Bayesian Edge Detection

- We now describe a Bayesian approach to edge detection (which can be extended to simple semantic segmentation).
- Edge detection is a fundamental computer vision task which may seem easy but is deceptively difficult. Its goal is to find the boundaries of objects and also significant texture changes (like letters on a shirt).
- The first strategy was to assume that edges occur at large intensity discontinuities which could be found by differentiating the image and thresholding the response. This strategy works on simple images containing little texture (e.g., the Canny edge detector) but works badly if images contain texture (e.g., images of the countryside) because there will be large discontinuities which are not at boundaries of objects.
- BDT formulates edge detection as a classification problem to distinguish between real edges (boundaries and significant texture changes) and the background (everything else).

Bayesian Edge Detection

- It is natural to formulate edge detection using filterbanks. BDT requires specifying the probability distribution of the filterbank response *on-edge* and *off-edges* (i.e., in the background). Suitable filterbanks consist of diffferential filters. Examples are families of Gabor functions, or derivative operators at different scales (convolving the image first with a Gaussian and then differentiating).
- Edges are only a small fraction (5-10 percent) of pixels in a typical image. So that statistics of filter responses off-edges is very similar to the statistics of the filter responses over the entire image. It is known (Simoncelli & Olshausen, 2001, M. Green 2020) that these filters responses are very similar for different images. For simple derivatives, the responses are strongly peaked at 0 and fall off rapidly like a Laplace distribution (i.e. the vast majority of pixels have very small image derivatives).
- This is a *naturalness prior* which distinguish between natural images and white noise (e.g., the image on an old television screen). Pop-art and line drawings also have derivative statistics that differ greatly from natural images.

Edge detectors/ texture detectors and decisions

- Now consider the tasks of deciding whether an *image patch* at position x contains an *edge* by which we mean the boundary of an object or a strong texture boundary (e.g., the writing on a t-shirt). The previous section showed that some Gabor filters are tuned (i.e., respond strongly) to edges at specific orientations. But such filters will also respond to other stimuli, such as texture patterns, so how can we decide if their response is due to an edge?
- The simplest way is to *threshold* the response so that an edge, at a specific orientation, is signalled if the filter response is larger than a certain threshold value. But what should that threshold be? How do we do a trade-off to balance *false negative* errors, when we fail to detect a true edge in the image, with *false positive* errors when we incorrectly label a pixel as an edge?
- Also each filter in a filterbank contains some evidence about the presence of an edge, so we can combine that evidence in an optimal manner using BDT.

Filters

- To start with, we consider the evidence for the presence of an edge using a single filter $f(.)$ only. We assume we have an annotated data set so that at each pixel, we have intensity $I(x)$ and a variable $y(x) \in \{\pm 1\}$ (where $y = 1$ indicates an edge, and $y = -1$ does the opposite).
- We apply the filter to the image to get a set of filter responses $f(I(x))$. If the filter is tuned to edges, then the response $f(I(x))$ is likely to be higher if an edge is present than if not. This requires selecting a filter $f(x)$, such as the modulus of the gradient of intensity $|\vec{\nabla} I(x)| = \sqrt{\frac{dI}{dx}^2 + \frac{dI}{dy}^2}$ (since $|\vec{\nabla} I(x)|$ is likely to be large on edges and small off edges).

Conditional probability distributions

- To quantify this, we use the annotated (benchmarked) data set to learn *conditional probability distributions* for the filter response $f(I)$ conditioned on whether there is an edge or not:

$$P(f(I)|y=1), P(f(I)|y=-1).$$

- Each distribution is estimated by computing the *histogram* of the filter response by counting the number of times the response occurs within one of N equally spaced bins and normalizing by dividing by the total number of responses. The histograms for $P(f(I)|y=1)$ and $P(f(I)|y=-1)$ are computed from the filter responses on the points labeled as edges $\{f(I(x)) : y(x)=1\}$ and not-edges $\{f(I(x)) : y(x)=1\}$ respectively. Typical conditional distributions are shown in the figure on the next slide. Since most pixels in images are not edges (typically $90-95\%$) the distributions $P(f(I)|y=-1)$ are similar to those for the entire image (e.g., tend to be Laplacian distributions).
- The histogram is a non-parametric way to represent a probability distribution. This means that, unlike parametric models, t makes no assumptions about the form of the distribution. Its disadvantage is that it needs much more data to train than parametric models (particularly when learning distributions of many filters).

