Recap: Subspace Methods

- Subspace methods:
  - **Reconstructive**: PCA, ICA, NMF
  - **Discriminative**: FLD, SVM, CCA

- Representation: classification, regression

Recap: Obj. Detection by Distance TO Eigenspace

- For each test image, compute the reprojection error
  - An n-pixel image \( x \in \mathbb{R}^n \) can be projected to the low-dimensional feature space \( y \in \mathbb{R}^m \) by
    \[ y = Ux \]
  - From \( y \in \mathbb{R}^m \), the reconstruction of the point is \( U^Ty \)
  - The error of the reconstruction is
    \[ \| x - U^TUx \| \]

- Accept a detection if this error is low.
- Assumption: subspace is optimized to the target object (class).
- Other classes are not represented well \( \Rightarrow \) large error.

Recap: Obj Identification by Distance IN Eigenspace

- Objects are represented as coordinates in an n-dim. eigenspace.
- Example:
  - 3D space with points representing individual objects or a manifold represented parametric eigenspace (e.g., orientation, pose, illumination).
- Estimate parameters by finding the NN in the eigenspace

Recap: Eigenfaces

- Example images of faces
Recap: Restrictions of PCA

- PCA minimizes projection error
- PCA is "unsupervised" no information on classes is used
- Discriminating information might be lost

Topics of This Lecture

- Object Categorization
  - Problem Definition
  - Challenges
- Sliding-Window based Object Detection
  - Detection via Classification
  - Global Representations
  - Classifier Construction
- Classification with Boosting
  - AdaBoost
  - Viola-Jones Face Detection
- Classification with SVMs
  - Support Vector Machines
  - HOG Detector

Identification vs. Categorization

- Find this particular object
- Recognize ANY car
- Recognize ANY cow

Object Categorization - Potential Applications

There is a wide range of applications, including:

- Autonomous robots
- Navigation, driver safety
- Consumer electronics
- Content-based retrieval and analysis for images and videos
- Medical image analysis

How many object categories are there?

~10,000 to 30,000

Source: Fei-Fei Li, Rob Fergus, Antonio Torralba.
Biederman 1987
Challenges: Robustness
- Detection in crowded, real-world scenes
  - Learn object variability
  - Changes in appearance, scale, and articulation
  - Compensate for clutter, overlap, and occlusion

Topics of This Lecture
- Object Categorization
  - Problem Definition
  - Challenges
- Sliding-Window based Object Detection
  - Detection via Classification
  - Global Representations
  - Classifier Construction
- Classification with Boosting
  - AdaBoost
  - Viola-Jones Face Detection
- Classification with SVMs
  - Support Vector Machines
  - HOG Detector

Detection via Classification: Main Idea
- Basic component: a binary classifier

Detection via Classification: Main Idea
- If object may be in a cluttered scene, slide a window around looking for it.
  - Essentially, this is a brute-force approach with many local decisions.
What is a Sliding Window Approach?

- Search over space and scale

Detection as subwindow classification problem

“In the absence of a more intelligent strategy, any global image classification approach can be converted into a localization approach by using a sliding-window search.”

Detection via Classification: Main Idea

Fleshing out this pipeline a bit more, we need to:
1. Obtain training data
2. Define features
3. Define classifier

Feature extraction: Global Appearance

Simple holistic descriptions of image content
- Grayscale / color histogram
- Vector of pixel intensities

Gradient-based Representations

- Idea
  - Consider edges, contours, and (oriented) intensity gradients
**Gradient-based Representations**

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

- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Still more spatial information than single global histogram
  - Contrast-normalization: try to correct for variable illumination

**Histograms of Oriented Gradients (HoG)**

- Map each grid cell in the input window to a histogram counting the gradients per orientation.

---

**Classifier Construction**

- How to compute a decision for each subwindow?

- Image feature

---

**Discriminative Methods**

- Learn a decision rule (classifier) assigning image features to different classes

**Linear Classifiers**

- Let $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

  $$w_1 x_1 + w_2 x_2 + b = 0$$

- Decision boundary

---

**Classifier Construction: Many Choices...**

- Nearest Neighbor
  - Berg, Berg, Malik 2005, Chum, Zisserman 2007, Boiman, Shechtman, Irani 2008, ...

