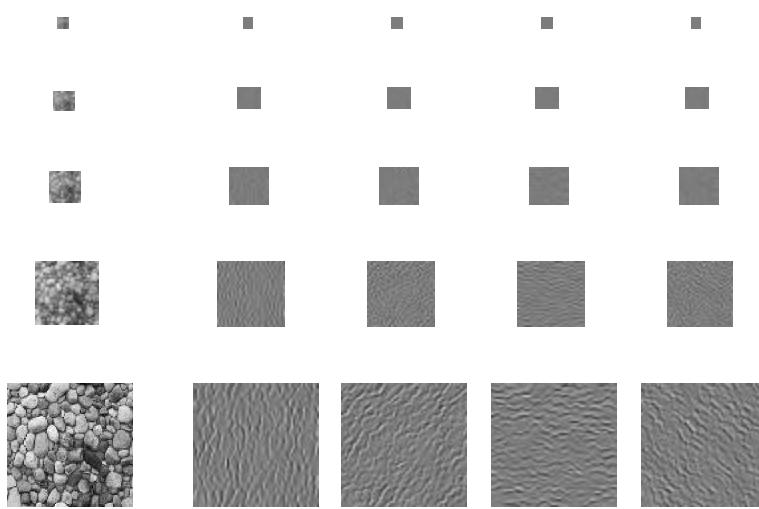
$$\begin{aligned} P(I) &\approx \prod_{l,\mathbf{q},x,y} P_{l,\mathbf{q},x,y}(F_l^{\mathbf{J}}(x, y)) \\ &\approx \prod_{l,\mathbf{q},x,y} P_{l,\mathbf{q}}(F_l^{\mathbf{J}}(x, y)) \end{aligned}$$

*Given :  $I, W$*   
*Observe :  $O_{l,q} = \{F_l^q(x, y)\}$*   
*Model :  $P_{l,q}(\cdot)$*

*Paul Viola*

*MIT AI Lab*

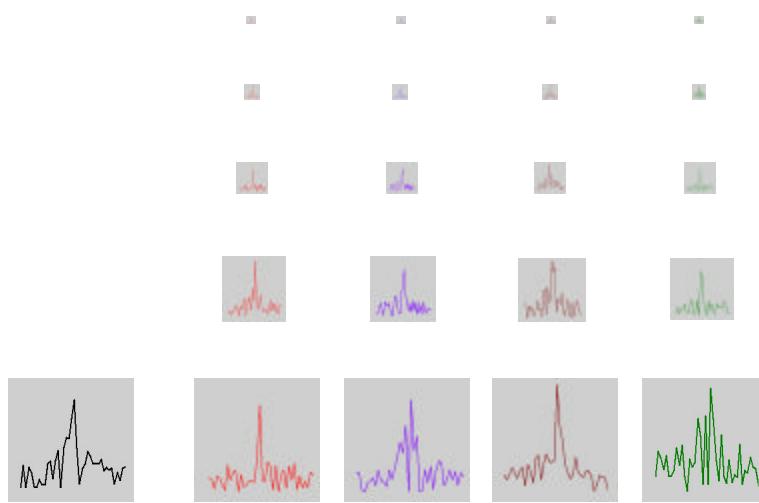
## Observe Coefficients



*Paul Viola*

*MIT AI Lab*

## Compute Histograms



*Paul Viola*

*MIT AI Lab*

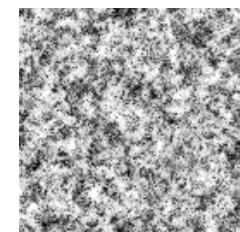
## Multi-scale Histograms

original  
texture patch



*Paul Viola*

## Multi-scale Sampling

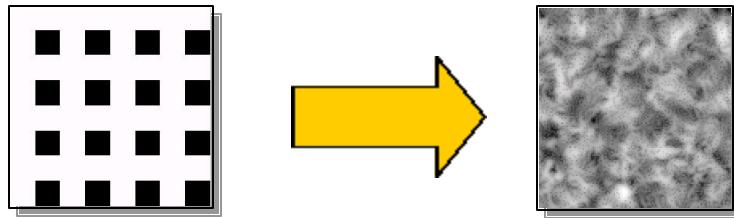


*MIT AI Lab*

Sampling Procedure

synthesized  
texture patch

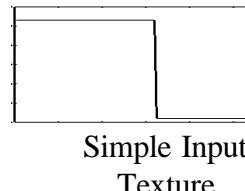
*Not quite right...*



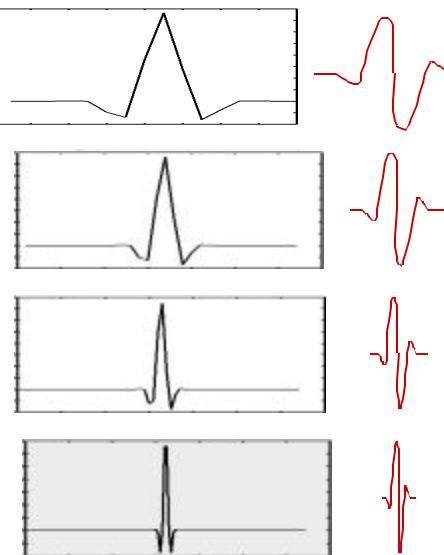
*Paul Viola*

*MIT AI Lab*

*Edges lead to aligned  
coefficients*



Wavelet  
Transform

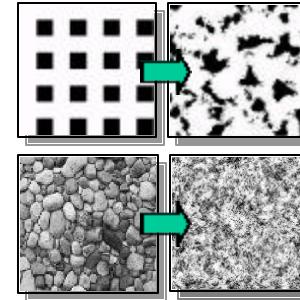
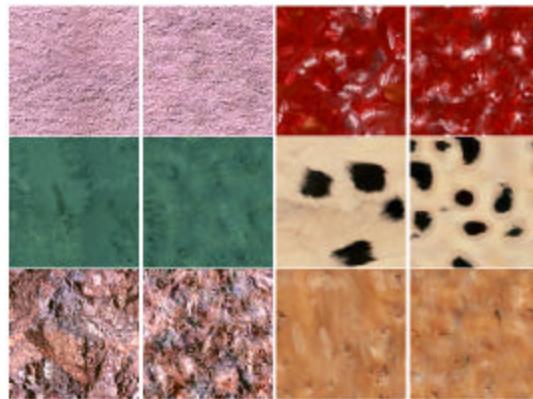


Simple Input  
Texture

*MIT Filters*

*Paul Viola*

## *Heeger and Bergen: Constrain the pixel histogram*



Models of structured images are weak.

