Learning and Classification

a Really brief introduction

Motivation: almost all computer vision algorithms use some learning.

Classification (Labeling) tasks

Many vision tasks are decisions between few option:
- Edge or not (in some point p)?
- Does the image contain an object from a category list?
- Are two pixels, p1, p2, in the same segment?
- Does point p1 in Image1, corresponds to p2 in Image2?

How to formalize this decision procedure?
Classification: $\phi(X) \rightarrow Y$

$X$ - image data (feature vector, continuous).
$Y$ - class label (a few options, discrete)

Classification: partition the $X$ space into (a few) regions
Probabilistic models

Let $x$ be the measurement. $x$ depends on:
- A discrete variable (label) $Y$ of interest (e.g. edge/non-edge)
- Other factors

How to express the variation?
A popular choice: probabilistic modeling

- $x$ is modeled as a random variable (r.v.) distributed by
  \[ p_Y(x) \]

Example (board): height distributions, uncertainty.

Bayesian Decisions

Underlying assumptions:
1. The object is selected randomly from the different classes with known probability $p(Y)$ (prior).
2. The distributions of $X$ for each class $Y$ (likelihood) \( p_Y(x) = p(x | Y) \) are also known.

Calculate posterior probability
\[ p(Y_j | x) = \frac{p(x | Y_j) p(Y_j)}{p(x)} \]

Minimize expected decision error
\[ p(error | x) = 1 - p(Y_{chosen} | x) \]
by choosing \[ y^* = \arg \max_y p(Y | x) = \arg \max_y p(x | Y) p(Y) \]

Bayesian decision (Statistical decision theory)
Estimating likelihoods and priors

For making Bayesian decisions we need to know

\[ p(y) \quad p_y(x) = p(x \mid y) \]

Sources of knowledge

- (Knowing the "physics")
- Examples \( \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)\} \)
  - available,
  - characteristic of the distributions

1. Parametric estimation from examples
   - E.g. Assume \( p_{\mathcal{y}_0} (x) \) is Gaussians, and estimate \( (\mu_{\mathcal{y}_0}, \Sigma_{\mathcal{y}_0}) \)
2. Nonparametric estimate from examples
   - E.g. \( p_{\mathcal{y}_0} (x) = c \sum_{(x_i, y_0)} G(||x - x_i||) \)

Learning from examples - Generative

Learning: generating a classifier from examples

The Generative approach

- Training: Estimate likelihood/prior functions from data
- Classification ("Testing"): Use Bayesian decision

Examples: (different ways to represent likelihoods)

- Gaussian mixture models
- Bayesian networks

(-) Need model of the distribution
(+), Provides decision certainty, allows varying priors
Learning from examples - Discriminative

The discriminative approach
- Training: Direct estimation of the decision function or the posterior probability from examples $\phi(X)$ or $P(Y|X)$
- Classification: Use the decision function ...

Examples: Support vectors machine, Perceptrons, k-nearest neighbors, Decision trees, Ada-boost, (Deep/convolutional) neural networks, ...

(+) Does not need a distribution model
(+) When there is no model, usually works better.
(+) Does not learn what is not essential

Image Categorization

[Diagram showing the process of image categorization with steps for training and testing]
Discriminative Learning

Given a set of training examples, how should we choose the classification function $\phi(X) \rightarrow Y$? Equivalently how should we choose the partition?

What do we want?

- Low training error rate - Classify all/most training examples correctly (consistency).
- Low test error rate - Classify unseen (test) examples correct (generalization).

But ... many classification functions can be consistent

Bias Variance Tradeoff

Many classification functions $\phi(X) \rightarrow Y$ are consistent with the training set. Which one to choose?

General rules:

- Simple classifier - low consistency, better generalization, work better with a few training examples
- Complex classifier - high consistency, low generalization, needs many examples.
Discriminative Learning Algorithms

Different algorithms: Different methods for specifying the classification function

Some examples - basic algorithms and some related to important vision algorithms.

Discriminative Learning - I

• Which decision function classify this training set correctly?

• Simplest form: "stub"
  - Choose one feature
  - Decide by thresholding

Simplistic but useful as a building block
**Discriminative Learning - I**

- Linear decision functions
  \[ D(x) = a^T x + b \geq 0 \Rightarrow x \]
  = linear separation

- Training = finding \( a, b \)
- Trained from examples by
  - Perceptron algorithm
  - Linear programming
  - Logistic regression
  - An important building block
  - Advanced linear classifier - Support Vector Machine (SVM)

---

**Feature Selection in Learning**

What if the space is not linearly separable?

- Choose other features
- Generate new features from old ones, explicitly (e.g. Lifting to higher dimension)

- Learn how to generate features, implicitly - done in most alg., e.g. Kernel SVM and Deep Neural Networks.
Discriminative Learning - II

Decision tree
- Old, established, fast
- Works for symbolic features but not only

Training involves finding the thresholded feature and the threshold in each node. (board)

Discriminative Learning - III

1-nearest neighbors
- "Lazy" learning
- Heuristic but theoretically, its asymptotic error is just twice that of Bayesian

• Training is not needed.
Discriminative Learning - IV

Decision by committee/ensemble
- Context: many "weak classifiers" are available
- Goal: merge them to one better classifier

- "Forests" of decision trees
  - Train trees of subsets of data and subsets of features
  - Classify by each tree and then majority vote.

- The Adaboost algorithm - Adaptive Boosting
  - Principle - choose independent weak classifiers
  - Change the weight of examples during learning
  - Final classifier = weight sum of weak classifier
  - Will be described later
Neural network perceptron

- Recall the linear decision functions / perceptron
  \[ D(x) = a^T x + b \geq 0 \Rightarrow \times \]

- Using other notations, and graphical description
  \[ \varphi(w,x) = \text{sign}\left( \sum_i w_i x_i + b \right) \]

- The perceptron is the simplest neural network

Motivation from biology

Rough description of the brain:
- About 10,000,000,000 neural cells
- Every neural cell collect electrical/chemical activities from others using dendrites, and "fires" if the sum is large enough.
Multilayer perceptron

The main idea:
- Build several perceptrons and use their output as inputs to another set of perceptrons
- Training – adjusting all the weights together so that, given an example’s input $X$, the network’s output is $Y$.

- Training by gradient descent (backpropagation alg.)
- Feedforward
- Fully connected
- Several/many hidden layers
- Theoretically: can be any function

Multilayer perceptron

Problems:
- Large tasks $\rightarrow$ a large number of weights $\rightarrow$ requires many examples for generalization, and large computational effort.
- Problem: Training is an optimization of a nonlinear functions, with many local minima.

Solutions: In last few years, however
- Simpler networks (locally connected = convolutional)
- Different nonlinearities & better training algorithms
- Stronger computers
- Richer annotated data sets made learning much better