Figure for conditional distributions

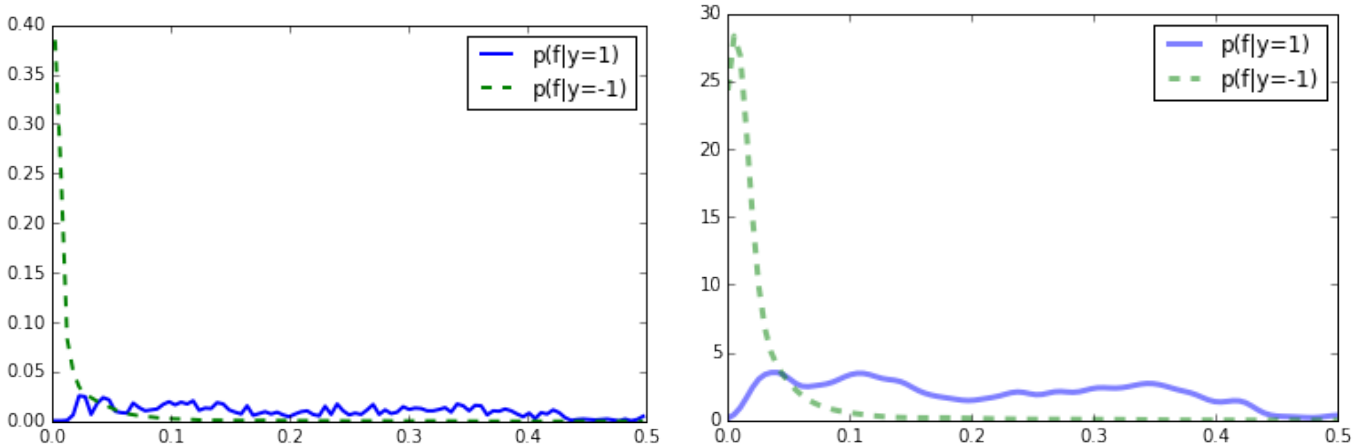

Figure 26: The probability of filter responses conditioned on whether the filter is *on* or *off* an edge – $P(f|y=1), P(f|y=-1)$, where $f(x) = |\vec{\nabla} I(x)|$. Left: The probability distributions learned from a data set of images. Right: The smoothed distributions after fitting the data to a parametric model.

Statistical edge detection

- We can now perform edge detection on an image. At each pixel x we compute $f(I(x))$ and calculate the conditional distributions $P(f(I(x))|y = 1)$ and $P(f(I(x))|y = -1)$. These distributions give local evidence for the presence of edges at each pixel.
- Note, however, that local evidence for edges is often highly ambiguous and it can be difficult for humans to decide if there is an edge in a small image region. Spatial context can supply additional information to help improve edge detection, and so can high-level knowledge (e.g., by recognizing the objects in the image).