*Paul Viola*

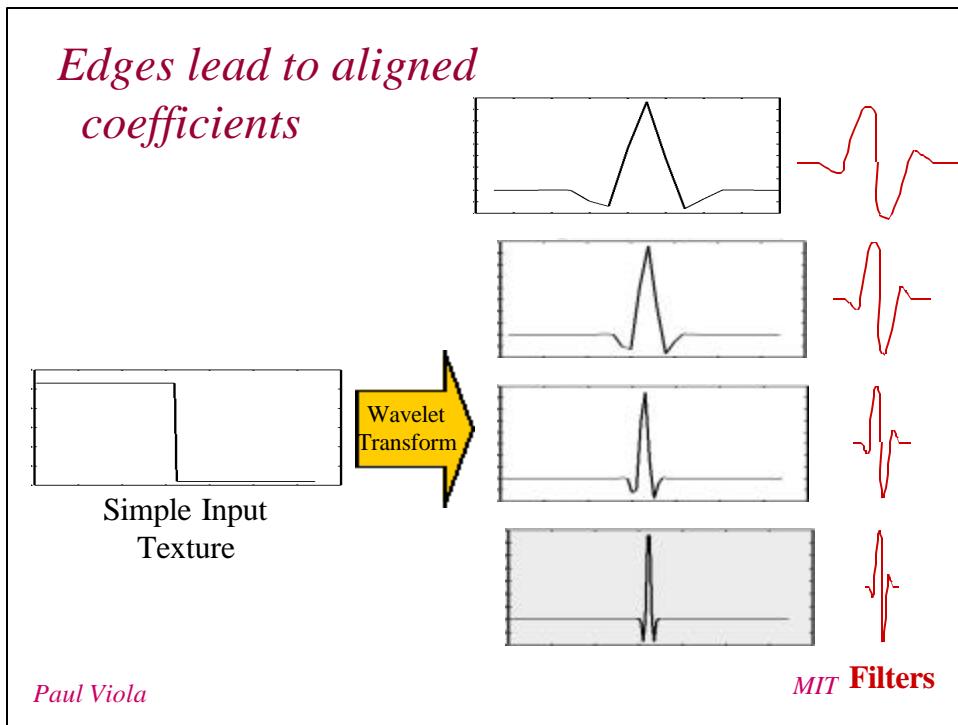
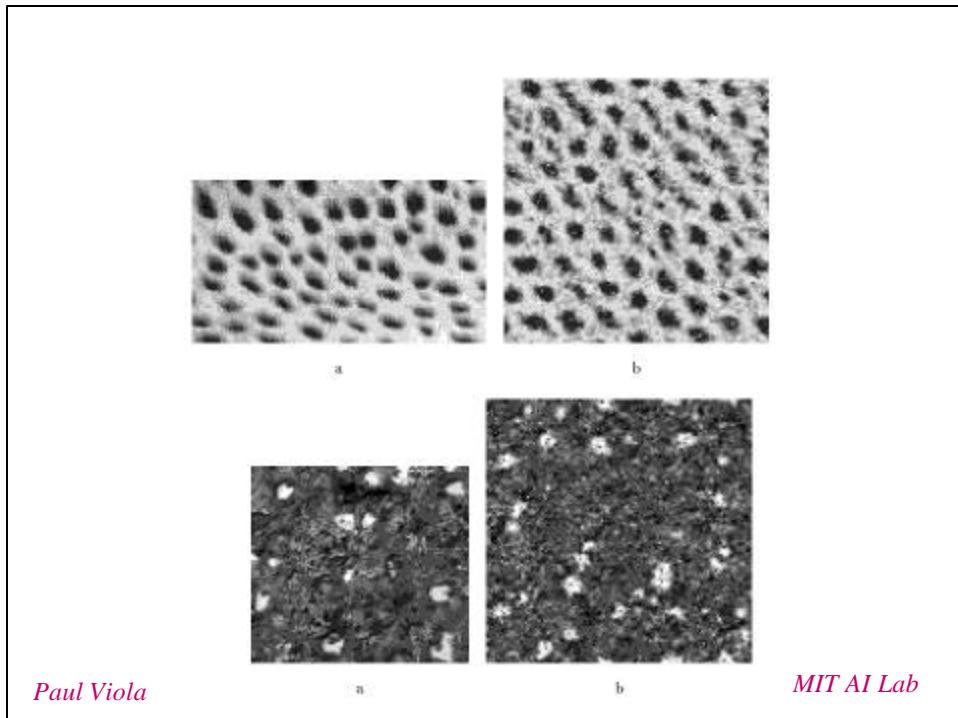
*MIT AI Lab*

## *FRAME: a generalization of B&H (Zu, Wu and Mumford)*

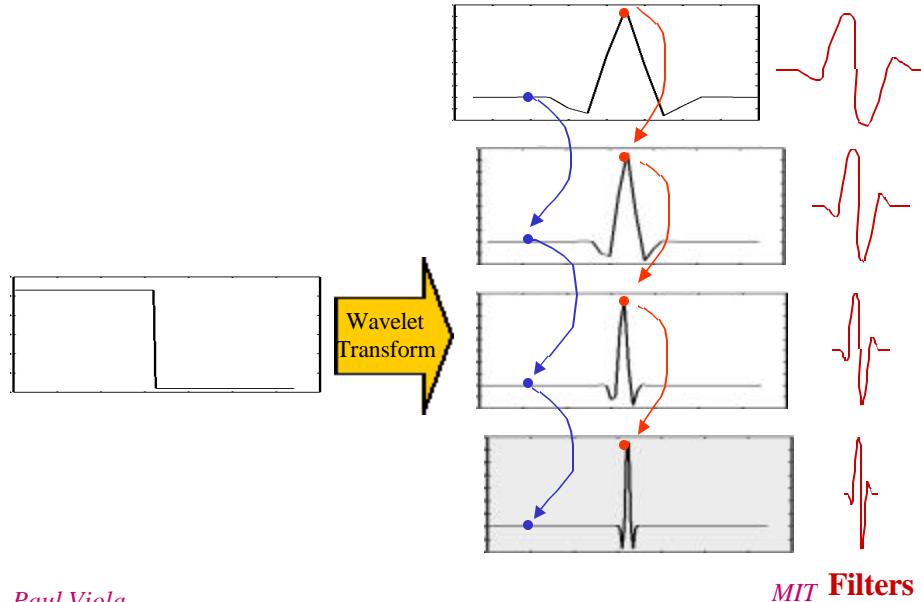
- Specify a set of filters
  - Not necessarily orthogonal or even linear.
- Measure the histogram of these filters
  - Type of statistic
- Construct a Boltzmann/Gibbs distribution which generates these statistics
  - Maximum Entropy
- Resulting algorithm is currently intractible
  - Days to generate a single image

