Lecture 3: Stat 238. Winter. 2012

A.L. Yuille

2012-01-20

1 Lecture 3

1. Review of Linear Filtering.
2. Non-linear filtering. Invariance.
3. Graphical Model Picture.
4. Labeling.
5. Representations – within-class and between-class.

2 Linear Filtering: Review

Denote images as $I(i,j)$ or $I(x,y)$ in continuous way. Linear filtering:

$$(F \star I)(x,y) = \int\int F(x-\mu, y-\nu) I(\mu,\nu) \mathrm{d}\mu \mathrm{d}\nu \tag{1}$$

or

$$(F \star I)(i,j) = \sum_k \sum_l F(i-k, j-l) I(k,l) \tag{2}$$

The purpose of linear filters:

- To smooth images, e.g. if F is a Gaussian filter.
- To enhance some image properties, e.g. use derivative of Gaussian to enhances edge information.
- To provide multi-scale descriptors. Smooth the images at different scales $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$ to get images $I_{\sigma_1}, I_{\sigma_2}, I_{\sigma_3}, I_{\sigma_4}$ which represent the images at different scales. I.e. small scale structure gets smoothed out so I_{σ_4} represents the image at coarse level.
- To provide a local representation of images by filterbanks.

3 Non-Linear Filters

There are a large variety of non-linear filters. In this section we focus on a special class of non-linear filters which obtain descriptions of images that are, to some extent, invariant to image transformation, e.g. illumination variance and spatial deformations. These class of non-linear filters include SIFT, HOG, and other alternatives. There are particularly useful for object recognition.

A simple example is the following photometric transformation:

$$I(x,y) \mapsto aI(x,y) + b \tag{3}$$

where the intensity of the image I is scaled by a factor a and then shifted by a factor b. To reduce the effects of a and b, a common strategy is to apply *differentiation* and *normalization* to I. Differentiation is computed by:

$$\nabla I(x,y) = \left(\frac{\partial}{\partial x} I(x,y), \frac{\partial}{\partial y} I(x,y) \right) \tag{4}$$

By taking a derivative we remove the factor b. To remove the factor a, we choose a domain $\mathcal{D}(x,y)$ centered on a pixel (x,y), and divide the gradient by $\sum_{(x',y') \in \mathcal{D}(x,y)} |\nabla I(x',y')|$.

Hence the non-linear quantity $\nabla I(x,y) / \sum_{(x',y') \in \mathcal{D}(x,y)} |\nabla I(x',y')|$ is independent of the photometric transformation.

Next we want to compute a quantity which is invariant to spatial transformations. We use the histogram of $\nabla I(x,y) / \sum_{(x',y') \in \mathcal{D}(x,y)} |\nabla I(x',y')|$ within $\mathcal{D}(x,y)$. The histogram is a measure of the statistics within an image region. It discards the spatial positions and hence gives a representation invariant to some spatial deformations.

Now we develop the mathematics for histograms. Suppose $\nabla I \in \mathcal{F}$. The idea is to partition $\mathcal{F}$ into M disjoint subsets (or bins)

$$\mathcal{F} = \cup_{i=1}^{M} \mathcal{F}_i, \quad \mathcal{F}_i \cap \mathcal{F}_j = \emptyset, \text{ for } i \neq j \tag{5}$$

and count how many filter responses fall into each bin. In more detail, consider the image derivatives $\nabla I = |\nabla I|(\cos\theta, \sin\theta)$ as example (i.e. drop the normalization to simply the mathematics). We can compute separate histogram bins for the angles θ:

$$F_n = \{\theta : \frac{n 2\pi}{M} \leq \theta \leq \frac{(n+1)2\pi}{M}\}, \quad n = 1, \ldots, N \tag{6}$$

and for the magnitudes:

$$G_m = \{\theta : mK \leq |\nabla I| \leq (m+1)K\}, \quad m = 1, \ldots, M \tag{7}$$

where $MK = \max |\nabla I|$.

