CSE 252C: Computer Vision III

Lecturer: Serge Belongie
Scribe: Catherine Wah

LECTURE 14
Generative Models

14.1. Introduction

The last part of the course is on techniques of statistical pattern recognition relevant to object recognition. Today we'll talk about "generative models," in contrast to "discriminative models"(for recognition). The distinction between the two is that generative models explicitly or implicitly model the distribution of inputs as well as outputs; by sampling from them, it is possible to generate synthetic data points in the input space.

For example, you could have a generative model of each handwritten digit from 0 to 9 (pixel brightnesses or pen strokes) and use it to determine the probability each one could have generated a novel test example. If we call the classes C_k, $k = 0, \ldots, 9$, then the generative model expresses the posterior class probability (using Bayes rule):

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}, \tag{14.1}$$

where $p(C_k)$ are the prior class probabilities, $p(\boldsymbol{x}|C_k)$ is the class conditional densities, and $p(\boldsymbol{x}) = \sum_k p(\boldsymbol{x}|C_k)p(C_k)$ is the marginal density. The vector

[1]Department of Computer Science and Engineering, University of California, San Diego.

December 31, 2009

$\boldsymbol{x}$ is the pattern (*i.e.* , the image, or a description of the image). In our early experiments with digit classification we encountered $p(k)$ (which was almost uniform in that case) but didn't use it; the distance based classification methods we used didn't look at any probabilities (unless we think of k-NN voting as a probability).

More generally, methods that skip the part about modeling class conditional densities are known as "discriminative models," discussed later in the course. At a high level, which type of approach is better is a matter of some debate.

There are several potential advantages of generative models. For training, only positive examples are needed. Also, these models allow for synthesis, that is, we can "look under the hood" at what is learned. Lastly, $p(\boldsymbol{x})$ allows "novelty detection," *i.e.* , the data points with low probability under the model, for which predictions may be of low accuracy.

14.2. Mixture of Gaussians (MoG)

Now let's look at an example of a generative model, and how to learn its parameters from training data. One of the most widely used models is the *Gaussian mixture*:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{14.2}$$

where K is the number of components or "modes," which could be the number of classes, or a finer distinction. The priors are represented with π_k or $p(C_k)$, and $\mathcal{N}(\cdot)$ is the normal density with mean $\boldsymbol{\mu}$ and covariance matrix Σ.

For example, one can use MoG to to model content in an image, as in Blobworld (1998), as an alternative to nonparametric density estimation. In this case, the feature vector $\boldsymbol{x}$ could contain position, color, texture, etc. for a pixel.

14.3. The EM algorithm

Given some training data, how do we "fit the model"? That is, how do we find values of π_k, $\boldsymbol{\mu}_k$, and Σ_k? We formulate this as a "missing data" or "latent variable" problem, by introducing a soft indicator vector $\boldsymbol{z} \in \mathbb{R}^k$, $z_k \in \{0, 1\}$, $\sum_k z_k = 1$. We'll assume for now that k is known, but just as in clustering (to which the approach we study here is closely related), choosing k is a difficult problem in itself.

In particular, $p(z_k = 1) = \pi_k$. What does it gain us to introduce this new vector? It turns out that z will allow us to set up a chicken-and-egg problem in which we can express the mixture density as if we had complete data.

The method we use to solve this problem is called "Expectation- Maximization" or "The EM Algorithm" (Dempster, Laird, and Rubin, 1977). We will skip the derivation and outline its application to MoG.

First, let's look at the steps in abstract:

$$\begin{aligned} \text{Given} &: \text{a joint distribution } p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) \\ \text{Goal} &: \text{maximize } p(\boldsymbol{X}|\boldsymbol{\theta}) \text{ w.r.t. } \boldsymbol{\theta}, \text{ a.k.a. the likelihood.} \end{aligned}$$

$\boldsymbol{X}$ is all observed data, θ is all parameters, and $\boldsymbol{Z}$ is all the latent variables. Note that we will have to perform gradient descent on the likelihood, which is greedy, so we can only hope for a local optimum.

