EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

Lecture 13
Object Recognition
140327

http://www.ee.unlv.edu/~b1morris/ecg795/
Outline

- Knowledge Representation
- Statistical Pattern Recognition
- Neural Networks
- Boosting
Object Recognition

- Pattern recognition is a fundamental component of machine vision.
- Recognition is high-level image analysis:
  - From the bottom-up perspective (pixels → objects).
  - Many software packages exist to easily implement recognition algorithms (E.g. Weka Project, R package).

- Goal of object recognition is to “learn” characteristics that help distinguish object of interest:
  - Most are binary problems.
Knowledge Representation

• Syntax – specifies the symbols that may be used and ways they may be arranged
• Semantics – specifies how meaning is embodied in syntax
• Representation – set of syntactic and semantic conventions used to describe things

• Book focuses on artificial intelligence (AI) representations
  ▫ More closely related to human cognition modeling (e.g. how humans represent things)
  ▫ Not as popular in vision community
Descriptors/Features

- Most common representation in vision
- Descriptors (features) usually represent some scalar property of an object
  - These are often combined into feature vectors
- Numerical feature vectors are inputs for statistical pattern recognition techniques
  - Descriptor represents a point in feature space

Figure 9.1: Recognition of basketball players and jockeys. © Cengage Learning 2015.
Statistical Pattern Recognition

- Object recognition = pattern recognition
  - Pattern – measurable properties of object

Pattern recognition steps:
- Description – determine right features for task
- Classification – technique to separate different object “classes”

- Separable classes – hyper-surface exists perfectly distinguish objects
  - Hyper-planes used for linearly separable classes
  - This is unlikely in real-world scenarios
General Classification Principles

- A statistical classifier takes in a $n$-dimensional feature of an object and has a single output
  - The output is one of the $R$ available class symbols (identifiers)
- Decision rule – describes relations between classifier inputs and output
  - $d(x) = \omega_r$
  - Divides feature space into $R$ disjoint subsets $K_r$
- Discrimination hyper-surface is the border between subsets
- Discrimination function
  - $g_r(x) \geq g_s(x), s \neq r$
    - $x \in K_r$
- Discrimination hyper-surface between class regions
  - $g_r(x) - g_s(x) = 0$

- Decision rule
  - $d(x) = \omega_r \iff g_r(x) = \max_{s=1,\ldots,R} g_s(x)$
  - Which subset (region) provides maximum discrimination
- Linear discriminant functions are simple and often used in linear classifier
  - $g_r(x) = q_{r0} + q_{r1}x_1 + \cdots + q_{rn}x_n$
- Must use non-linear for more complex problems
  - Trick is to transform the original feature space into a higher dimensional space
    - Can use a linear classifier in the higher dim space
  - $g_r(x) = q_r \cdot \Phi(x)$
    - $\Phi(x)$ – non-linear mapping to higher dimensional space
Nearest Neighbors

- Classifier based on minimum distance principle
- Minimum distance classifier labels pattern \( x \) into the class with closest exemplar
  - \( d(x) = \arg \min_s |v_s - x| \)
  - \( v_s \) - exemplars (sample pattern) for class \( \omega_s \)
- With a single exemplar per class, results in linear classifier

- Nearest neighbor (NN) classifier
  - Very simple classifier uses multiple exemplars per class
  - Take same label as closest exemplar
- k-NN classifier
  - More robust version by examining \( k \) closest points and taking most often occurring label

- Advantage: easy “training”
- Problems: computational complexity
  - Scales with number of exemplars and dimensions
  - Must do many comparisons
  - Can improve performance with K-D trees

Figure 9.6: Minimum distance discrimination functions. © Cengage Learning 2015.
Classifier Optimization