Log-likelihood ratio

- The log-likelihood ratio $\log \frac{P(f(I(x))|y=1)}{P(f(I(x))|y=-1)}$ gives evidence for the presence of an edge in image I at position x. This ratio takes large positive values if $P(f(I(x))|y=1) > P(f(I(x))|y=-1)$ (i.e., if the probability of the filter response is higher given an edge is present) and large negative values if $P(f(I(x))|y=-1) > P(f(I(x))|y=1)$.
- A natural decision criterion is to decide that an edge is present if the log-likelihood ratio is greater than zero and that otherwise there is no edge. This can be formulated as a *decision rule* $\alpha(x)$:

$$\alpha(x) = 1, \text{ if } \log \frac{P(f(I(x))|y=1)}{P(f(I(x))|y=-1)} > 0, \quad \alpha(x) = -1, \text{ if } \log \frac{P(f(I(x))|y=1)}{P(f(I(x))|y=-1)} <$$

  This can expressed, more compactly, as

$$\alpha(x) = \arg \max_{y \in \{\pm 1\}} y \log \frac{P(f(I(x))|y=1)}{P(f(I(x))|y=-1)}.$$

  .
- But this is no a good choice for edge detection. From the perspective of BDT this corresponds to assuming a uniform prior of the edges (pixels are equally likely to be edge or not edge) and to assuming that all types of error are equal.

Statistical edge detection figure

Figure 27: The input image and its groundtruth edges (far left and left). The derivative dI/dx of the image in the x direction (center). The probabilities of the local filter responses $P(\vec{f}(I(x))|y = 1)$ (right) and $P(\vec{f}(I(x))|y = -1)$ (far right) have their biggest responses on the boundaries and off the boundaries, respectively, hence the log-likelihood ratio $\log \frac{P(\vec{f}(I(x))|y=1)}{P(\vec{f}(I(x))|y=-1)}$ gives evidence for the presence of edges.

Ambiguities in edge detection

- Note that this rule gives perfect results (i.e., is 100% correct) if the two distributions do not overlap, i.e., if $P(f(I(x))|y=1)P(f(I(x))|y=-1)=0$ for all I. In this case it is impossible to confuse the filter responses to the different types of stimuli. But this situation never happens for ambiguous tasks like edge detection. Now consider a more general *log-likelihood ratio test* that depends on a threshold T; this gives a rule:

$$\alpha_T(x) = \arg \max_{y \in \{\pm 1\}} y\{\log \frac{P(f(I(x))|y=1)}{P(f(I(x))|y=-1)} - T\}.$$

- By varying T we get different types of mistakes. We can distinguish between the *false positives*, which are non-edge stimuli that the decision rule mistakenly decides are edges, and *false negatives*, which are edge stimuli that are mistakenly classified as not being edges. Increasing the threshold T reduces the number of false positives but at the cost of increasing the number of false negatives, while decreasing T has the opposite effect.

Ambiguity of edges figure

Figure 28: The local ambiguity of edges. An observer has no difficulty in detecting all of the boundary of the horse if the full image is available (left). But it is much more difficult to detect edges locally (other panels).

Decision theory and trade-offs

- Making a decision requires a trade-off between these two types of errors. Bayes decision theory says this trade-off should depend on two factors.
- First, the *prior* probability that the image patch is an edge. Statistically most image patches do not contain edges, so we would get a small number of total errors (false positives and false negatives) by simply deciding that every image patch is non-edge. This would encourage us to increase the threshold T (to $-\infty$ so that every image patch would be classified as non-edge).
- Second, we need to consider the *loss* if we make a mistake. If our goal is to detect edges, then we may be willing to tolerate many false positives provided we keep the number of false negatives small. This means we choose a decision rule, by reducing the threshold T, so that we detect all the real edges but also output "false edges," which we hope to remove later by using contextual cues. Later we show how this approach can be justified using the framework of decision theory.

Combining multiple cues for edge detection

- Now we consider combining several different filters $\{f_i(.)|i = 1, ..., M\}$ to detect an edge by estimating the *joint* response of all the filters $P(f_1, f_2, \ldots|y) = P(\{f_i(I(x))\}|y)$ *conditioned* on whether the image patch I at x is an edge $y = 1$ or not an edge $y = -1$. This leads to a decision rule:

$$\alpha_T(I(x)) = \arg \max_{y \in \{\pm 1\}} y\{\log \frac{P(\{f_i(I(x))\}|y = 1)}{P(\{f_i(I(x))\}|y = -1)} - T\}.$$

- This approach has two drawbacks which are related. First, the joint distributions require a large amount of data to learn, particularly if we represent the distributions by histograms. Second, the joint distributions are "black boxes" and give no insight into how the decision is made. So it is better to try to get a deeper understanding of how the different filters contribute to making this decision by studying whether they are *statistically independent*.