*Paul Viola*

*MIT AI Lab*



## *Preserving Cross Scale Alignment*

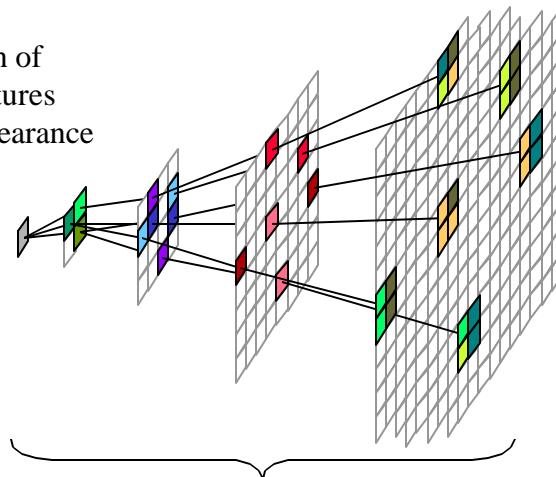


*Paul Viola*

*MIT Filters*

## *Statistical Distribution of Multi-scale Features*

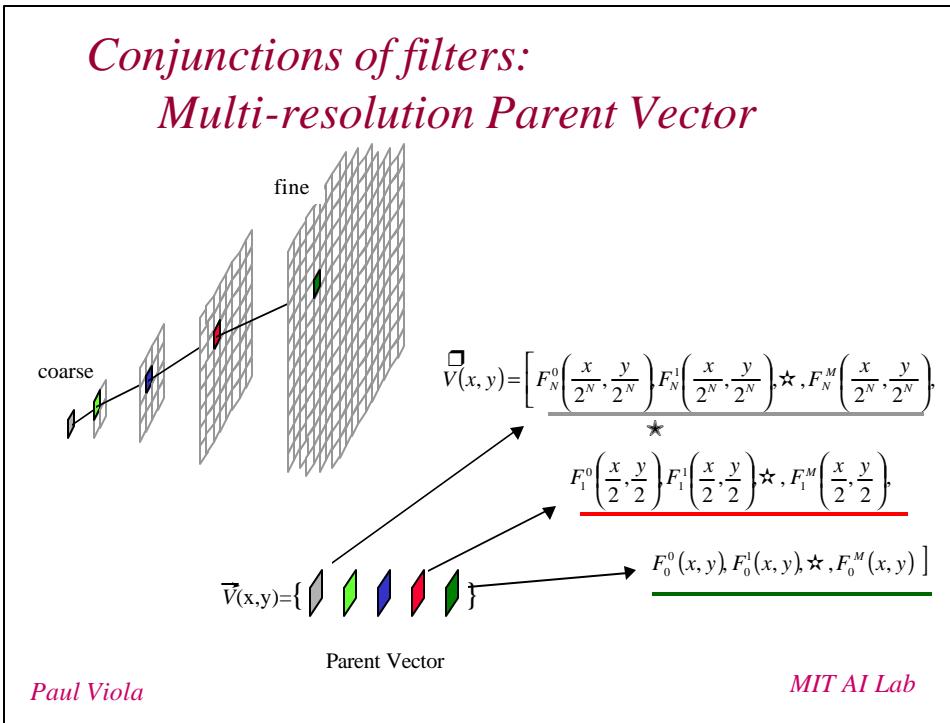
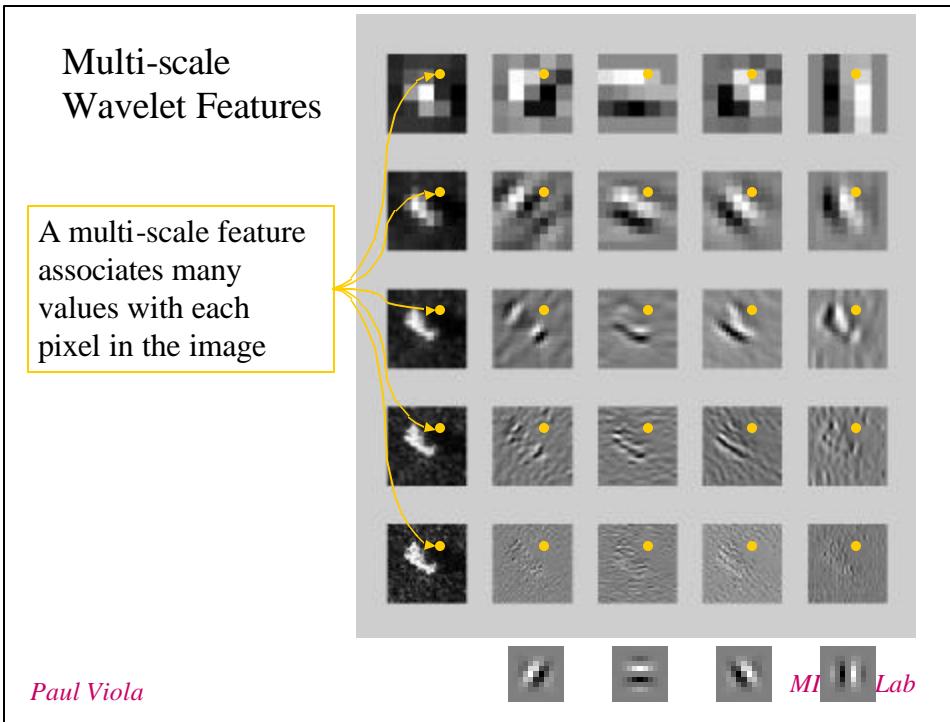
The distribution of multi-scale features determines appearance



*Paul Viola*

*Wavelet Pyramid*

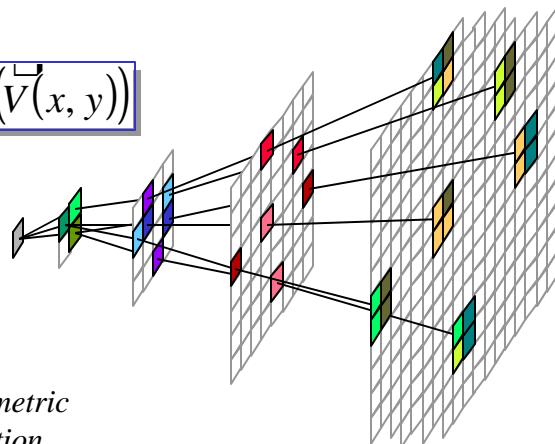
*MIT AI Lab*



## *Build a Model for Observed Distribution*

$$P(I) = P(V(x, y))$$

*Non-parametric  
Distribution*

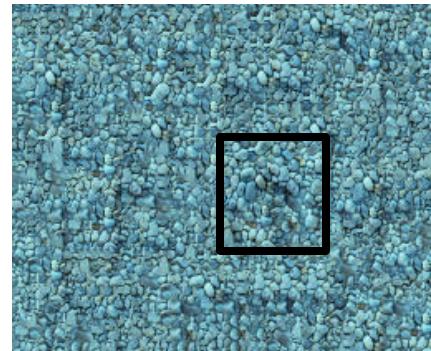


*Paul Viola*

**Related to the MAR models of Willsky et. al.** *MIT AI Lab*



Original  
Texture

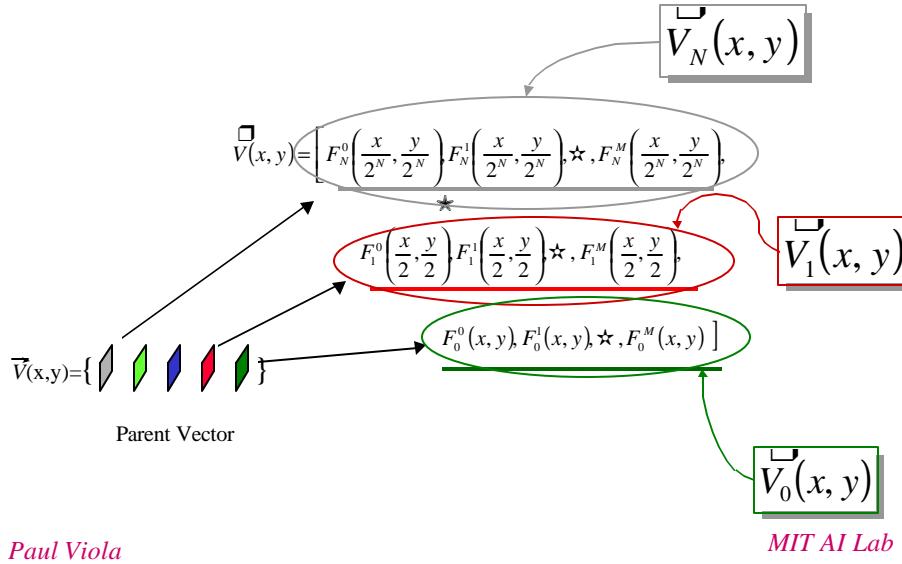


Synthesis Results

*Paul Viola*

*MIT AI Lab*

## Multi-resolution Parent Vector



## Probabilistic Model

$$P(V(x,y))$$

Markov

$$\begin{aligned} P(V_l(x,y) | \{WI\} - V_l(x,y)) \\ = P(V_l(x,y) | V_{l+1}(x,y), V_{l+2}(x,y) \dots) \end{aligned}$$

Conditionally  
Independent

$$P(V_l) = \prod_{x,y} P(V_l(x,y) | V_{l+1}(x,y), V_{l+2}(x,y) \dots)$$

Successive  
Conditioning

$$\begin{aligned} P(I) = P(WI) &= P(V_M) \times P(V_{M-1} | V_M) \\ &\quad \times P(V_{M-2} | V_M, V_{M-1}) \\ &\quad \times P(V_{M-3} | V_M, V_{M-1}, V_{M-2}) \dots \end{aligned}$$

*Paul Viola*

*MIT AI Lab*

## *Estimating Conditional Distributions*

- Non-parametrically

$$P^*(x) = \sum_i R(x - x_i)$$

$$\begin{aligned} & P(V_l(x, y) | V_{l+1}(x, y), V_{l+2}(x, y) \dots ) \\ &= \frac{P(V_l(x, y), V_{l+1}(x, y), V_{l+2}(x, y) \dots )}{P(V_{l+1}(x, y), V_{l+2}(x, y) \dots )} \\ &\cong \frac{P^*(V_l(x, y), V_{l+1}(x, y), V_{l+2}(x, y) \dots )}{P^*(V_{l+1}(x, y), V_{l+2}(x, y) \dots )} \end{aligned}$$

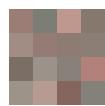
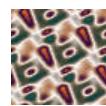
*Paul Viola*

*MIT AI Lab*

## *Shannon Resampling on a Tree*

*Step 1: Build analysis pyramid*

64x64



2x2



Input  
Image

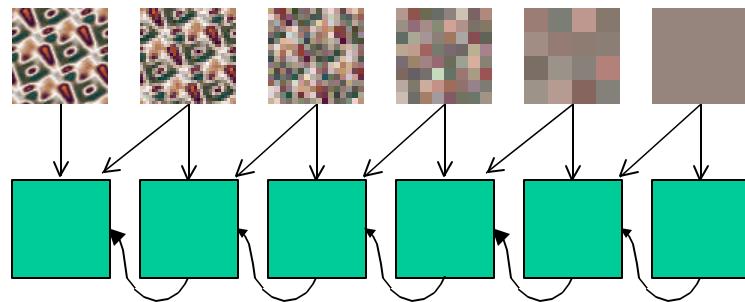
Note: We are using only the Gaussian pyramid here!

