Book: Model, Learning, and Inference: Lecture 3

Alan Yuille
Department of Statistics, UCLA
Los Angeles, CA 90095
yuille@stat.ucla.edu

Abstract

Learning MRF's

1 Introduction

The previous lecture discussed the weak membrane model. But how can the model be learnt from data? Many empirical studies have shown that the distribution of derivative filters (e.g., $d/dx, d^2/dx^2, ...$) have a characteristic shape (which is not Gaussian). The distribution is strongly peaked about 0 and falls off rapidly at first and then more slowly as the magnitude of the derivative increases. This differs from a Gaussian distribution – which falls off more slowly at first and then more rapidly as the magnitude increases – but it is, intuitively, what you would expect if images are piecewise smooth. But how to relate the histograms of derivative filters to the piecewise smooth model more precisely?

In this lecture, we first review background material on exponential models and then present the work of Zhu, Wu, Mumford and Della Pietra et al. The connection between the weak membrane model and the derivative statistics is made in (Zhu and Mumford). This relationship applies to other signals – e.g., depth and lines – and not just to images (similar statistics appear in many physical domains – Mark Green).

2 Exponential Distributions

Almost all distributions (e.g. Gaussian, Poisson, etc.,) can be expressed as exponential models of form:

$$P(\vec{x}|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}, \tag{1}$$

where $\vec{\phi}(\vec{x})$ denotes the *statistics* and $\vec{\lambda}$ the *parameters*. The normalization term $Z[\vec{\lambda}] = \sum_{\vec{x}} \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}$ (replace $\sum$ by $\int$ if $\vec{x}$ is continuous valued). The derivative of the logarithm of the normalization term is the expectation of the statistics:

$$\frac{\partial}{\partial \vec{\lambda}} \log Z[\vec{\lambda}] = \sum_{\vec{x}} \vec{\phi}(\vec{x}) P(\vec{x}|\vec{\lambda}). \tag{2}$$

Learning a distribution of this type from training data $\{\vec{x}_1, ..., \vec{x}_N\}$ can be formulated as Maximum Likelihood (ML) estimation of the parameter $\vec{\lambda}$:

$$\vec{\lambda}^* = \arg\max_{\vec{\lambda}} P(\{\vec{x}_i\}|\vec{l}ambda) = \arg\min_{\vec{\lambda}} \{\log Z[\vec{\lambda}] - \vec{\lambda} \cdot \vec{\psi}, \tag{3}$$

where $\vec{\psi} = \frac{1}{N}\sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$.

It can be checked that $\log Z[\vec{\lambda}] - \vec{\lambda}\cdot\vec{\psi}$ is a convex function of $\vec{\lambda}$ (because its Hessian $\frac{\partial^2}{\partial\vec{\lambda}\partial\vec{\lambda}} \log Z[\vec{\lambda}]$ is positive definite – as can be seen by using the Cauchy-Schwartz inequality).

For well-known (brand name) distributions – like Gaussian and Poisson – the solution $\vec{\lambda}^*$ can be computed analytically. But for Markov Random Field (define somewhere!!) the solution is difficult to find. The fact that $\log Z[\vec{\lambda}] - \vec{\lambda}\cdot\vec{\psi}$ is convex means that there are several algorithms (steepest descent, Generalized Iterative Scaling) which are guaranteed to decrease this function and end up at the global minima (the solution). But the problem is that all these algorithms involve computing terms like $\sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}|\vec{\lambda})$ which is often impractical due to the difficultly of summing over $\vec{x}$ (the same difficulty makes computing $Z[\vec{\lambda}]$ very difficult. For example here are two algorithms:

$$\vec{\lambda}^{t+1} = \vec{\lambda}^t - \sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}|\vec{\lambda}^t) + \vec{\psi}, \ \text{steepest descent}, \tag{4}$$

$$\vec{\lambda}^{t+1} = \vec{\lambda}^t - \log\sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}|\vec{\lambda}^t) + \log\vec{\psi}, \ \text{generalized iterative scaling}. \tag{5}$$

Each algorithm requires taking the expectation of $\vec{\phi}(\vec{x})$ with respect to the current model specified by $\vec{\lambda}^t$. If this expectation agrees with the observed statistics $\vec{\psi}$ then the model has converged to the correct parameters values. If not, then the parameters need to be adjusted to give better agreement.

