Recap: MRFs for Image Segmentation

- MRF formulation
  - Unary potentials \( \phi(x_i, y_i) \)
  - Pairwise potentials \( \psi(x_i, x_j) \)

\[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i \neq j} \psi(x_i, x_j) \]

Recap: Energy Formulation

- Energy function
  \[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i < j} \psi(x_i, x_j) \]

- Unary potentials \( \phi \)
  - Encode local information about the given pixel/patch
  - How likely is a pixel/patch to belong to a certain class (e.g., foreground/background)?

- Pairwise potentials \( \psi \)
  - Encode neighborhood information
  - How different is a pixel/patch’s label from that of its neighbor? (e.g., based on intensity/color/texture difference, edges)

Recap: How to Set the Potentials?

- Unary potentials
  - E.g., color model, modeled with a Mixture of Gaussians
    \[ \phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_k(x_i, k) p(k|x_i) \mathcal{N}(y_i; \bar{y}_k, \Sigma_k) \]
  - Learn color distributions for each label

- Pairwise potentials
  - Potts Model
    \[ \psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j) \]
    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: "Contrast sensitive Potts model"
    \[ \psi(x_i, x_j; g_{ij}; \theta_\psi) = -\theta_\psi g_{ij}(y_i) \delta(x_i \neq x_j) \]
    where
    \[ g_{ij}(y) = e^{-\beta(y_i - y_j)^2} \]
    - Discourages label changes except in places where there is also a large change in the observations.
Recap: Graph-Cuts Energy Minimization

- Solve an equivalent graph cut problem
  1. Introduce extra nodes: source and sink
  2. Weight connections to source/sink (t-links) by \( \phi(x_t = s) \) and \( \phi(x_t = t) \), respectively.
  3. Weight connections between nodes (n-links) by \( \psi(x_i, x_j) \).
  4. Find the minimum cost cut that separates source from sink.
     \( \Rightarrow \) Solution is equivalent to minimum of the energy.

- s-t Mincut can be solved efficiently
  - Dual to the well-known max flow problem
  - Very efficient algorithms available for regular grid graphs (1-2 MPixels/s)
  - Globally optimal result for 2-class problems

Topics of This Lecture

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

Object Recognition: Challenges

- Viewpoint changes
  - Translation
  - Image-plane rotation
  - Scale changes
  - Out-of-plane rotation
- Illumination
- Noise
- Clutter
- Occlusion

Appearance-Based Recognition

- Basic assumption
  - Objects can be represented by a set of images (“appearances”).
  - For recognition, it is sufficient to just compare the 2D appearances.
  - No 3D model is needed.

\( \Rightarrow \) Fundamental paradigm shift in the 90’s

Global Representation

- Idea
  - Represent each object (view) by a global descriptor.

  - For recognizing objects, just match the descriptors.
  - Some modes of variation are built into the descriptor, the others have to be incorporated in the training data.
    - E.g., a descriptor can be made invariant to image-plane rotations.
    - Other variations:
      - Viewpoint changes
      - Illumination
      - Noise
      - Clutter
      - Out-of-plane rotation
      - Occlusion
Appearance based Recognition

- Recognition as feature vector matching

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

Detection via Classification: Main Idea

- Basic component: a binary classifier

Slide credit: Kristen Grauman
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

Eigenfaces: Global Appearance Description

This can also be applied in a sliding-window framework...

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
  - Localized histograms offer more spatial information than a single global histogram (tradeoff invariant vs. discriminative)
  - Contrast-normalization: try to correct for variable illumination

Histograms of Oriented Gradients (HoG) [Dalal & Triggs, CVPR 2005]

Classifer Construction

- How to compute a decision for each subwindow?

Discriminative Methods

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

Linear Classifiers

Let \( w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \) \( x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \)

\[ w_1 x_1 + w_2 x_2 + b = 0 \]

\[ w^T x + b = 0 \]
Linear Classifiers
- Find linear function to separate positive and negative examples

\[ x_n \text{ positive: } w^T x_n + b \geq 0 \]
\[ x_n \text{ negative: } w^T x_n + b < 0 \]

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

Support Vector Machines
- Want line that maximizes the margin.

\[ x_n \text{ positive (} t_n = 1\text{): } w^T x_n + b \geq 1 \]
\[ x_n \text{ negative (} t_n = -1\text{): } w^T x_n + b < -1 \]

For support vectors, \( w^T x_n + b = \pm 1 \)

Finding the Maximum Margin Line
- Solution: \( w = \sum_{n=1}^{N} a_n t_n x_n \)

\[ \text{Quadratic optimization problem} \]
\[ \text{Minimize} \quad \frac{1}{2} w^T w \]
\[ \text{Subject to} \quad t_n (w^T x_n + b) \geq 1 \]