*Paul Viola*

Normally we use an oriented pyramid...

*MIT AI Lab*

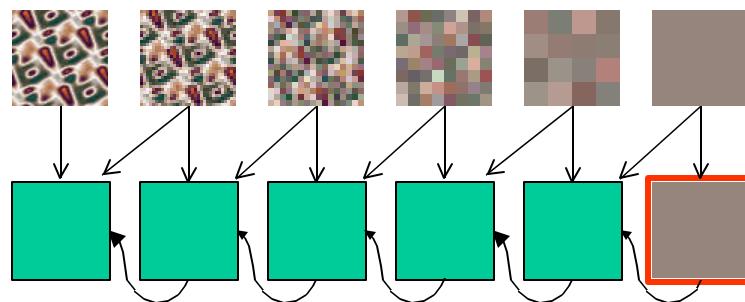
*Shannon Resampling*  
*Step 2: Build synthesis pyramid*



*Paul Viola*

*MIT AI Lab*

*Shannon Resampling*  
*Step 2a: Fill in the top...*



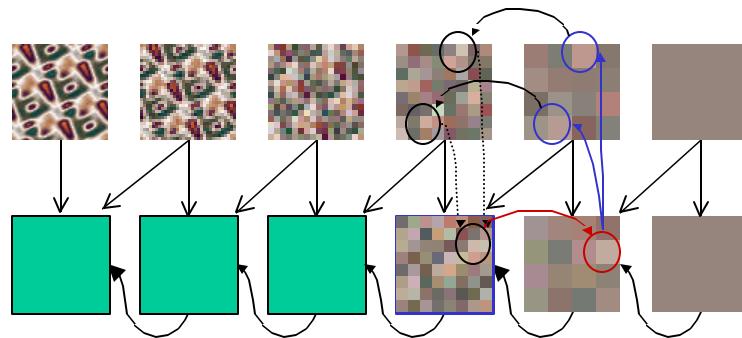
Pixels are generated by sampling  
from the analysis pyramid.

*Paul Viola*

*MIT AI Lab*

## *Shannon Resampling*

### *Step 2b: Fill in subsequent levels*



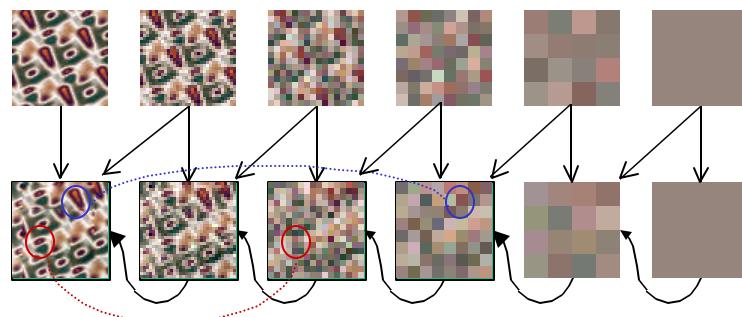
*Paul Viola*

Pixels are generated by  
*conditional sampling*  
(dependent on the parent).

*MIT AI Lab*

## *Shannon Resampling*

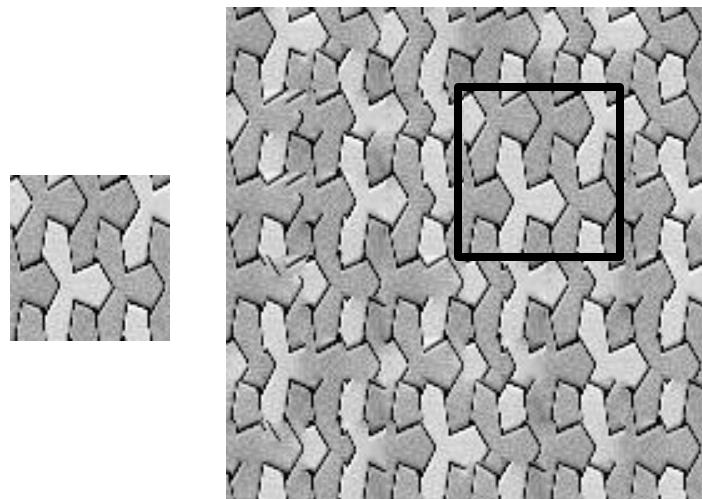
### *Finish the pyramid*



*Paul Viola*

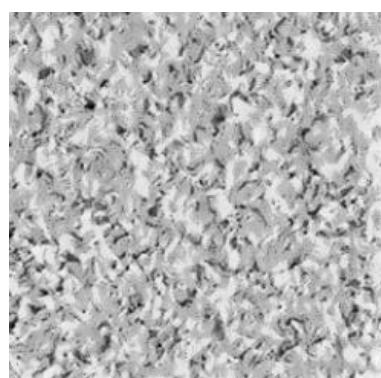
Decisions made at low resolutions  
generate discrete features in the final image.

*MIT AI Lab*



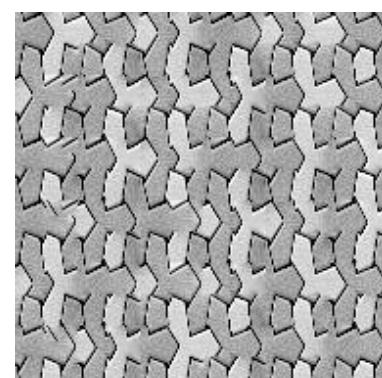
*Paul Viola*

*MIT AI Lab*



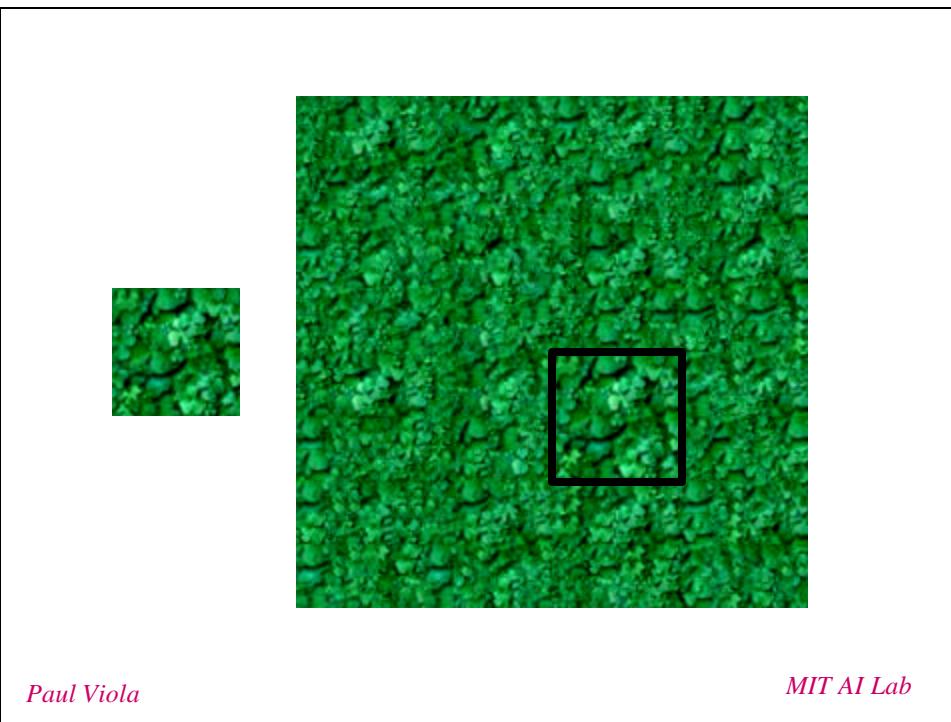
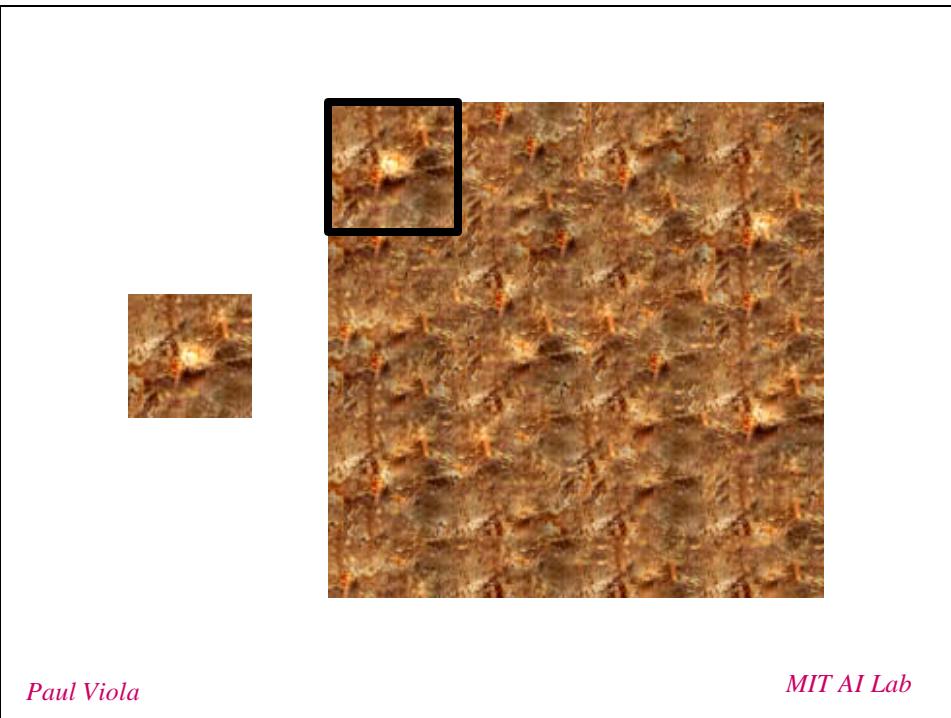
**B & H**