We can also put all of them together to form two-dimensional histogram bins:

$$H_{n \times m} = \{\theta, |\nabla I| \mid \frac{n 2\pi}{M} \leq \theta \leq \frac{(n+1)2\pi}{M}, mK \leq |\nabla I| \leq (m+1)K\} \tag{8}$$

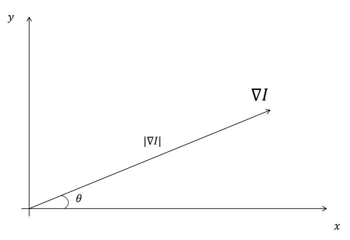

Figure 1: Sketch map for gradient

Where the histogram is given by the bin counts:

$$C_{n,m}(\mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} \mathbb{I}\left[\frac{\nabla I(x,y)}{\sum_{(x',y')\in\mathcal{D}} |\nabla I(x',y')|} \in H_{n,m}\right] \tag{9}$$

We put these together – i.e. at each pixel (x,y) take a local histogram of the normalized gradient of the intensity. This gives a non-linear filter which is invariant to the photometric transformations defined above and to some spatial deformations.

More complicated non-linear filters, sharing the same main ideas with the above examples, are SIFT[1] and HOG[2].

4 Edge Detection and the Bigger Picture

How does the edge detection example fit into the bigger picture of probability distributions defined on graphs, see figure (2).

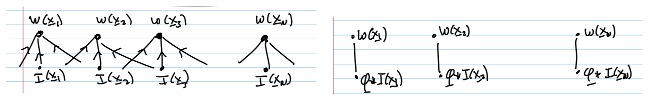

Figure 2: Graphical Models for edge detection.

It corresponds to a *discriminative* model $P(W|I)$ where $W = \{W(\vec{x}) : \vec{x} \in D\}$ is a set of binary-valued variables – $W(\vec{x}) = 1$ if $(\vec{x})$ is an edge, and $W(\vec{x}) = 0$ otherwise. The conditional distribution $P(W|I)$ is factorizable and can be *represented by the graphical structure* is illustrated in figure (2):

[1]A powerful and handy tool including SIFT can be found here: `http://www.vlfeat.org/overview/sift.html`

[2]`http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf`

$$P(W|I) = \prod_{\vec{x}} P(W(\vec{x})|\vec{\phi} * I(\vec{x})), \tag{10}$$

where $P(W(\vec{x}) = 1|\vec{\phi}*I(\vec{x})) = \frac{P(\vec{\phi}*I(\vec{x})|W(\vec{x})=1)P(W(\vec{x})=1)}{P(\vec{\phi}*I(\vec{x}))}$, with $P(\vec{\phi}*I(\vec{x})|W(\vec{x}) = 1) = P(\vec{\phi} * I(\vec{x})|\vec{x}$ on edge) and $P(\vec{\phi} * I(\vec{x})|W(\vec{x}) = 0) = P(\vec{\phi} * I(\vec{x})|\vec{x}$ off edge). (Here $\phi(.)$ are the filters). The prior $P(W(\vec{x}) = 1$ takes into account that far fewer image pixels are likely to be edges (roughly five to ten percent of image pixels are edges in typical images). (A loss function is also used so that false negatives – labeling edges as non0edges – cost more than false positives).

The factorizable form of $P(W|I)$ means that the probability of a pixel being an edge is independent of the probability that other pixels are edges, including its nearest neighbors, from being edges. (Note: this is independent conditioned on the values of $\vec{\phi} * I(.,.)$). Factorization is a big approximation and, as we will show later in the course, non-factorized models perform better (see also the discussion in the next section). But a big advantage of factorization is that the inference algorithm (and the learning) is very simple.