This shows that the relation between learning and inference is tightly coupled. In this case, inference corresponds to estimating $\vec{x}^* = \arg\max_{\vec{x}} P(\vec{x}|\vec{\lambda}) = \arg\max_{\vec{x}} \vec{\lambda}\cdot\vec{x}$ while learning requires the ability to compute $\sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}|\vec{\lambda})$. Inference requires a *maximum computation* while learning requires a *summation computation*. We will return to these learning and inference issues later.

3 Weak Membrane and Image Derivative Statistics

The image derivative statistics $h(\vec{x}; z)$ (note that z corresponds to the coefficient of the vector $\vec{\phi}(\vec{x})$) of an image $\vec{x}$ can be represented by a histogram (other distributions are possible):

$$h(\vec{x}; z) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_{i+1}-x_i, z}, \tag{6}$$

where $\delta_{a,b}$ is the delta function (i.e., $\delta_{a,b} = 1, \text{ if } a = b$ and $\delta_{a,b} = 0, \text{ if } a \neq b$).

The distribution is given by:

$$P(\vec{x}|\lambda) = \frac{1}{Z[\lambda]} \exp\{\sum_z \lambda(z) h(\vec{x}; z)\},$$
$$= \frac{1}{Z[\lambda]} \exp\{\sum_z \sum_{i=1}^{N} (1/N)\lambda(z)\delta_{x_{i+1}-x_i, z}\},$$
$$= \frac{1}{Z[\lambda]} \exp\{(1/N)\sum_{i=1}^{N} \lambda(x_{i+1}-x_i)\}. \tag{7}$$

Hence by using the derivative histogram as the model statistics we have recovered a Markov Random Field (MRF) model with nearest neighbor interactions. See graph figures. Observe that using the histogram of the statistics of the second order derivative d^2/dx^2 would give a model with interactions to pairs of neighbors. *Hence the graph representation of the distribution is determined by the statistics.*

To determine the value of the potentials $\vec{\lambda}(.)$ we have to observe the statistics of the training image dataset $h(z)$ and solve for $\vec{\lambda}^*$ so that $\sum_{\vec{x}} h(\vec{x}; z)P(\vec{x}|\vec{\lambda}^*) = h(z)$. As described in the previous section, this is conceptually easy but often very difficult in practice. We will return to it later.

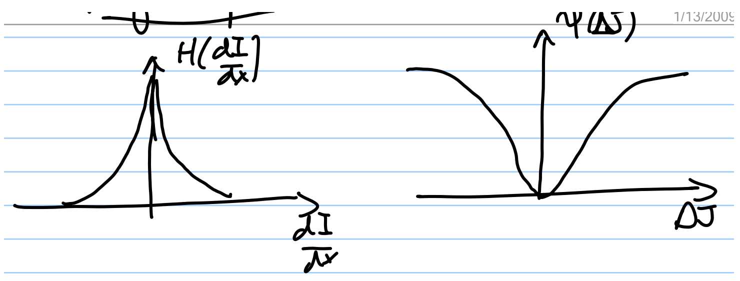

Figure 1: Left Panel: the typically histogram observed for derivative filters. Right panel: the corresponding potential obtained by maximum likelihood learning.

For this example, there is a simple approximation (Coughlan Yuille) that can be used to determine an analytic form for the potentials $\lambda(.)$ in terms of the observed statistics $h(.)$. This approximation requires that the distribution of derivative filters at different pixels in the image are independent, *when the distribution of the image is independent random noise* (per pixel). This is only an approximation (because even for random noise images there is some dependence of the derivatives at neighboring pixels in the image), but it is not bad (note that this approximation is generally poor for natural images which have long range dependencies between derivatives).

Assuming this approximation, it can be shown (Coughlan Yuille) that

$$\lambda(z) = \log \frac{h(z)}{a(z)}, \tag{8}$$

where $a(z)$ is computed from the derivative filter response on random noise images.