Finding the Maximum Margin Line
- Classification function:
  \[ f(x) = \begin{cases} 
  \text{sign}(w^T x + b) & \text{if } f(x) < 0, \text{classify as neg.} \\
  \text{sign} \left( \sum_{n=1}^{N} a_n t_n x_n^T x \right) + b & \text{if } f(x) > 0, \text{classify as pos.}
  \end{cases} \]

- Notice that this relies on an inner product between the test point \( x \) and the support vectors \( x_n \).
- (Solving the optimization problem also involves computing the inner products \( x_n^T x \) between all pairs of training points)

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:

  \[ \Phi: x \rightarrow \phi(x) \]

More on that in the Machine Learning lecture...

Some Often-Used Kernel Functions

• Linear:
  \[ K(x_i, x_j) = x_i^T x_j \]

• Polynomial of power p:
  \[ K(x_i, x_j) = (1 + x_i^T x_j)^p \]

• Gaussian (Radial-Basis Function):
  \[ K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]

Nonlinear SVMs

• The kernel trick: Instead of explicitly computing the lifting transformation \( \phi(x) \), define a kernel function \( K \) such that

  \[ K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \]

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

  \[ \sum_a t_a K(x_{ai}, x) + b \]

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

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. Pass 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.

Topics of This Lecture

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

Pedestrian Detection

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

HOG Descriptor Processing Chain

- Optional: Gamma compression
  - Goal: Reduce effect of overly strong gradients
  - Replace each pixel color/intensity by its square-root
    \[ x \mapsto \sqrt{x} \]
  - Small performance improvement
HOG Descriptor Processing Chain

- Gradient computation
  - Compute gradients on all color channels and take strongest one
  - Simple finite difference filters work best (no Gaussian smoothing)
  \[
  \begin{bmatrix}
  -1 & 0 & 1 \\
  -1 & 0 & 1
  \end{bmatrix}
  \]

- Spatial/Orientation binning
  - Compute localized histograms of oriented gradients
  - Typical subdivision: \(8 \times 8\) cells with 8 or 9 orientation bins

HOG Cell Computation Details

- Gradient orientation voting
  - Each pixel contributes to localized gradient orientation histogram(s)
  - Vote is weighted by the pixel’s gradient magnitude
  \[
  \theta = \tan^{-1}\left(\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right) \\
  \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}
  \]

- Block-level Gaussian weighting
  - An additional Gaussian weight is applied to each 2x2 block of cells
  - Each cell is part of 4 such blocks, resulting in 4 versions of the histogram.

HOG Descriptor Processing Chain (2)

- Feature vector construction
  - Collect HOG blocks into vector
  \[\ldots, \ldots, \ldots, \ldots\]

HOG Cell Computation Details (2)

- Important for robustness: Tri-linear interpolation
  - Each pixel contributes to (up to) 4 neighboring cell histograms
  - Weights are obtained by bilinear interpolation in image space:
  \[
  \begin{aligned}
  h(x_1, y_1) &\leftarrow w_1 \left(1 - \frac{x_1 - x_1}{x_2 - x_1}\right) \left(1 - \frac{y_1 - y_1}{y_2 - y_1}\right) \\
  h(x_1, y_2) &\leftarrow w_1 \left(1 - \frac{x_1 - x_1}{x_2 - x_1}\right) \left(\frac{y_2 - y_1}{y_2 - y_1}\right) \\
  h(x_2, y_1) &\leftarrow w_1 \left(\frac{x_2 - x_1}{x_2 - x_1}\right) \left(1 - \frac{y_1 - y_1}{y_2 - y_1}\right) \\
  h(x_2, y_2) &\leftarrow w_1 \left(\frac{x_2 - x_1}{x_2 - x_1}\right) \left(\frac{y_2 - y_1}{y_2 - y_1}\right)
  \end{aligned}
  \]

- Contribution is further split over (up to) 2 neighboring orientation bins via linear interpolation over angles.
HOG Descriptor Processing Chain

- SVM Classification
  - Typically using a linear SVM

Object/Non-object
  - Linear SVM
  - Collect HOGs over detection window
  - Contrast normalize over overlapping spatial cells
  - Weighted vote in spatial & orientation cells
  - Compute gradients
  - Gamma compression

Image Window

Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template

Non-Maximum Suppression

- After multi-scale dense scan
- Clip detection score
- Map each detection to 3D \((x,y,s)\) space
- Apply robust mode detection, e.g. mean shift

Pedestrian detection with HoGs & SVMs

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

References and Further Reading

- Read the HOG paper

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