The *inference* algorithm to estimate $W^* = \arg\max_W P(W|I)$ is simple because of the factorizability of $P(W|I)$. At each image pixel $\vec{x}$ – label the pixel as edge if $P(W(\vec{x}) = 1|\vec{\phi}*I(\vec{x})) > P(W(\vec{x}) = 0|\vec{\phi}*I(\vec{x}))$ and as non-edge otherwise (we can modify this using a loss function). This takes linear time in the size of the image.

The *learning* is also simple because of factorizability. The training set consists of two sets of image pixels $i \in X$ with edge labels $W(i) \in \{0, 1\}$ and with filter values $\vec{\phi}*I(i)$. We subdivide this into two sets $X_1 = \{i : \text{ s.t.} W(i) = 1\}$ and $X_0 = \{i : \text{ s.t.} W(i) = 0\}$. We learn $P(\vec{\phi}*I(i)|W(i) = 1)$ and $P(\vec{\phi}*I(i)|W(i) = 0)$ using X_1 and X_0 respectively. We also learn the prior $P(W(i))$ from the relative sizes of X_1 and X_0.

This is a discriminative model since it directly learns the posterior distribution $P(W|I)$ and has no explicit model $P(I|W)$ for generating the image. But it has a prior distribution $P(W) = \prod_{\vec{x}} P(W(\vec{x}))$. Observe that it does have a generative model for generating the features:

$$P(\vec{\phi} * I|W) = \prod_{\vec{x}} P(\vec{\phi} * I(\vec{x})|W(\vec{x})). \tag{11}$$

So we can use $P(W)$ and $P(\vec{\phi}*I|W)$ to generate an image of features $\vec{\phi}*I = \{\vec{\phi} * I(\vec{x})\}$. But there are two problems with this: (i) there is no guarantee that the generated feature image $\vec{\phi} * I$ are consistent with an underlying real image I, (ii) even if it is consistent, it will not correspond to the correct distribution on the image. We will return and clarify these problems later (give example).

5 Labelling Image Regions

We can apply the same approach to the task of labeling image pixels in terms of region class– e.g., sky, road, vegetation, building, edge, other. To do this we define $W = \{W(\vec{x})\}$ so that $W(\vec{x}) \in L$, where $L = \{sky, road, vegetation, other\}$ is a set of labels. We use similar filters $\vec{\phi}(.)$ as before to learn distributions $P(\vec{\phi} * I(\vec{x})|W(\vec{x}))$ for $W(\vec{x}) \in L$.

This results in a factorized model of similar form to equation (10) and with similar graph structure:

$$P(W|I) = \prod_{\vec{x}} P(W(\vec{x})|\vec{\phi} * I(\vec{x})), \tag{12}$$

The learning and inference are simple generalizations of those for the binary task of edge detection. The only difference is the large number of labels.

This simple model does surprisingly well for several classes, see Konishi and Yuille handout. If the filters use color or texture or a combination – then they can achieve very high accuracy on labeling classes such as sky, road and vegetation. This success happens because these classes have low *within-class* variability – all parts of the sky are fairly similar to each other. But it is less successful for detecting classes like buildings which have large within-class variation (e.g., filter responses will appear different if evaluated on the door, the window, the roof, the facade, and even in different parts of the door).

How to deal with large within-class variation? There are three strategies which can be used separately or in combination. The first is to try to find features which are invariant to the within-class variations. This is similar to the invariant features described earlier this lecture. The second is to exploit spatial context – e.g., neighboring pixels are likely to have the same labels. The third is to explicitly represent the different types of building, or even different types of sky (blue, grey, or white).

Currently most computer vision research uses a combination of the first two strategies. The problem is that this approach tends to lead to "black-box" models which may perform well on some datasets but which are difficult to understand and which may not adapt well to other datasets. Moreover, they need are only able to perform certain tasks – e.g., detecting buildings – but need other algorithms to detect the doors, roofs, and windows. Building models that explicitly represent the variations is harder to do but will lead to more effective and flexible algorithms.