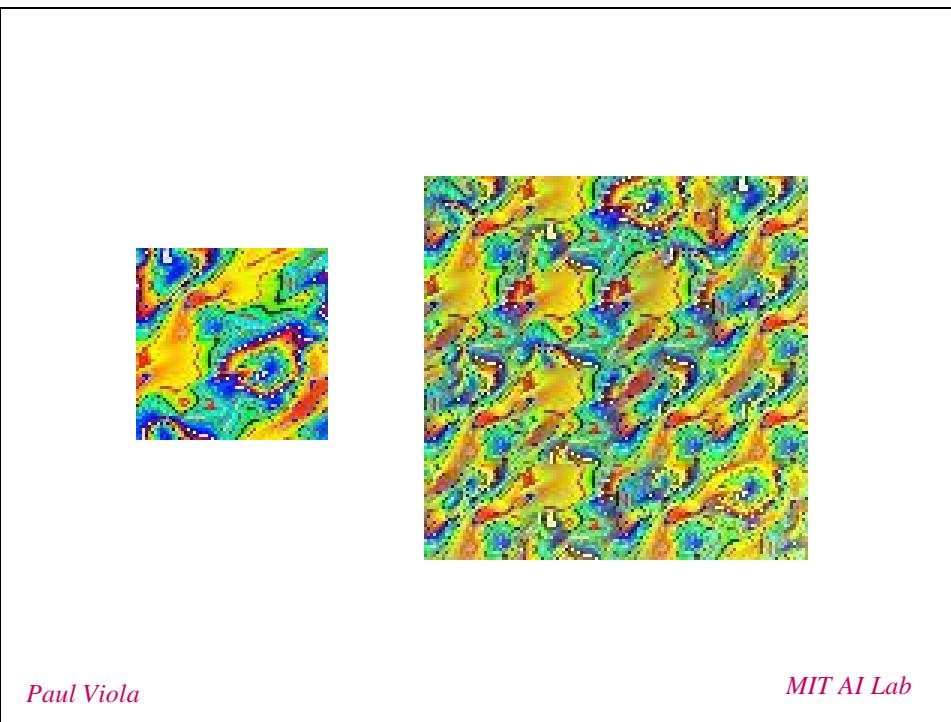
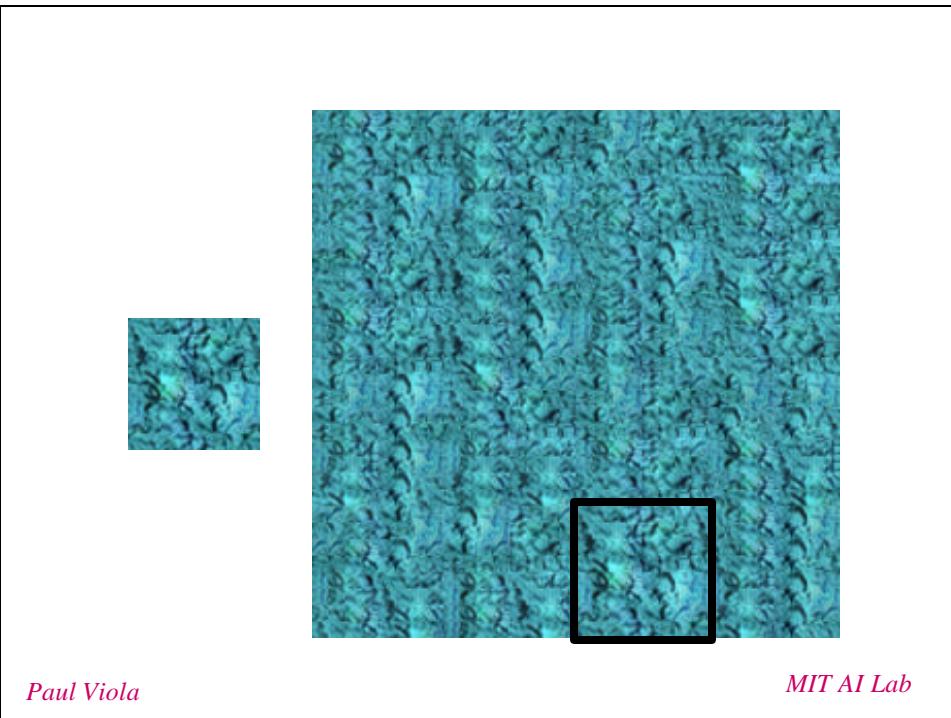
*Paul Viola*



