CS 280, Computer Vision (Prof. Jitendra Malik) May 5, 1999
Lecture #28 (Object Reconition II and Class Summary) **DRAFT** Notes by Amar Chaudhary

---

- Object Recognition, Part II
- In Conclusion...

1 Objection Recognition, Part II

In the previous lecture, we discussed two algorithms for object recognition for specific classes of objects and source images. **Alignment** was particularly suited for line drawings of geometric objects with corners, while **Blobword** used color and texture to locate objects in photos. As presented in the previous lecture, this techique has been useful for recognition of animals in photos, as well as more generalized image database querying.
We will now discuss a more general approach to object recognition.

1.1 Object Recognition as Classification of Feature Vectors

Much effort in the artificial intelligence community has been devoted to the problems of recognition and classification. Most techniques for solving these problem rely on a training phase, in which examples of the target classes are presented to the learning algorithm. The algorithm is then presented a separate set of test items, for which it tries to estimate the class of each item. The error rate of the classification is measured, and the algorithm is refined before another round of training and testing ensues.
We can leverage these ideas for visual recogition and classification. Consider the classes of objects in the following figure:

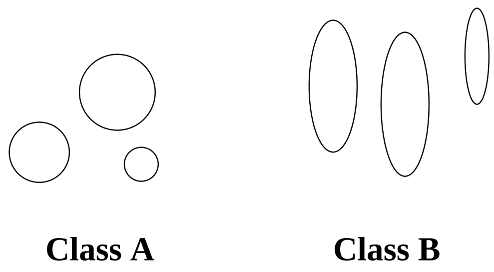

Figure 1: Classes of objects in 2D

We clearly see a difference between the two classes of objects. This difference is not measured by lines, vertices, color or texture, but rather by another *feature* that is significant in these particular objects, namely the aspect ratio. We can describe each object to be recognized by a collection of such features, where each feature is a number (such as 1 for circles and greater than 1 for the y/x aspect ratio of ellipses). This collection of features is

called a *feature vector*. We have now reduced the large number of pixels in each object to a small vector of numbers, a significant data reduction. Moreover, since a feature vector is a vector in the mathematical sense of the word, we can take its norm and measure its distance from other vectors using an appropriate metric.
We now present the classification problem using feature vectors. However, it is important to realize that any of the techniques presented are only as good as features used to describe the objects!

Pattern classification problem

Given a feature vector $\vec{v} \in \mathbf{R}^n$ associated with labelled examples of classes C_1 and C_2, develop a dicision rule that can classify some new new unlabelled instance $\vec{v}'$.

Example. *Suppose we have the following two-dimensional feature vectors for examples of tigers and zebras:*

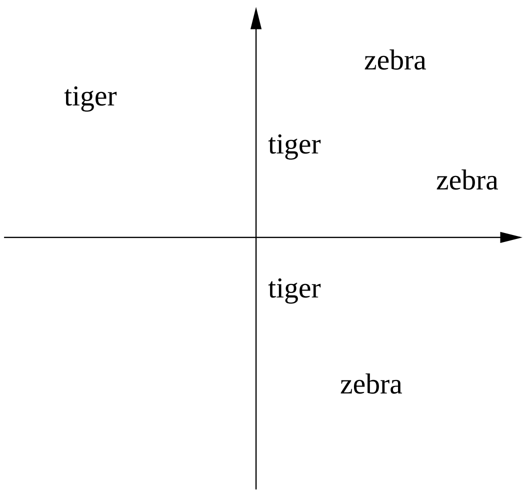

Figure 2: Tigers and zebras in feature-vector space.

Some common classifiers include:

Nearest Neighbor. For a vector $\vec{v}'$ and labelled vectors $\vec{v}_i \in \mathbf{R}^n$, choose the $vecv_i$ that minimizes the distance $\|\vec{v}' - \vec{v}_i\|$. This is a good technique by which to measure the success of a more refined, but perhaps less robust, classifier.

K Nearest Neigbors. Find the k nearest neighbors to the unlabelled vector.

Best Separating Hyperplane. Find the hyperplane that either separates the classes, or at least minimizes the number of vectors of different classes on either side of the hyperplane. This tecnique requires less memory than nearest neighbors and runs faster when labelling new points. However, it will do poorly if the classes are not separated by a hyperplane.

Neural Networks. Essentially, neural networks are classification by multiple hyperplanes. Neural nets have been quite successful in digit recognition.

Faced with the choise between these and many other potential classifiers, what is the correct thing to do? Ultimately, we want use whichever technique has the smallest error rate.

