Statistics for Image Modeling
CMSC 426

Motivation

To do vision, we need to model the world. For example, in edge detection, we used a qualitative model based on generalities about geometry and objects, which told us that the image usually changes rapidly at the boundary of objects. We will use mathematical certainties about 3D geometry a lot in the weeks to come. But often our knowledge of the world is more statistical in nature.

Some examples:

1. To detect contours, it will help to have some statistical model of contours that says that object boundaries are more likely to have certain shapes, though any shape is possible. This might tell us, for example, that two boundary fragments are more likely to be connected by a smooth, simple shape than by a long, convoluted path. To model this, we would need a probability distribution over all shapes, that tells us how likely any shape is to be a boundary.
2. Face detection. There is great variety in how a face can look, but some appearances are much more likely to be faces than are others. When we want to model the diversity of the real world, a probability distribution is useful.
3. Background subtraction. This will be a case we look at in more detail. The basic idea of background subtraction is that we want to separate the background in front of the camera, which is more or less the same over time, from foreground objects that may briefly enter and leave the scene. To model the background, we will take video of the scene with just the background present. Then we will use this background model to understand a new image, and to classify every pixel in the image as either background or foreground.

   For example, suppose we set up a camera in front of your house, and want to detect when someone comes by. One way to do this is to look for changes in the image. When is it not just an image of your front steps? One way to do this is to take a picture of the steps when no one is there, and compare this to a new image. When they are different, something has changed. For example, there may be a pixel that is dark, because the steps are dark. If this pixel becomes bright, maybe a person is there. This doesn't work, however. The brightness of pixels varies depending on lighting and weather, even when no one is there. So the problem is how do we tell the difference between the normal variation in the intensity of a background pixel, and variation that is due to the presence of a foreground object, like a person? We can use statistics to model the background variation, so we can tell it from the foreground. So, for example, we may find that when there is no one present, one pixel can vary tremendously in intensity due to lighting changes, while a second pixel is almost always dark. Then, if the first pixel is bright in an image, this doesn't provide much evidence that a person is present, but if the second pixel is bright, it's a more useful sign that someone is out there.

Probability Distribution

Let's just recall what a probability distribution would be like for a single pixel in an image. The pixel will have a value from 0 to 255. Each of these values is assigned a probability between 0 and 1, and the sum of the probabilities for all these possible values should be 1. Let's write *p(I(x,y)=k)* for the probability that the pixel at *(x,y)* will have an intensity of *k*, and we can say:

$$\sum_{k=0}^{255} p(I(x,y)=k)=1$$

To do background subtraction, we really want to talk about the probability distribution of a pixel's intensity, given that it is background. We can denote the state that a pixel is background as *B(x,y)*, and then *p(I(x,y)=k|B(x,y))* denotes the probability that pixel *(x,y)* has intensity *k*, given that the pixel is background. Similarly, we can use *F(x,y)* to denote that pixel *(x,y)* is foreground. Then either *F(x,y)* or *B(x,y)* must be the case.

It will also be important to recall Bayes' law. Suppose we have two events, *C* and *D*, and *p(C|D)* denotes the probability of event *C* occurring, given that event *D* has occurred. Then Bayes' law states:

$$P(C \mid D)=\frac{P(D \mid C)P(C)}{P(D)}$$

In the case of background subtraction, we can use Bayes' law to determine:

$$P(B(x,y) \mid I(x,y)=k)=\frac{P(I(x,y)=k \mid B(x,y))P(B(x,y))}{P(I(x,y)=k)}$$
$$=\frac{P(I(x,y)=k \mid B(x,y))P(B(x,y))}{P(I(x,y)=k \mid B(x,y))P(B(x,y))+P(I(x,y)=k \mid F(x,y))P(F(x,y))}$$

The left hand side of the equation is the background subtraction problem we want to solve. Given the intensity of pixel *(x,y)*, what is the probability that it is background? The right hand side tells us how to determine this. It says that we need to know *p(I(x,y)=k|B(x,y))*, which is the probability distribution for a background pixel. And we need to know *P(B(x,y))*, which is called our *prior* on the probability that a pixel is background. For example, experience might tell us that in only one image in 1,000 has a person come to our door and created a foreground pixel. The denominator, as shown on the far right, can be broken into the sum of two parts, a probability distribution for the background and for the foreground. So the main problem of background subtraction (in this formulation) is to find a way to compute a probability distribution for the foreground and background. In our simple implementation, we ignore the priors and the probability distribution for foreground pixels, because we don't have much knowledge of these, and instead just suppose that:

$$P(B(x,y) \mid I(x,y) = k) \propto P(I(x,y) = k \mid B(x,y))$$

Then we use video of the background to estimate the right hand side of the equation, and heuristically use some threshold to identify background and foreground pixels.