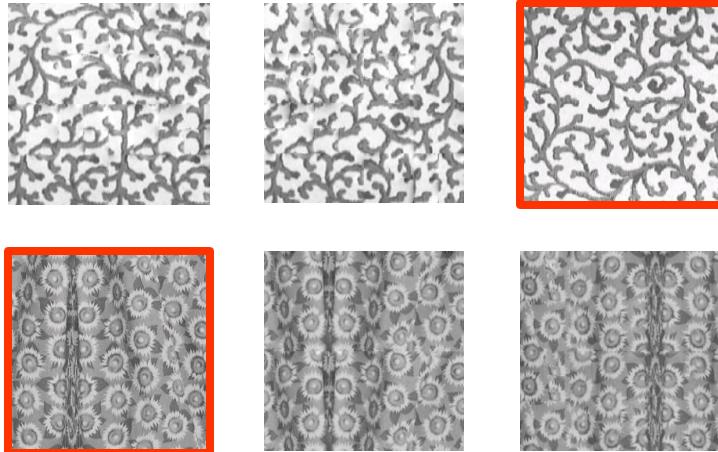
**D & V**

*MIT AI Lab*





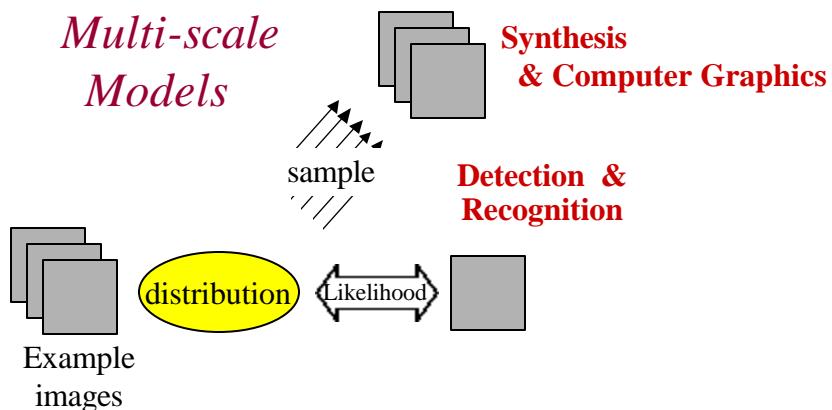
## *FRAME: Challenge*



*Paul Viola*

*MIT AI Lab*

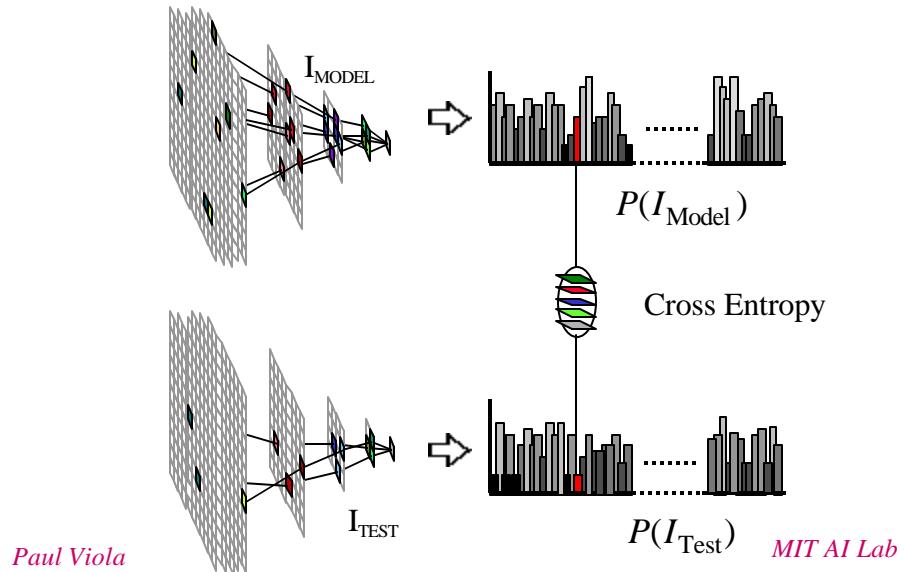
## *Non-Parametric Multi-scale Models*



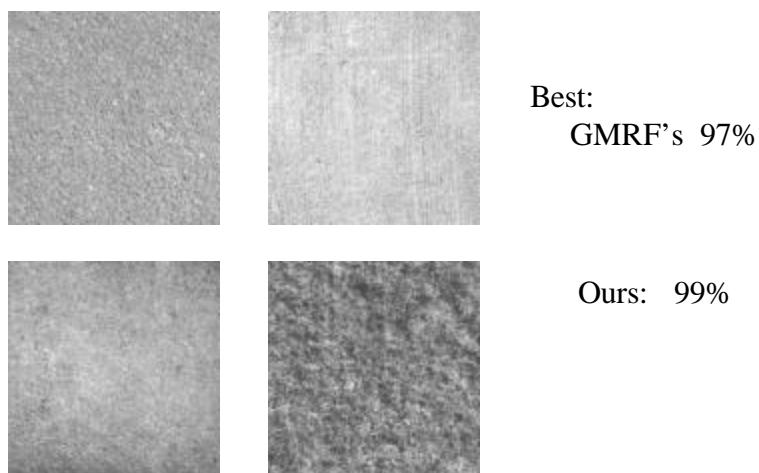
*Paul Viola*

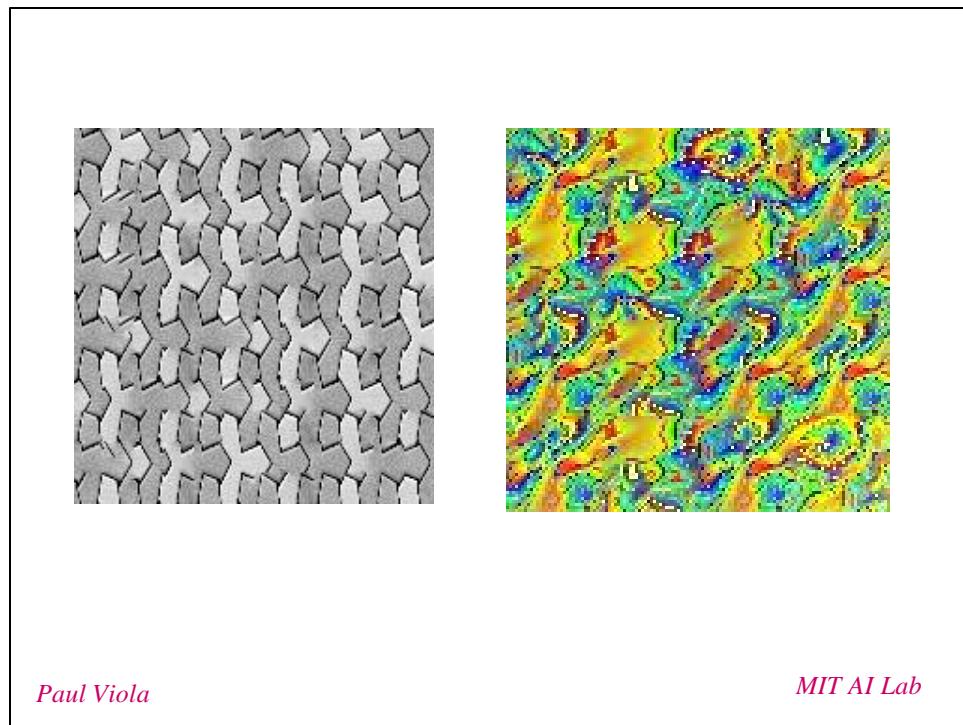
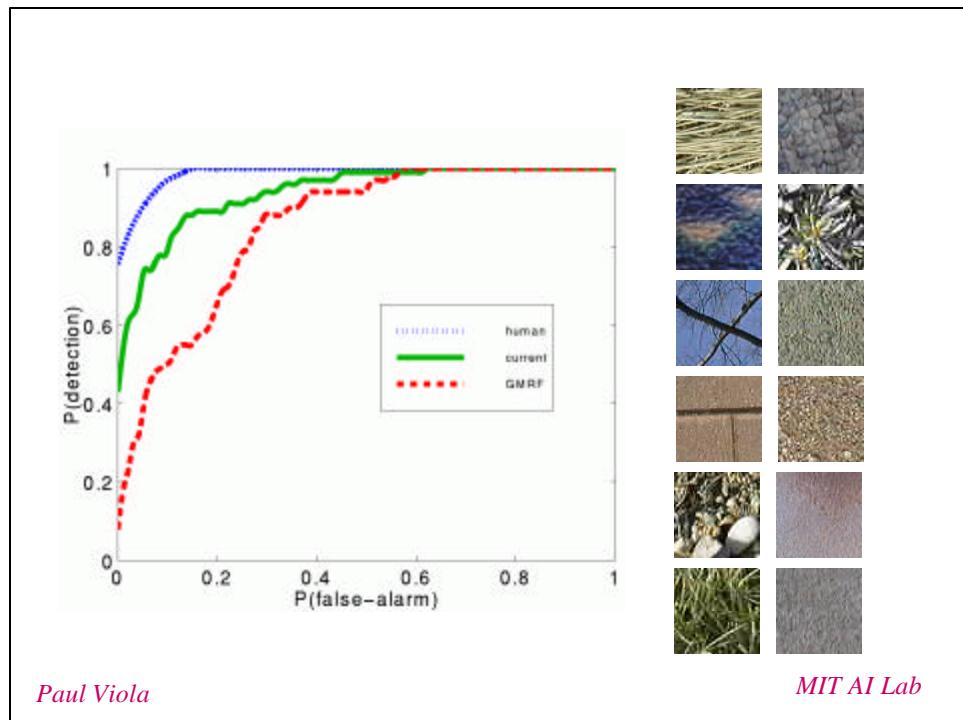
*MIT AI Lab*

## *Discrimination via Cross Entropy*



## *Meastex: Texture Classification*





## *Where is the boundary between texture and objects?*

- Our model can synthesize and recognize complex and structured textures.
  - Far beyond previous older definitions of texture.
- Where is the boundary between these complex textures and other patterns in images
  - Like faces, human forms, automobiles, etc?
- Only experiments can tell...