Combining multiple cues for edge detection: Konishi et al. 2003

- This approach was developed by Konishi et al. Statistical Edge Detection: Learning and Evaluating Edge Cues. TPAMI 2003. It was applied to two datatsets with annotated edges, Sowerby (English countryside) and South Florida (indoor images), which had just been released. This was the first attempt to perform edge detection using annotated datasets and probabilities.
- On the South Florida dataset this learning-based edge detector was only slightly better than conventional (not learning based) methods. The reason was that the background, the non-edges, were very simple in this datasets and contained no texture (all texture had been masked out). This meant that the probability distributions $P(f(I)|y = 10$ and $P(f(I) = -1)$ were very different if $f(I)$ was a derivative filter. So thresholding a single filter, like a conventional edge detector, was good enough.
- But on Sowerby where the background contained a lot of texture (gress, tress, etc.) there were many small edges which a derivative filter would correspond to, So conventional edge detectors worked poorly. The statistical/probabilistic approach by Konishi et al. used probabilities as s principled way to combine differential filters at different scales. Thereby giving much better results.

Combining cues with statistical independence – Out of Scope

- The response of the filters is statistically independent if:

$$P(\{f_i(I(x))\}|y) = \prod_i P(f_i(I(x))|y) \text{ for each y}$$

- This implies that the distributions $P(f_i(I(x))|y)$ can be learned separately (which decreases the amount of data) and also implies that the log-likelihood test can be expressed in the following form:

$$\alpha_T(x) = \arg \max_{y \in \{\pm 1\}} y\{\sum_i \log \frac{P(f_i(I(x))|y=1)}{P(f_i(I(x))|y=-1)} - T\}$$

- Hence the decision rule corresponds to summing the evidence (the log-likelihood ratio) for all the filters to determine whether the sum is above or below the threshold T. This means that each filter gives a "vote," which can be positive or negative, and the decision is based on the sum of these votes. This process is very simple, so it is easy to see which filters are responsible for the decision.

Combining cues with conditional independence – Out of Scope

- But very few filters are statistically independent. For example, the response of each filter will depend on the total brightness of the image patch, so all of them will respond more to a "strong" edge than to a "weak" edge. This suggests a weaker independence condition known as *conditional independence*. Suppose we add an additional filter $f_0(I(x))$ that, for example, measures the overall brightness. Then it is possible that the other filters are statistically independent conditioned on the value of $f_0(I(x))$:

$$P(\{f_i(I(x))\}, f_0(I(x))|y) = P(f_0(I(x))|y) \prod_i P(f_i(I(x))|f_0(I(x)), y)$$

- This requires only representing (learning) the distributions $P(f_i(I(x))|f_0(I(x)), y)$ and $P(f_0(I(x))|y)$.

Combining cues with conditional independence – Out of Scope

- It also leads to a simple decision rule:

$$\alpha_T(x) = \arg \max_{y \in \{\pm 1\}} y\{\log \frac{P(f_0(I(x))|y=1)}{P(f_0(I(x))|y=-1)} + \sum_i \log \frac{P(f_i(I(x))|f_0(I(x)), y=1)}{P(f_i(I(x))|f_0(I(x)), y=-1)} - T\} \quad (19)$$

- It has been argued (Ramachandra & Mel, 2013) that methods of this type can be implemented by neurons and may be responsible for edge detection. Note that the arguments here are general and do not depend on the type of filters $f_i(.)$ or whether they are linear or nonlinear. It has, for example, been suggested that edge detection is performed using the energy model of complex cells (Morrone & Burr, 1988).

