Announcements

- Midterm back today (end of class)

Recognition

The "Margaret Thatcher Illusion", by Peter Thompson

Readings

- C. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, 1998, Chapter 1.
- Forsyth and Ponce, Chap 22.3 (through 22.3.2--eigenfaces)

Recognition

The "Margaret Thatcher Illusion", by Peter Thompson

Readings

- C. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, 1998, Chapter 1.
- Forsyth and Ponce, 22.3 (eigenfaces)

Recognition problems

What is it?

- Object detection

Who is it?

- Recognizing identity

What are they doing?

- Activities

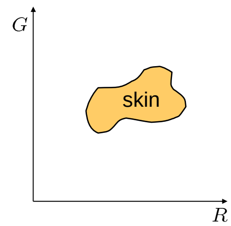
All of these are **classification** problems

- Choose one class from a list of possible candidates

Face detection

How to tell if a face is present?

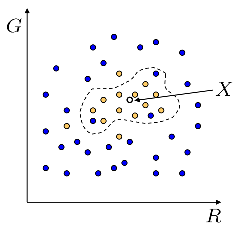
One simple method: skin detection



Skin pixels have a distinctive range of colors
- Corresponds to region(s) in RGB color space
  - for visualization, only R and G components are shown above

Skin classifier
- A pixel X = (R,G,B) is skin if it is in the skin region
- But how to find this region?

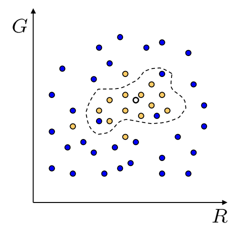
Skin detection



Learn the skin region from examples
- Manually label pixels in one or more “training images” as skin or not skin
- Plot the training data in RGB space
  - skin pixels shown in orange, non-skin pixels shown in blue
  - some skin pixels may be outside the region, non-skin pixels inside. Why?

Skin classifier
- Given X = (R,G,B): how to determine if it is skin or not?

Skin classification techniques

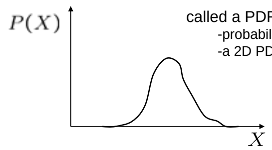


Skin classifier
- Given X = (R,G,B): how to determine if it is skin or not?
- Nearest neighbor
  - find labeled pixel closest to X
  - choose the label for that pixel
- Data modeling
  - fit a model (curve, surface, or volume) to each class
- Probabilistic data modeling
  - fit a probability model to each class

Probability

Basic probability

- X is a random variable
- P(X) is the probability that X achieves a certain value



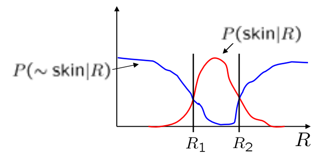
called a PDF
-probability distribution/density function
-a 2D PDF is a surface, 3D PDF is a volume

- $0 \leq P(X) \leq 1$
- $\int_{-\infty}^{\infty} P(X)dX = 1$ or $\sum P(X) = 1$

  continuous X discrete X

- Conditional probability: P(X | Y)
  - probability of X given that we already know Y

Probabilistic skin classification



Now we can model uncertainty

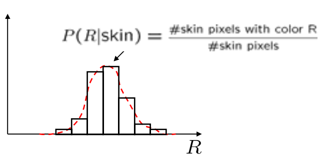
- Each pixel has a probability of being skin or not skin
  - $P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$

Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?
- Choose interpretation of highest probability
  - set X to be a skin pixel if and only if $R_1 < X \leq R_2$

Where do we get $P(\text{skin}|R)$ and $P(\sim \text{skin}|R)$?

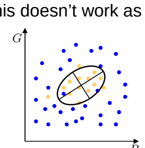
Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

- It is simply a histogram over the pixels in the training images
  - each bin R_i contains the proportion of skin pixels with color R_i

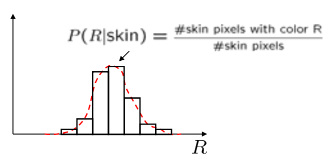
This doesn't work as well in higher-dimensional spaces. Why not?



Approach: fit parametric PDF functions

- common choice is rotated Gaussian
  - center $\mathbf{c} = \overline{X}$
  - covariance $\sum_X (X - \overline{X})(X - \overline{X})^T$
    - » orientation, size defined by eigenvecs, eigenvals

Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

- It is simply a histogram over the pixels in the training images
  - each bin R_i contains the proportion of skin pixels with color R_i

But this isn't quite what we want

- Why not? How to determine if a pixel is skin?
- We want P(skin | R) not P(R | skin)
- How can we get it?

Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

In terms of our problem:

what we measure (**likelihood**)

domain knowledge (**prior**)

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want (**posterior**)

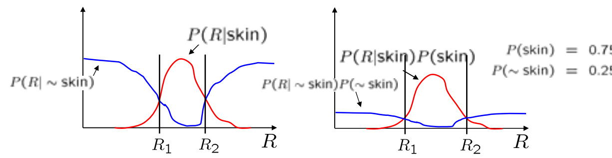
normalization term

$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$

What could we use for the prior P(skin)?

- Could use domain knowledge
  - P(skin) may be larger if we know the image contains a person
  - for a portrait, P(skin) may be higher for pixels in the center
- Could learn the prior from the training set. How?
  - P(skin) may be proportion of skin pixels in training set

Bayesian estimation



likelihood

posterior (unnormalized)

Bayesian estimation

= minimize probability of misclassification

- Goal is to choose the label (skin or ~skin) that maximizes the posterior
  - this is called **Maximum A Posteriori (MAP) estimation**
- Suppose the prior is uniform: P(skin) = P(~skin) = 0.5
  - in this case $P(\text{skin}|R) = cP(R|\text{skin})$, $P(\sim\text{skin}|R) = cP(R|\sim\text{skin})$
  - maximizing the posterior is equivalent to maximizing the likelihood
    - » $P(\text{skin}|R) > P(\sim\text{skin}|R)$ if and only if $P(R|\text{skin}) > P(R|\sim\text{skin})$
  - this is called **Maximum Likelihood (ML) estimation**

Skin detection results

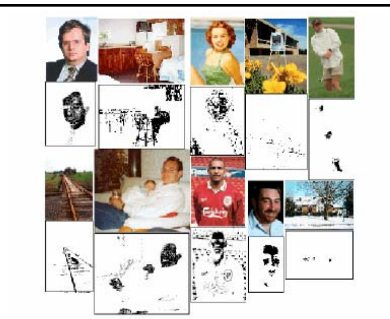
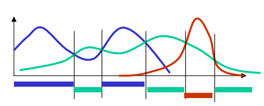


Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

General classification

This same procedure applies in more general circumstances

- More than two classes
- More than one dimension



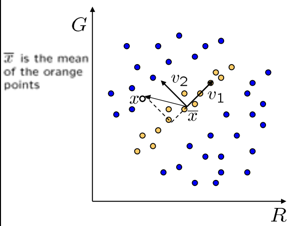
Example: face detection

- Here, X is an image region
  - dimension = # pixels
  - each face can be thought of as a point in a high dimensional space

H. Schneiderman and T.Kanade

H. Schneiderman, T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000) http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/hws/www/CVPR00.pdf

Linear subspaces



convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

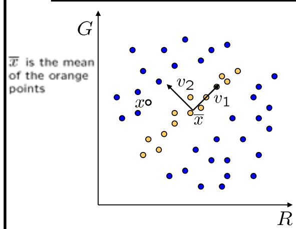
What does the $\mathbf{v_2}$ coordinate measure?
- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v_1}$ coordinate measure?
- position along line
- use it to specify which orange point it is

Classification can be expensive
- Must either search (e.g., nearest neighbors) or store large PDF's

Suppose the data points are arranged as above
- Idea—fit a line, classifier measures distance to line

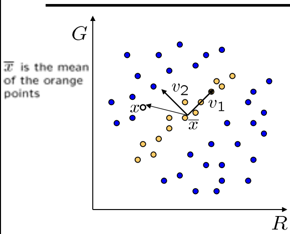
Dimensionality reduction



How to find $\mathbf{v_1}$ and $\mathbf{v_2}$?
- work out on board

Dimensionality reduction
- We can represent the orange points with *only* their $\mathbf{v_1}$ coordinates
  - since $\mathbf{v_2}$ coordinates are all essentially 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems

Linear subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\|^2$$

What unit vector **v** minimizes *var*?

$$\mathbf{v}_2 = min_{\mathbf{v}} \{var(\mathbf{v})\}$$

What unit vector **v** maximizes *var*?