Sample Distribution

The most straightforward way to model a probability distribution is by collecting samples. For example, suppose we observe the pixel *I(x,y)* 1,000,000 times when we know that the pixel is due to background, and we find that it has an intensity of 117 10,000 times. Then it seems reasonable to suppose that *P(I(x,y)=117|B(x,y))= 10,000/1,000,000.* If we estimate every probability in the background distribution like this, we call the resulting distribution the *Sample Distribution*. That is, we are assuming that the true probability of an event is the fraction of times we have observed it.

Using the sample distribution means making the implicit assumption that the probability distribution that's appropriate for a newly observed pixel is exactly the same as the probability distribution that produced all our previous observations. We say that we are assuming that the distribution is *ergodic*, which means that it doesn't change over time. Like all assumptions, it isn't exactly true. For example, the distribution of the intensity of a background pixel will likely change over time, as the sun rises, or the rain stops. In practice, though, we will try to collect samples in which ergodicity is approximately true.

This already gives us a method for performing background subtraction, using Bayes law and sample distributions to represent what the foreground and background intensity distributions are like.

Kernel Density Estimation

The problem with this approach is that we usually do not have enough information to get a really accurate estimate of the background distribution. We can smooth out the noisy sample distribution using kernel density estimation. In this case, we estimate:

$$P(I(x,y) = k) = \sum_{i=1}^{N} \frac{1}{N\sigma\sqrt{2\pi}} \exp\left(-(k - s_i)^2 \Big/ 2\sigma^2 \right)$$

Here there are N samples, and we are writing s_i to denote the intensity of sample i. One way to think about this smoothing is to imagine what happens if we have just one sample, say, we observe a pixel with an intensity of 100. Using a sample distribution, we assume that the probability that the intensity equals 100 in the future will be 1, and all other intensities have a probability of 0. When we have many samples, we use a sample distribution that is the average of the sample distribution for one sample. However, this really doesn't seem like the best assumption. Instead, with Kernel Density Estimation, when we observe the intensity 100, we assume the distribution of future intensities is a Gaussian centered at 100. So 100 is the intensity that we most expect, but we also assign

non-zero probabilities to similar intensities. We assume 99 is also fairly likely, and 98 a bit less likely, etc…. How wide the Gaussian is depends on the parameter sigma, which we will probably choose heuristically. Then when we have many samples, we average all these Gaussians together.

Markov Models

Everything we've done so far just models the distribution of the image in a single pixel. We can use these distributions on a whole image, if we assume that every pixel is independent. Then, for example, to classify a pixel as foreground or background, we only need to use information that has to do with that pixel. Independence means that what's happening with other pixels doesn't matter.

These independence assumptions are usually pretty unrealistic. So now we introduce a statistical model that does not make this assumption, called a Markov model. We will describe Markov models using a 1D image. The basic idea is much the same for a 2D image (although it turns out that in many cases, algorithmic issues that arise in dealing with 2D Markov models are very different than those for 1D Markov models). We will consider these issues in trying to characterize textures.

If we assume that every pixel has an identical, independent distribution, we can generate some simple textures. For example, we could assume that every pixel has an equal chance of being 0 or 1, and if we randomly generate images from that distribution we would get something like:

```
>> rand(5,12) > .5
ans =
   1   1   1   1   1   0   0   1   1   1   0   0
   0   0   0   0   1   1   1   1   0   1   1   1
   1   0   1   1   0   1   1   0   0   0   1   1
   1   0   1   1   0   0   1   1   1   1   0   1
   0   1   0   0   0   0   1   0   1   1   1   0
```

Each row is a different sample of this texture. However, textures like this aren't very interesting. Suppose instead we want to describe a texture of alternating 0s and 1s, which can look like:

01010101010101010101010101010

or like:

10101010101010101010101010101

Every pixel has an equal chance of being a 0 or 1, the same as in the previous distribution. But the key thing about this new texture is that every pixel's probability distribution depends on its neighbor. So if the previous pixel is 0, then the next pixel

must be a 1. This texture is pretty simple; there are really only two instances of it. To make it a little more statistical, we could describe a texture in which there each pixel has an 80% of being the same as the previous pixel, and a 20% chance of being different. Some instances of this texture look like the rows below:

0	1	1	1	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This creates patterns in which there are strings of consecutive 0s and 1s, but the lengths of these strings is stil random. When the probability distribution of a pixel depends on its neighbors, we call this a Markov model. The size of the neighborhood can vary. The bigger the neighborhood, the more complex the textures we can generate. For example, suppose we have a texture which consists of stripes five pixels wide, where each stripe has an arbitrary intensity that is different than its neighbors. For example:

0 0 0 0 0 2 2 2 2 2 5 5 5 5 5 1 1 1 1 1 5 5 5 5 5

We can generate textures like this if each pixel looks at its last five neighbors. If they are not all the same, then there is probability 1 that this pixel will have the same intensity as its immediate neighbor. If the last five pixels are all the same, then this pixel has a randomly chosen intensity that is different from its immediate neighbor.