• Discriminative classifiers are deterministic
  ▫ Pattern $x$ always mapped to same class
• Would like to have an optimal classifier
  ▫ Classifier that minimizes the errors in classification
• Define loss function to optimize based on classifier parameters $q$
  ▫ $J(q^*) = \min_q J(q)$
  ▫ $d(x, q) = \omega$
• Minimum error criterion (Bayes criterion, maximum likelihood) loss function
  ▫ $\lambda(\omega_r|\omega_s)$ - loss incurred if classifier incorrectly labels object $\omega_r$
    ▪ $\lambda(\omega_r|\omega_s) = 1$ for $r \neq s$
• Mean loss
  ▫ $J(q) = \int_X \sum_{s=1}^R \lambda(d(x, q)|\omega_s)p(x|\omega_s)p(\omega_s)dx$
    ▪ $p(\omega_s)$ - prior probability of class
    ▪ $p(x|\omega_s)$ - conditional probability density
• Discriminative function
  ▫ $g_r(x) = p(x|\omega_r)p(\omega_r)$
  ▫ Corresponds to posteriori probability $p(\omega_r|x)$
• Posteriori probability describes how often pattern $x$ is from class $\omega_r$
• Optimal decision is to classify $x$ to class $\omega_r$ if posteriori $P(\omega_r|x)$ is highest
  ▫ However, we do not know the posteriori
• Bayes theorem
  ▫ $p(\omega_s|x) = \frac{p(x|\omega_s)p(\omega_s)}{p(x)}$
• Since $p(x)$ is a constant and prior $p(\omega_s)$ is known,
  ▫ Just need to maximize likelihood $p(x|\omega_s)$
• This is desirable because the likelihood is something we can learn using training data
Classifier Training

- Supervised approach
- Training set is given with feature and associated class label
  - $T = \{(x_i, y_i)\}$
  - Used to set the classifier parameters $q$
- Learning methods should be inductive to generalize well
  - Represent entire feature space
  - E.g. work even on unseen examples
- Usually, larger datasets result in better generalization
  - Some state-of-the-art classifiers use millions of examples
  - Try to have enough samples to statistical cover space
- N Cross-fold validation/testing
  - Divide training data into a train and validation set
  - Only train using training data and check results on validation set
  - Can be used for “bootstrapping” or to select best parameters after partitioning data N times
Classifier Learning

- Probability density estimation
  - Estimate the probability densities \( p(x|\omega_r) \) and priors \( p(\omega_r) \)

- Parametric learning
  - Typically, the distribution \( p(x|\omega_r) \) shape is known but the parameters must be learned
    - E.g. Gaussian mixture model
  - Like to select a distribution family that can be efficiently estimated such as Gaussians
  - Prior estimation by relative frequency
    - \( p(\omega_r) = K_r/K \)
      - Number of objects in class \( r \) over total objects in training database
Support Vector Machines (SVM)

- Maybe the most popular classifier in CV today
- SVM is an optimal classification for separable two-class problem
  - Maximizes the margin (separation) between two classes → generalizable and avoids overfitting
  - Relaxed constraints for non-separable classes
  - Can use kernel trick to provide non-linear separating hyper-surfaces
- Support vectors – vectors from each class that are closest to the discriminating surface
  - Define the margin
- Rather than explicitly model the likelihood, search for the discrimination function
  - Don’t waste time modeling densities when class label is all we need
SVM Insight

- SVM is designed for binary classification of linearly separable classes
- Input $\mathbf{x}$ is $n$-dimensional (scaled between $[0,1]$ to normalize) and class label $\omega \in \{-1,1\}$
- Discrimination between classes defined by hyperplane s.t. no training samples are misclassified
  - $\mathbf{w} \cdot \mathbf{x} + b = 0$
    - $\mathbf{w}$ – plane normal, $b$ offset
  - Optimization finds “best” separating hyperplane

Figure 9.9: Basic two-class classification idea of support vector machines. (a) and (b) show two examples of non-optimal linear discrimination. (c) An optimal linear discriminator maximizes the margin between patterns of the two classes. The optimal hyperplane is a function of the support vectors. © Cengage Learning 2015.
# SVM Power

- Final discrimination function
  - $f(x) = w \cdot x + b$

- Re-written using training data
  - $f(x) = \sum_{i \in SV} \alpha_i \omega_i (x_i \cdot x) + b$
    - $\alpha_i$ - weight of support vector SV
  - Only need to keep support vectors for classification

- Kernel trick
  - Replace $(x_i \cdot x)$ with non-linear mapping kernel
    - $k(x_i, x) = \Phi(x_i) \cdot \Phi(x_j)$
  - For specific kernels this can be efficiently computed without doing the warping $\Phi$
    - Can even map into an infinite dimensional space
  - Allows linear separation in a higher dimensional space

---

**Figure 9.10:** Achieving linear separability by application of a kernel function. On the left, the two classes are not linearly separable in 1D; on the right, the function $\Phi(x) = x^2$ creates a linearly separable problem. © Cengage Learning 2015.