1. Initialize $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^{old}$
2. Expectation step: evaluate $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old})$
3. Maximization step: evaluate $\boldsymbol{\theta}^{new}$ as

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$$

   where

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old}) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}).$$

   By convention, we work with the log likelihood.
4. Check for convergence, either of the log likelihood or parametric values; if not satisfied, let $\boldsymbol{\theta}^{old} \leftarrow \boldsymbol{\theta}^{new}$ and return to step 2.

Now let's apply this to MoG: in this case, the "complete data" likelihood reduces to:

$$p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)^{z_{nk}} \tag{14.3}$$

with N data points and K components, where z_{nk} denotes the kth component of z_n. In the discrete $\{0, 1\}$ case, it selects out exactly one component per data point; when relaxed, you get a soft, convex combination.

Now take the log:

$$\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}[\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)]. \tag{14.4}$$

The second log term on the right is convenient as $\mathcal{N}(\cdot)$ is a member of the exponential family.

Since we don't know the true values of the latent variables $\boldsymbol{Z}$, we instead use their expected value, denoted $\gamma(z_{nk})$, which one can show is given by:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}, \tag{14.5}$$

a.k.a. the "responsibilities."

Now let's look a the above four steps for MoG:

1. Initialize $\boldsymbol{\mu}_k$, Σ_k, π_k for all k
2. Expectation step: evaluate responsibilities $\gamma(z_{nk})$ using current parameter values
3. Maximization step: re-estimate parameters:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{x}_n$$

$$\Sigma_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^{\mathsf{T}}$$

$$\pi_k^{new} = \frac{N_k}{N} \text{ where } N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

4. Evaluate log likelihood:

$$\ln p(\boldsymbol{X}|\boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)\right\}$$

and check for convergence; if necessary, return to step 2.

An exercise for the reader: show that the k-means algorithm is equivalent to a hard-thresholded version of EM applied to MoG, where all $\sum$'s are ϵI an $\epsilon \to 0$.

Cases with more interesting density shapes, *e.g.* , an annulus, prompt the question of how suitable MoG might be for such distributions; this foreshadows kernel-based methods.

14.4. Hidden Markov Model

Another important model is something we encounter when we have sequential data, *e.g.* , a time series, or text (*e.g.* , OCR) scanned from left to right: the *Hidden Markov Model* (HMM). Consider the plot of clumps, and now generalize it to include "time" steps. Visualize this as a point hopping around with transition probabilities defined over k modes, each approximately a

Gaussian in this case. Alternatively, we can visualize it as a graphical model (Figure 1).

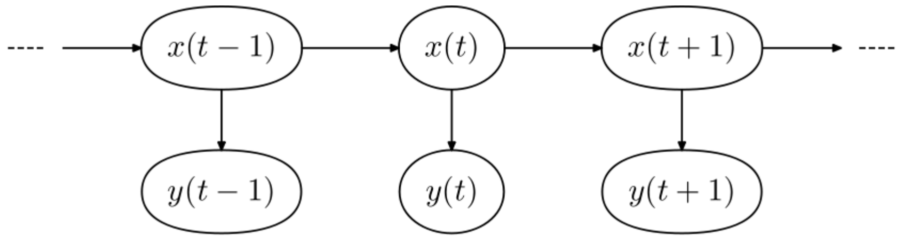

Figure 1. Graphical model of a Hidden Markov Model. The value of the observed variable $y(t)$ only depends on the value of the hidden variable $x(t)$. (http://en.wikipedia.org/wiki/Hidden_Markov_model)

HMMs are very widely used in speech recognition and handwriting recognition. We can again apply the EM algorithm, now with the extra requirement of estimating transition probabilities; in this context, it is known as the "forward-backward" or "Baum-Welch" algorithm. The hard thresholded version is called the "Viterbi algorithm."

Recall that model selection is a difficult problem; however, if adequate domain knowledge is available to do this, then this can be highly effective. We can train up one generative model per class and check the likelihood of the data w.r.t. each model. Next class we will cover discriminative models.