For classes C_1 and C_2 and a given feature $x \in R$, should it be labeled as C_1 or C_2? Theoritically, the solution is provided using Bayesian classification and choosing the class with the higher probability given x:

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x)} \tag{1}$$

$$P(C_2|x) = \frac{P(x|C_2)P(C_2)}{P(x)} \tag{2}$$

The probabilities on the right-hand side can be learned from example data (i.e., the training set). However, to infer $P(x|C_1)$ and $P(x|C_2)$, we must assume the form of the probability distributions. For the remainder of this discussion, we will assume that the distributions are Gaussian:

$$P(x|C_1) = \frac{1, \sqrt{2\pi\sigma}}{e}^{-(1/2)(\frac{x-\mu_1}{\sigma_1})^2}$$

where μ_1 and σ_1 are found statistically from the data. A similar formula can be used to info $P(x|C_2)$. We can now manipulate equation **??** as follows:

$$\begin{aligned} P(C_1|x) &= \frac{P(x|C_1)P(C_1)}{P(x)} \\ &= \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} \\ &= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} \end{aligned}$$

Define the bf log-likelihood ratio $z = ln(P(x|C_1)P(C_1) - P(x|C_2)P(C_2))$. Substituting in z into the previous equation, we get:

$$P(C_1|x) = \frac{1}{1 + e^{-z}} \tag{3}$$

which is a sigmoid function, as illustrated in figure **??**:

If both C_1 and C_2 are Gaussian distributions such that $\sigma_1 = \sigma_2$, then we can calculate z as follows:

$$z = (\frac{\mu_1 - \mu_2}{\sigma^2})x + ln\frac{P(C_1)}{P(C_2)} + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2} \tag{4}$$

If $P(C_1) = P(C_2) = 1/2$, then we can label x as C_1 if $P(C_1|x)$ is greater than $1/2$, or C_2 otherwise. In figure **??**, this corresponds to position of $P(C_1|x)$: x should be labeled C_1 if it lies on the left-hand side of the sigmoid curve, C_2 otherwise. Note that this classification rule is similar to the separating hyperplane discussed above.

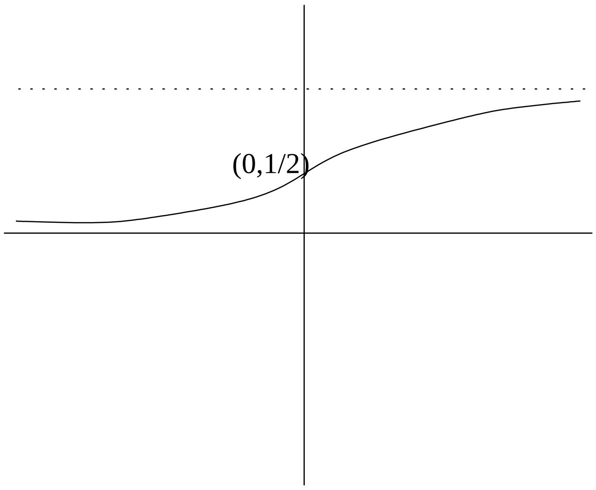

Figure 3: A sigmoid function.

2 Conclusion: The Big Picture

The field of computer vision includes several types of problems, and draws upon a wide range of mathematical and computational tools.
There are four major types of vision problems:

Reconstruction of an image from motion, texture, stereo, etc.

Grouping of pixels into regions and objects. We have seen techniques for detecting edges and texture boundaries, as well as segmentation of images using the normalized cut algorithm.

Recognition is perhaps the most important problem in vision, yet the least understood. This is where most of the current work in computer vision is focused.

Visual Control is really a combination of problems using vision to perform a complex task, like driving a car.

Some of the mathematic tools we explored included:

Geometry of image formation: shape, pose, projection, etc.

Physics of images and light.

Statistics for grouping and recognition.

Control Theory will have to wait until next semester...

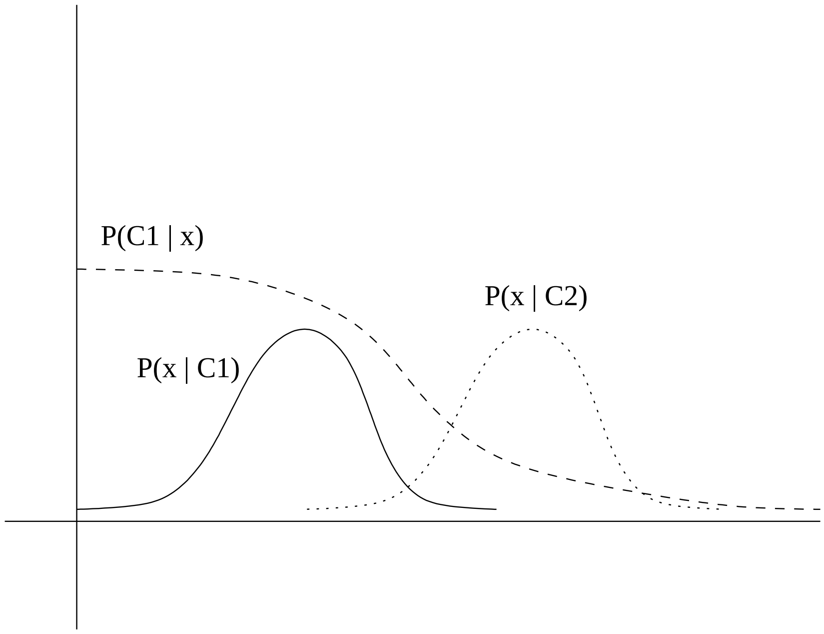

Figure 4: Probabilities with Gaussian distributions.