*Paul Viola*

*MIT AI Lab*

## *The Jacket Hypothesis*



*Paul Viola*



*AI Lab*

## *What about face detection?*



- Synthesis is convincing
- Train a texture model to detect faces

*Paul Viola*

Tom Rikert & *MIT AI Lab*

## *Detecting Objects*



- *Key Difficulties:*
  - Variation in Pose, Deformation, & variation across class
- Most Object Recognition approaches are either:
  - Very dependent on precise shape and size
  - Entirely dependent on simple features (... color, edge histograms)
- *Hypothesis:*
  - Object recognition is closely related to texture perception

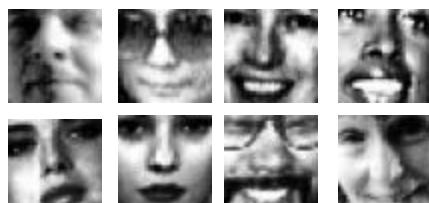
*Paul Viola*

*MIT AI Lab*

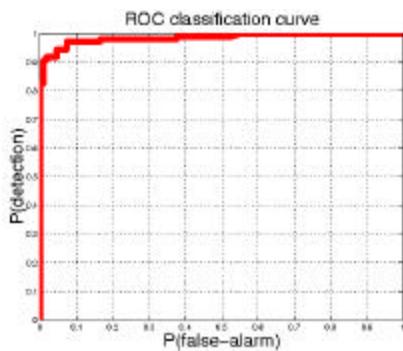
## *Detection Results*



Non-face test images

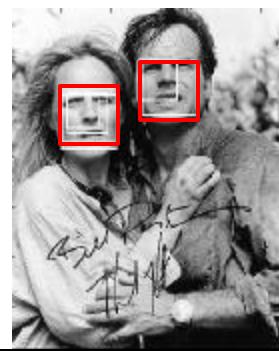
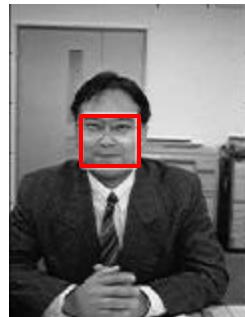
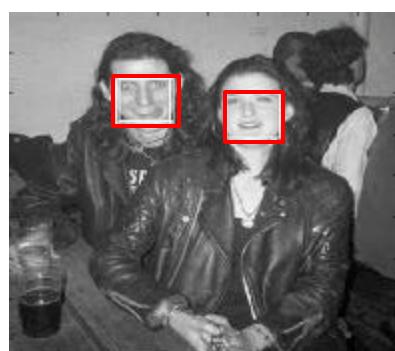


*Paul Viola*  
Web face test images



MIT AI Lab

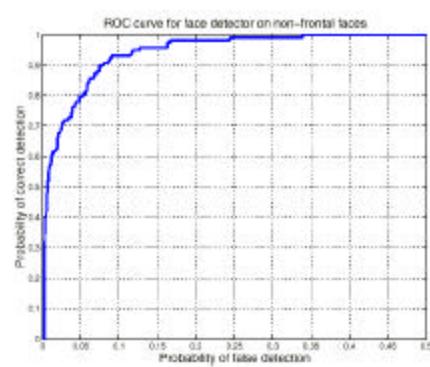
## *Detection Results:*



*Paul Viola*

MIT AI Lab

## *Non-frontal faces*

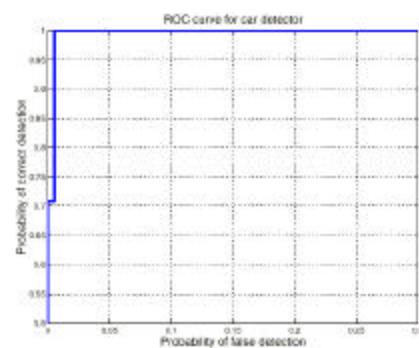


But naïve detection  
is expensive

*Paul Viola*

*MIT AI Lab*

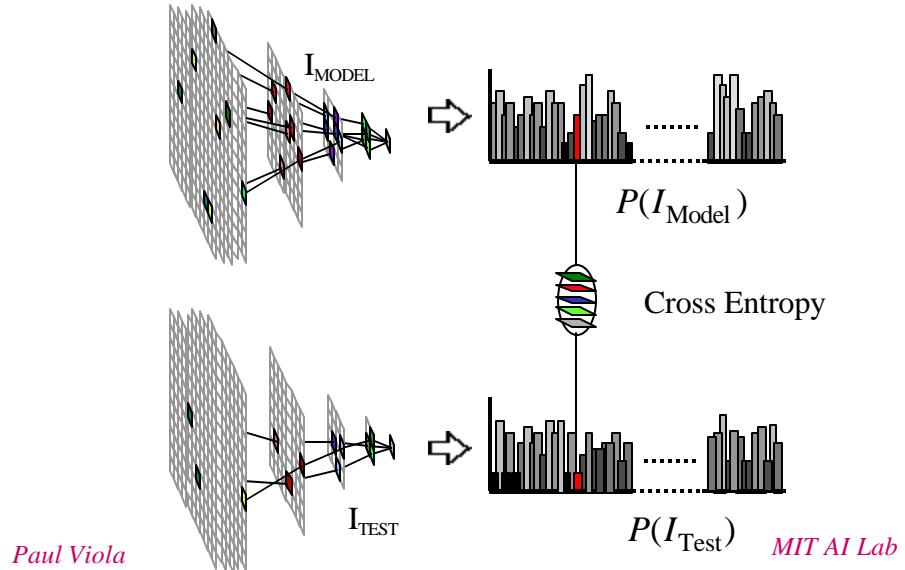
## *Car Images*



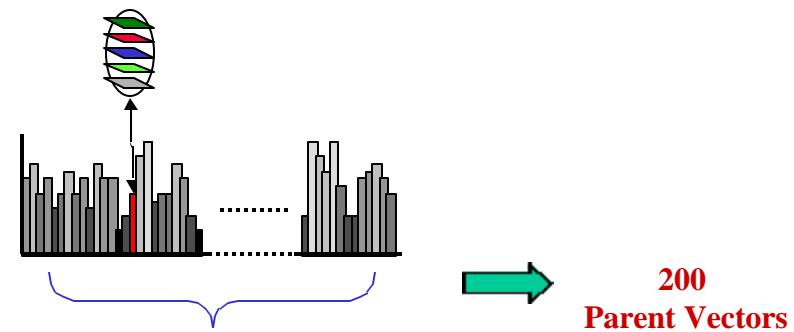
*Paul Viola*

*MIT AI Lab*

## *Texture recognition via Cross Entropy*



## *Pruning the density estimator*

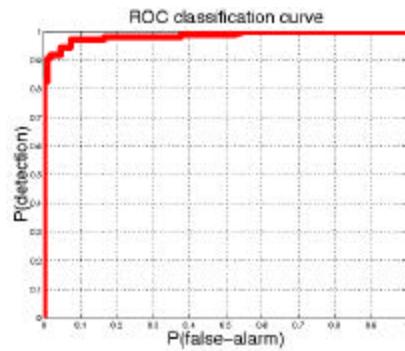


**Result:** Detection/Classification is **faster** than template correlation

*Paul Viola*

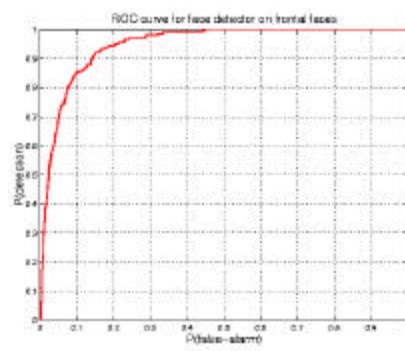
*MIT AI Lab*

*ROC using 200 vectors...*



2000 Vector Model

*Paul Viola*



200 Vector Model

*MIT AI Lab*

*Scanning results:*



Time: 9 secs



*Paul Viola*

*MIT AI Lab*

## *Key facial features*

- determined automatically
- located automatically



Multi-scale features which are come  
from the face model can be automatically  
detected for many individuals

*Paul Viola*

*MIT AI Lab*



*Paul Viola*

*b*



*Pa... 1321-2*

*b*



*Pa... 1321-2*

*b*

*Future Work:*  
*New Face Recognition Algorithm*

- Facial identity depends both on the types of features and their location.