$$\mathbf{v}_1 = max_{\mathbf{v}} \{var(\mathbf{v})\}$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\| \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathbf{T}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \mathbf{v} \\ &= \mathbf{v}^{\mathbf{T}} \left[\sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \right] \mathbf{v} \\ &= \mathbf{v}^{\mathbf{T}} \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \end{aligned}$$

Solution: $\mathbf{v_1}$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v_2}$ is eigenvector of **A** with *smallest* eigenvalue

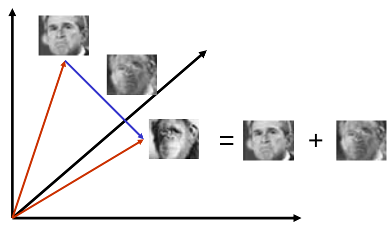
Principal component analysis

Suppose each data point is N-dimensional
- Same procedure applies:

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\| \\ &= \mathbf{v}^{\mathbf{T}} \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \end{aligned}$$

- The eigenvectors of **A** define a new coordinate system
  - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
  - eigenvector with smallest eigenvalue has least variation
- We can compress the data by only using the top few eigenvectors
  - corresponds to choosing a "linear subspace"
    - » represent points on a line, plane, or "hyper-plane"
  - these eigenvectors are known as the ***principal components***

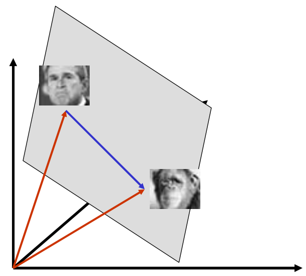
The space of faces



An image is a point in a high dimensional space

- An N x M image is a point in R^{NM}
- We can define vectors in this space as we did in the 2D case

Dimensionality reduction



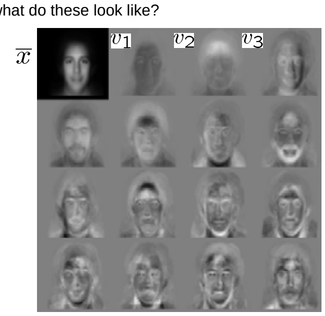
The set of faces is a "subspace" of the set of images

- Suppose it is K dimensional
- We can find the best subspace using PCA
- This is like fitting a "hyper-plane" to the set of faces
  - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_K}$
  - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces

PCA extracts the eigenvectors of **A**

- Gives a set of vectors $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$, ...
- Each one of these vectors is a direction in face space
  - what do these look like?



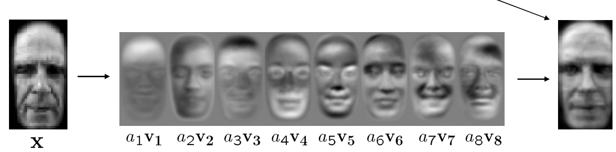
Projecting onto the eigenfaces

The eigenfaces $\mathbf{v_1}$, ..., $\mathbf{v_K}$ span the space of faces

- A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v_1}}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v_2}}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}}) \cdot \mathbf{v_K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$



Recognition with eigenfaces

Algorithm

1. Process the image database (set of images with labels)
   - Run PCA—compute eigenfaces
   - Calculate the K coefficients for each image
2. Given a new image (to be recognized) **x**, calculate K coefficients

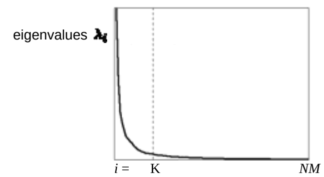
$$\mathbf{x} \to (a_1, a_2, \ldots, a_K)$$

3. Detect if x is a face

$$\|\mathbf{x} - (\overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K})\| < \text{threshold}$$

4. If it is a face, who is it?
   - Find closest labeled face in database
     - nearest-neighbor in K-dimensional space

Choosing the dimension K



How many eigenfaces to use?

Look at the decay of the eigenvalues

- the eigenvalue tells you the amount of variance “in the direction” of that eigenface
- ignore eigenfaces with low variance

Object recognition

This is just the tip of the iceberg

- We’ve talked about using pixel color as a feature
- Many other features can be used:
  - edges
  - motion (e.g., optical flow)
  - object size
  - SIFT
  - ...
- Classical object recognition techniques recover 3D information as well
  - given an image and a database of 3D models, determine which model(s) appears in that image
  - often recover 3D pose of the object as well