**Figure 9.11:** Support vector machine training; Gaussian radial basis function kernel used (equation 9.48). (a,c) Two-class pattern distribution in a feature space. (Note that the “+” patterns in (a) and (b) are identical while the “o” patterns in (a) are a subset of patterns in (b)). (b,d) Non-linear discrimination functions obtained after support vector machine training. © Cengage Learning 2015.
SVM Resources

• More detailed treatment can be found in
  ▫ Duda, Hart, Stork, “Pattern Classification”
• Lecture notes from Nuno Vasconcelos (UCSD)
  ▫ [link]
    ▫ http://www.svcl.ucsd.edu/courses/ece271B-F09/handouts/SVMs.pdf

• SVM software
  ▫ LibSVM [link]
  ▫ SVMLight [link]
Cluster Analysis

- Unsupervised learning method that does not require labeled training data
- Divide training set into subsets (clusters) based on mutual similarity of subset elements
  - Similar objects are in a single cluster, dissimilar objects in separate clusters
- Clustering can be performed hierarchically or non-hierarchically
- Hierarchical clustering
  - Agglomerative – each sample starts as its own cluster and clusters are merged
  - Divisive – the whole dataset starts as a single cluster and is divided
- Non-hierarchical clustering
  - Parametric approaches – assumes a known class-conditioned distribution (similar to classifier learning)
  - Non-parametric approaches – avoid strict definition of distribution
K-Means Clustering

- Very popular non-parametric clustering technique
  - Based on minimizing the sum of squared distances
    \[ E = \sum_{i=1}^{K} \sum_{x_j \in V_i} d^2(x_j, v_i) \]
  - Simple and effective
- K-means algorithm
  - Input is \( n \)-dimensional data points and number of clusters \( K \)
  - Initialize cluster starting points
    \[ \{v_1, v_2, \ldots, v_K\} \]
  - Assign points to closest \( v_i \) using distance metric \( d \)
  - Recompute \( v_i \) as centroid of associated data \( V_i \)
  - Repeat until convergence
K-Means Demo

- http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Neural Networks

- Early success on difficult problems
  - Renewed interest with deep learning
- Motivated by human brain and neurons
  - Neuron is elementary processor which takes a number of inputs and generates a single output
- Each input has associated weight and output is a weighted sum of inputs

The network is formed by interconnecting neurons
- Outputs of neurons as inputs to others
- May have many inputs and many outputs

NN tasks:
- Classification – binary output
- Auto-association – re-generate input to learn network representation
- General association – associations between patterns in different domains
NN Variants

- Feed-forward networks
  - Include “hidden” layers between input and output
  - Can handle more complicated problems
- Networks “taught” using back-propagation
  - Compare network output to expected (truth) output
  - Minimize SSD error by adjusting neuron weight
- Kohonen feature maps
  - Unsupervised learning that organizes network to recognize patterns
  - Performs clustering
    - Neighborhood neurons are related
  - Network lies on a 2D layer
    - Fully connect neurons to all inputs
    - Neuron with highest input
      - \[ x = \sum_{i=1}^{n} v_i w_i \]
      - is the winner (cluster label)
Boosting

- Generally, a single classifier does not solve problem well enough
  - Is it possible to improve performance by using more classifiers (e.g. experts)?
- Boosting – intelligent combination of weak classifiers to generate a strong classifier
  - Weak classifier works a little better than chance (50% for binary problem)
  - Final decision rule combines each weak classifier output by weighted confidence majority vote
    - \( C(x) = \text{sign}(\sum_i \alpha_i C_i(x)) \)
    - \( \alpha_i \) - confidence in classifier \( C_i(.) \)
- Training
  - Sequentially train classifiers to focus classification effort on “hard” examples
  - After each training round, re-weight misclassified examples

Advantages:
- Generally, does not overfit but is able to achieve high accuracy
  - Training rounds increase margin
- Many modification exist to improve performance
  - Gentle and BrownBoost for outlier robustness
  - Strong theoretical background
- Flexible with only “weak” classifier requirement
  - Can use any type of classifier (statistical, rule-based, of different type, etc.)