Classification for semantic segmentation

- The same approach can be applied to other visual tasks like semantic segmentation. For example, consider using local filter responses to classify whether the local image patch at x is "sky," "vegetation," "water," "road," or "other"). We denote these by a variable $y \in \mathcal{Y}$ (e.g., where $\mathcal{Y} = \{"sky", "vegetation", "water", "road", or "other"\}$. We choose a set of filters $\{f_i(I(x))\}$ that are sensitive to texture and color properties of image patches. Then, as before, we learn distributions $P(\{f_i(I(x))\}|y)$ for $y \in \mathcal{Y}$. We select a decision rule of form:

$$\alpha(I(x)) = \arg\max_{y \in \mathcal{Y}} P(\{f_i(I(x))\}|y) T_y,$$

  where T_y is a set of thresholds (which can be derived from decision theory).
- Experiments on images show that this method can locally estimate the local image class with reasonable error rates for these types of classes (Konishi & Yuille, 2000) and computer vision researchers have improved these kinds of results using more sophisticated filters.

Classifying other image classes

Figure 29: Classifying local image patches. The images show the groundtruth (Mottaghi et al., 2014). Certain classes – sky, grass, water – can be classified approximately from small image patches.

Bayes versus Regression (1)

- Konishi et al. formulated edge detection as Bayesian inference by learning conditional generative distributions $P(f(I)|y=1)$, $P(f(I)|y=-1)$, and a prior $P(y)$. The regression approach is to learn a distribution $P(y|f(I))$ directly. Deep Networks are the most popular regression based approaches and are very effective. What are the advantages of the Bayesian approach compared to regression?
- The regression approach is currently state of the art on annotated (benchmarked) datasets (S. Xie and Z. Tu 2015). The feature vectors are hierarchical parameterized functions (e.g., $f(I) = f(I;\omega)$, where ω are the weights of the network). A loss function is specified (also with a bias towards detecting edges) and it can be chosen to give rewards at different levels of the network (reflecting the fact that cues for images occur at multiple scales in the image). By contrast, the Bayesian approach used a limited hand-specified set of filters so that the probability distribution of them could be learnt (but could be extended).

Bayes versus Regression (2)

- *But the Bayesian approach has one big advantage*. For edge detection it can do domain transfer between two types of datasets. Domain transfer is a problem for vision algorithms because the statistical properties of images can differ greatly between different domains. Hence algorithms which perform a visual task on one domain may fail completely on another domain, because the algorithms have been trained on images in the first domain, so they may not work well on images in the second domain which are very different from those in the first.
- For example, the Sowerby edge detection dataset consists of image of the English countryside. These contain sharp edges like the boundaries of roads, buildings, and houses. But there is a lot of texture, e.g., vegetation, in the non-edge regions. By contrast the South Florida dataset consisted of indoor images where there are sharp edges but no texture in the background. Hence an edge detector trained on the South Florida dataset will perform badly on Sowerby because it will tend to find edges in the texture (i.e. in the vegetation).

Bayes versus Regression (3)

- Regression methods find it hard to transfer between the two datasets, unless they have training data for both. But it fairly easy to transfer edge detectors between Sowerby and South Florida using Bayesian methods.
- This is based on two observations: (I) the probability distributions $P(f(I)|y = 1)$ will be very similar between the two datasets (because the edges have similar properties in each) and hence only need to be learnt in one dataset. (II) Most of the pixels in images are not edges, so it is possible to learn $P(f(I)|y = -1)$ approximately by using all the pixels in the images and hence $P(f(I)|y = -1)$ can be learnt without needing annotation.
- Hence we can define edge detectors using the log-likelihood ratio test for both datasets, but only using the annotations on one dataset. (See Konishi et al. 2003 for more details).