This shows the relations between the image histograms and the resulting potential – see figure (1).

4 Selecting Features: Minimax and Della-Pietra

What if we do not know the features $\phi(x)$? One strategy is to specify a dictionary of features $\{\vec{\phi}^{\mu}(.)\}$ and assign parameters $\{\vec{\lambda}_\mu\}$ to all of them:

$$P(\vec{x}|\{\vec{\lambda}_\mu\}) = \frac{1}{Z[\{\vec{\lambda}_\mu\}]} \exp\{\sum_\mu \vec{\lambda}_\mu \cdot \vec{\phi}_\mu(\vec{x})\}. \tag{9}$$

Then search for which features to use by greedy pursuits. Initialize $\vec{\lambda}_\mu = 0, \ \forall \mu$. Define

$$F[\{\vec{\lambda}_\mu\}] = \log Z[\{\vec{\lambda}_\mu\}] - \sum_\mu \vec{\lambda}_\mu \cdot \vec{\phi}_\mu \tag{10}$$

Perform coordinate descent and minimize with respect to each $\vec{\lambda}_\mu$. Solve for $Z[\{\vec{\lambda}\}]$. Select the feature that gives the biggest decrease. Then repeat. The only problem with this is the amount of computation required.

This approach is sometimes called minimax entropy (Zhu, Wu, Mumford) (Della Pietra et al. published an earlier paper and called it something else). This is for two reasons.

Firstly, the exponential distributions can be obtained from the maximum entropy principle (Jaynes). Suppose we have statistics $\vec{\phi}(\vec{x})$ whose observed value is $\vec{\psi}$. This is not enough to uniquely determine a probability distribution. Instead we use the distribution with minimal entropy which has the same expected statistics – this is an optimization problem.

$$\text{maximize wrt} P(\vec{x}) \quad -\sum_{\vec{x}} P(\vec{x}) \log P(\vec{x}) + \tau\{\sum_{\vec{x}} P(\vec{x}) - 1\} + \vec{\lambda}\{\sum_{\vec{x}} \vec{\phi}(\vec{x}) P(\vec{x}) - \vec{\psi}\}, \tag{11}$$

where τ and $\vec{\lambda}$ are Lagrange multipliers used to impose the constraints $\sum_{\vec{x}} P(\vec{x}) = 1$ and $\sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}) = \vec{\psi}$.

This gives the maximum entropy solution $P(\vec{x}) = \frac{1}{Z[\lambda]} \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}$ where $\vec{\lambda}$ is chosen to ensure that $\sum_{\vec{x}} \vec{\phi}(\vec{x})P(\vec{x}) = \vec{\psi}$. Without constraints, the maximum entropy distribution is the uniform distribution.

Secondly, feature selection corresponds to picking the distribution with smallest entropy. The entropy of the distribution is:

$$H[\{\vec{\lambda}\}] = -\sum_{\vec{x}} P(\vec{x}|\{\vec{\lambda}_\mu\}) \log P(\vec{x}|\{\vec{\lambda}_\mu\}) = \log Z[\{\vec{\lambda}_\mu\}] - \sum_\mu \vec{\lambda}_\mu \sum_{\vec{x}} \vec{\phi}_\mu(\vec{x}) P(\vec{x}|\{\vec{\lambda}_\mu\}) = \log Z[\{\vec{\lambda}_\mu\}] - \sum_\mu \vec{\lambda}_\mu \cdot \vec{\psi}_\mu. \tag{12}$$

Hence the entropy of the distribution (after fitting it to the data) is equal to the negative loglikelihood of the data. So minimizing the entropy maximizes the loglikelihood. So *minimax entropy* first uses the maximum principle to get the exponential form of the distribution and then uses the minimum entropy principle to perform maximum likelihood.

Zhu and his collaborators have applied minimax entropy to modeling: (i) natural images, (ii) texture images, (iii) depth, and (iv) curves.

The paper by Della Pietra et al. is similar but is applied to language and not to vision. It uses words made from strings of text. The first set of potentials are the unary statistics of the frequencies of letters. The second set of statistics are the statistics of pairs of letters.