*Paul Viola*

*MIT AI Lab*

# *6.891 Machine Learning and Neural Networks*

## Lecture 24: The End

© Paul Viola 1999

Machine Learning

## *News*

- The Final is on Monday of Final's week at 1:30
  - » In this room...
- Conflict exam will be in NE43 on Tuesday Morning at 9:30.
  - » Come to Kinh's office at 9:15 so we can set people up.
- Last year's final will be on the web by 1PM.

© Paul Viola 1999

Machine Learning

## *Review & Overview*

- Lectures 22 and 23:
  - » Statistical image processing
    - Estimate statistical models from examples
    - Applications
      - Denoising
      - Synthesis
      - Recognition
- An overview of Machine Learning
- What I would have liked to cover...

© Paul Viola 1999

Machine Learning

## *6891 at a Glance*

- Probability
  - » Bayes Law
- Linear Algebra
  - » Eigenvectors and inverses
- Bayesian Classification
- Discriminant Functions
  - » Perceptron's, MLP's
- Support Vector Machines
- Regularization
  - » Radial Basis Functions
- Unsupervised Learning and PCA
- Bayes Nets and HMMs

© Paul Viola 1999

Machine Learning

## *In the beginning... Probability*

- The key concepts of probability
  - » The basic algebra of probability
    - Independent events add
    - Relationships between conditional and joint distributions
  - » Densities work like probabilities (mostly)
  - » Bayes Law allows us to make decisions
    - Loss functions are critical
  - » Maximum likelihood allows us to learn distributions
    - Bayesian estimation averages over parameters
  - » Exponential densities are easiest to work with
  - » Mixtures of Gaussians are powerful (but EM is slow)
  - » Non-parametric estimators are more powerful
    - But are difficult to represent

© Paul Viola 1999

Machine Learning

## *Linear Algebra*

- The inverse and pseudo-inverse are everywhere
  - » Solving least squares problems
- Covariance and co-occurrence are everywhere
  - » Estimating a Gaussian
  - » Fitting a line to data
  - » Principal components analysis
- Eigenvectors simplify most linear algebra
  - » Especially for symmetric positive semi-definite mats
  - » Allow you to compute inverses & square roots
  - » Allow you to understand distributions and linear dependence

© Paul Viola 1999

Machine Learning

## *Bayesian Classification*

- Start out with strong assumptions about your data
  - » Number of classes, structure of the classes
- Use data to estimate the distribution of each class
- Use Bayes' law to classify new examples
- Advantages:
  - » Can estimate the probability of classes (confidence)
  - » Can validate the model
  - » Harder to over-train or over-fit
- Disadvantages:
  - » May not use data efficiently
  - » Sensitive to poor assumptions

© Paul Viola 1999

Machine Learning

## *Discriminant Functions*

- Attempt to estimate the discriminant function directly
  - » Linear
  - » Polynomial
  - » Multi-layer perceptron
- Specifically minimizes the number of errors
- Advantages:
  - » Don't waste time on distributions (just the boundary)
- Disadvantages:
  - » No natural measure of confidence
  - » Can over-train

© Paul Viola 1999

Machine Learning

## *Support Vector Machines*

- A principled and direct way to simultaneously minimize errors while yielding the simplest possible classifier
  - » Occam's razor
- Using the Kernel Trick ™
  - » Can find a very complex polynomial with little work
- Using the Margin Trick™
  - » Maximizes generalization in the face of complexity
- Simple learning criteria
- Well studied learning algorithm
  - » Quadratic programming

© Paul Viola 1999

Machine Learning

## *Regularization*

- Sometimes you would like to find the smoothest function which is close to the data
  - » Minimize the squared error
  - » Minimize the squared first derivative (or 2<sup>nd</sup> deriv.)
- The least squares solution:
  - » Is a sum of kernel functions centered on the data
  - » Kernel functions depend on the smoothness penalty
- Derivative penalties yield polynomial kernels
  - » First -> linear, Second -> cubic, Hairy -> Gaussian

© Paul Viola 1999

Machine Learning

## *Unsupervised Learning*

- Transforming the input so that it is more manageable
  - » PCA: The data can be represented using fewer numbers
    - Can compress data, make learning simpler
  - » ICA: The resulting data is now more independent
    - Can separate signals that were mixed
  - » Informative Features (by John Fisher)
    - Can represent just the critical information

© Paul Viola 1999

Machine Learning

## *Bayes Nets*

- Models of the conditional dependencies between variables
  - » Usually many variables
- A complete model would be intractable
  - » Exponential number of parameters
    - Impossible to learn or reason
- By assuming that certain vars are independent
  - » Number of params goes down rapidly
  - » Efficient reasoning is possible
- Bayes Nets are very general and can be used in many ways

© Paul Viola 1999

Machine Learning

## *Hidden Markov Models*

- A type of Bayes Net that allows reasoning over time
- The true state of the world is unknown
  - » You have noisy observations
- HMM use temporal dependencies to differentiate ambiguous states

© Paul Viola 1999

Machine Learning

## *The VC dimension*

- Each class of learned functions has a VC dimension
  - » Perceptron:  $\text{VCdim} = \text{number of weights}$
- VC dimension measures the capacity of the classifier
  - »  $\text{VCdim}$  is the max number of points which can be shattered
  - » Shattered = assigned any set of labels
- Intuition: larger capacity requires more data
  - » Like polynomials:  $N^{\text{th}}$  order requires  $N+1$  points
- The bounds are actually probabilistic
  - » The probability of that the error rates exceeds a particular rate is bounded by a function of VC and N.

© Paul Viola 1999

Machine Learning

## *Symbolic Learning*

- Often the correct classification rule is symbolic
  - » If  $BP < 50$  and  $HR < 50$  then administer DRUG
- While Bayes Nets can reason in this way, they do not offer much help in learning the relationships from data
  - » If structure of net is given, then params can be estimated
- This is sometimes called rule learning
- Decision Trees – ID3, CART, etc.
  - » Pick a feature, split into ranges
  - » For each case, pick another feature and repeat
  - » Each leaf should have only one label

© Paul Viola 1999

Machine Learning

## *Combining Classifiers*

- We have encountered many learning techniques
  - » Each has multiple variants
- Bagging
  - » Train the same classifier on different subsets of the data
  - » Related to cross-validation (or the Bootstrap)
- Stacking
  - » Perhaps the best approach is to train each type of classifier and then have them vote.
    - Combine 100 different types of neural networks
    - Many types of generalized perceptrons
- Boosting
  - » Train a sequence of classifiers on re-weighted data sets

© Paul Viola 1999

Machine Learning

## *Policy Learning*

- You must act over time to maximize some reward
  - » Portfolios: Buy and sell stock to max return and min risk
  - » Two armed bandit: tradeoff exploration for exploitation
  - » Learn a sequence of action which takes you from the start to the goal – like in a video game
- Sometimes your feedback is delayed
  - » Rarely do you get detailed feedback on your actions
- Policy
  - » Mapping from state of the world to actions
- Reinforcement Learning (Leslie Kaelbling)
- Game Learning (Backgammon)

© Paul Viola 1999

Machine Learning

## *Language Learning*

- How can you learn to pluralize? (phonetically)
  - » Wug
- How do you discover parts of speech?
- How do you learn the grammar of English?
  - » Stochastic Context Free Grammar
    - Generalization of HMM
    - $S \rightarrow NP\ VP$ ,  $VP \rightarrow V\ NP$ , etc.

© Paul Viola 1999

Machine Learning