- Neural networks
  - LeCun, Bottou, Bengio, Haffner 1998, Rowley, Baluja, Kanade 1998, ...

- Boosting
  - Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006, Benenson 2012, ...

- Support Vector Machines

- Randomized Forests
  - Amit, Geman 1997, Breiman 2001, Opelt, Fua 2006, Gall, Lempitsky 2009, ...

---

**Support Vector Machines**

- Suitable for high-dimensional feature spaces.
- Minimize structural risk.
- Margin maximization.

**Randomized Forests**

- Ensemble learning.
- Robust to overfitting.
- Can handle high-dimensional data.

**Linear Classifiers**

- Simple and interpretable.
- Effective for linearly separable data.
- Can be used for classification and regression.
**Linear Classifiers**

- Find linear function to separate positive and negative examples

\[ \mathbf{x}_+ \text{ positive: } \mathbf{w}^T \mathbf{x}_+ + b \geq 0 \]
\[ \mathbf{x}_- \text{ negative: } \mathbf{w}^T \mathbf{x}_- + b < 0 \]

Which line is best?

**Support Vector Machines (SVMs)**

- Discriminative classifier based on optimal separating hyperplane (i.e. line for 2D case)
- Maximize the margin between the positive and negative training examples

**Finding the Maximum Margin Line**

- Solution: \( \mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n \)

**Questions**

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

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

Non-Linear SVMs: Feature Spaces

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

\[ \Phi: \mathbf{x} \rightarrow \Phi(\mathbf{x}) \]

More on that in the Machine Learning lecture...

Some Often-Used Kernel Functions

- Linear:
  \[ K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \]
- Polynomial of power p:
  \[ K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p \]
- Gaussian (radial-basis function):
  \[ K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right) \]

Nonlinear SVMs

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

\[ K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \]

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

\[ \sum a_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \]


Questions

- What if the features are not 2d?
  - Generalizes to d-dimensions - replace line with “hyperplane”
- What if the data is not linearly separable?
  - Non-linear SVMs with special kernels
- What if we have more than just two categories?
Multi-Class SVMs

- Achieve multi-class classifier by combining a number of binary classifiers
- **One vs. all**
  - Training: learn an SVM for each class vs. the rest
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- **One vs. one**
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

SVMs for Recognition

1. Define your representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples.
4. Given this "kernel matrix" to SVM optimization software to identify support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.

Pedestrian Detection

- Detecting upright, walking humans using sliding window’s appearance/texture; e.g.,

Pedestrian detection with HoGs & SVMs

- Navneet Dalal, Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Classifier Construction: Many Choices...

- Nearest Neighbor
  - Shakharovich, Viola, Darrell 2003
  - Berg, Berg, Malik 2005, Boiman, Shechtman, Irani 2008, ...
- Neural networks
  - LeCun, Bottou, Bengio, Haffner 1998
  - Rowley, Baluja, Kanade 1998
- Support Vector Machines
  - Vapnik, Schölkopf 1995
  - Papageorgiou, Poggio, IJCV 2000
  - Viola, Jones & Snow, ICCV 2003
  - Dalal, Triggs, CVPR 2005
  - Vedaldi, Zisserman 2012
- Randomized Forests
  - Amit, Geman 1997
  - Breiman 2001
  - Lepetit, Fua 2006
  - Gall, Lempitsky 2009

Boosting

- Build a strong classifier \( H \) by combining a number of "weak classifiers" \( h_1, \ldots, h_M \), which need only be better than chance.
- Sequential learning process: at each iteration, add a weak classifier
  - Flexible to choice of weak learner
    - including fast simple classifiers that alone may be inaccurate
  - We’ll look at Freund & Schapire’s AdaBoost algorithm
    - Easy to implement
    - Base learning algorithm for Viola-Jones face detector

### AdaBoost: Detailed Training Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$ iterations:
   - a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function $J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x_n) \neq t_n)$. Set $h_m(x) = 1$ if $I$ is true, 0 otherwise.
   - b) Estimate the weighted error of this classifier on $X$: $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$.
   - c) Calculate a weighting coefficient for $h_m(x)$: $\alpha_m = \ln \left( \frac{1 - \epsilon_m}{\epsilon_m} \right)$.
   - d) Update the weighting coefficients: $w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(h_m(x_n) \neq t_n)\}$. 

### AdaBoost: Recognition

- Evaluate all selected weak classifiers on test data.
- $h_1(x), \ldots, h_m(x)$
- Final classifier is weighted combination of selected weak classifiers: $H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)$.

- Very simple procedure!
  - Less than 10 lines in Matlab!
  - But works extremely well in practice.

### AdaBoost - Formalization

- 2-class classification problem
  - Given: training set $X = \{x_1, \ldots, x_N\}$ with target values $T = \{t_1, \ldots, t_N\}$, $t_n \in \{-1, 1\}$.
  - Associated weights $W = \{w_1, \ldots, w_N\}$ for each training point.
- Basic steps
  - In each iteration, AdaBoost trains a new weak classifier $h_m(x)$ based on the current weighting coefficients $W^{(m)}$.
  - We then adapt the weighting coefficients for each point:
    - Increase $w_i$ if $x_i$ was misclassified by $h_m(x)$.
    - Decrease $w_i$ if $x_i$ was classified correctly by $h_m(x)$.
  - Make predictions using the final combined model:
    $$ H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right) $$

### AdaBoost: Intuition

- Consider a 2D feature space with positive and negative examples.
- Each weak classifier splits the training examples with at least 50% accuracy.
- Examples misclassified by a previous weak learner are given more emphasis at future rounds.
Example: Face Detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a "patch"/window

- Now we'll take AdaBoost and see how the Viola-Jones face detector works

Feature extraction

"Rectangular" filters

Feature output is difference between adjacent regions

- Efficiently computable with integral image: any sum can be computed in constant time
- Avoid scaling images \rightarrow scale features directly for same cost

Example

Integral Image

Value at \((x,y)\) is sum of pixels above and to the left of \((x,y)\)

\[ D = x + y + u - x - y = 2D \]

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

Avoid scaling images \rightarrow scale features directly for same cost

Large Library of Filters

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

Weak classifier: \( f(x) > \theta \)

AdaBoost for Feature+Classifier Selection

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

Resulting weak classifier:

\[ h_i(x) = \begin{cases} +1 & \text{if } f_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases} \]

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

AdaBoost for Efficient Feature Selection

- Image features = weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
  - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this features is a simple function of error rate
  - Reweight examples

(first version appeared at CVPR 2001)
Cascading Classifiers for Detection

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

[Burget & Geman, IJCV 2001]
[Rowley et al., PAMI 1998]
[Viola & Jones, CVPR 2001]

Viola-Jones Face Detector: Summary

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Viola-Jones Face Detector: Results

Chain classifiers that are progressively more complex and have lower false positive rates:

Viola-Jones Face Detector: Results
You Can Try It At Home...

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements

- C++ implementation available in OpenCV (Lienhart, 2002)
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here

Summary: Sliding-Window

- Pros
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes
  - Good detectors available (Viola & Jones, HOG, etc.)

- Cons/Limitations
  - High computational complexity
    - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
    - This puts tight constraints on the classifiers we can use.
    - If training binary detectors independently, this means cost increases linearly with number of classes.
  - With so many windows, false positive rate better be low

Limitations (continued)

- Not all objects are “box” shaped

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

Limitations (continued)

- If considering windows in isolation, context is lost

Example Application

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

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

Limitations (continued)

- With so many windows, false positive rate better be low
Limitations (continued)

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

References and Further Reading

• Read the Viola-Jones paper
  - P. Viola, M. Jones, 
    Robust Real-Time Face Detection, 
    IJCV, Vol. 57(2), 2004, 
    (first version appeared at CVPR 2001)

• Viola-Jones Face Detector
  - C++ implementation available in OpenCV [Lienhart, 2002]
    http://sourceforge.net/projects/opencvlibrary/
  - Matlab wrappers for OpenCV code available, e.g. here
    http://www.mathworks.com/matlabcentral/fileexchange/19912

• HOG Detector
  - Code available: http://pascal.inrialpes.fr/software/olt/