EECS225d Audio Signal Processing in Humans and Machines **Spring 1995**

Lecture 30: April 5, 1995

Lecturer: Prof. Nelson Morgan *Scribe: Eric Fosler*

30.1 Recap from last time

Last time, Professor Morgan discussed feature extraction for ASR. Many of the same techniques that were used for vocoding are also applicable to ASR, e.g. LPC processing. Typically, features are extracted within successive analysis windows which overlap. For example, a window of 20 msec overlapping every 10 msec is typically used, although frame rates of 5-20 msec (or longer) are often used. The features extracted are generally spectral or cepstral coefficients which condense the information in the speech signal to a vector of numbers, typically of length 5 to 40.

Often robustness can be achieved by looking across time in deriving features. Many techniques in use today emphasize changes in the speech spectrum. Delta techniques estimate a local time derivative, for example. Cepstral Mean Subtraction (CMS) subtracts the mean of the cepstral coefficients in order to eliminate constant variations between speech signals, as in a microphone change. RASTA-PLP provides an online version of this by either high-pass or band-pass filtering the spectral coefficient trajectories to remove constant or slowly changing parts of the times series for each feature.

30.2 Pattern Recognition

In general, the task of a pattern recognizer is to associate each feature vector with a particular label. In handwriting recognition, or in optical character recognition, we can associate feature vectors with characters in the input. In speech recognition, we take the spectral or cepstral features and associate them with linguistic entities, such as phones or words.

Speech pattern recognition is essentially pattern classification with temporal sequences. For now, we are going put aside the issue of temporal sequencing to concentrate on the core issues of supervised pattern recognition. A *supervised* pattern recognition system is trained with labeled examples; that is, each pattern (feature vector) has a class label associated with it. Pattern recognizers can also be trained in an unsupervised fashion. For example, a technique known as vector quantization takes a set of speech feature vectors and clusters them in groups by finding implicit groupings in the data. The resulting structure is known as a codebook, which can be used to index new vectors by finding the closest class to the new vector.

In general, pattern recognition concerns classifying individuals into groups. For example, consider classifying humans with the features of height and weight. In figure 30.1, the open circles may be Swedish basketball players, whereas the closed circles may represent speech recognition researchers. It is very easy to divide the two sets of people into classes ω_1 and ω_2.

In speech, general regions for vowels in terms of the formant frequencies F1 and F2 can also be seen, as in figure 30.2.

The vowels represented are as pronounced in the words *beet* (/i/), *bat* (/ae/), *bot* (/a/), and *boot* (/u/). Notice that they fall into nice groupings. Unfortunately, speech data in general is not this nice. For instance, if more vowels were added to the chart, the classes often intermingle in the F1/F2 space.

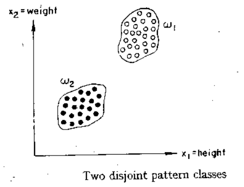

Figure 30.1: Distinct classes of Swedish basketball players and speech recognition researchers

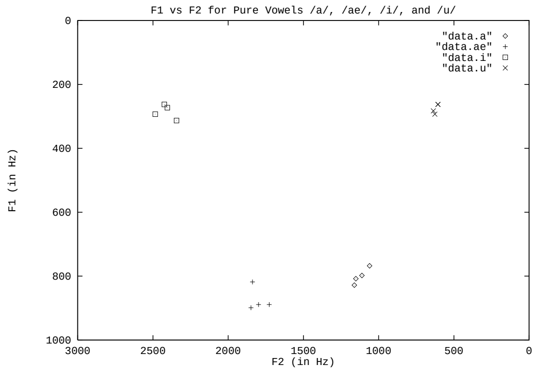

Figure 30.2: F1 and F2 for various vowels

30.3 Dimensionality

One problem in pattern recognition is trying to pick the appropriate feature vector size. The best representations for discrimination on the training set have a large dimensionality. If you have enough dimensions, you can split up a group of patterns in any arbitrary way. However, the best representations for generalization to the test set are usually succinct. There is a trade off between how well you want to classify the training set to how well you want to generalize to the test set.

In general, you want to find a good set of generalizable features. There are several methods of reducing the dimensionality of the features. Two direct analytical approaches are Principal Components analysis, and Linear Discriminant Analysis. The object of Principal Components analysis is to find a projection of the feature vector down to a lower dimension, such that the variance accounted for is maximized. This is the most common approach used, and is reasonable if you don't know anything else about the data. The limitation is that the components are chosen based on most of the variance, which can be a problem if most of the variance isn't important, e.g. the variance around DC within speech signals. For pattern classification, you really want to have components which account for discriminability.

Linear Discriminant Analysis tries to maximize variance between classes, and minimize variance within classes. Duda and Hart[DH73] give a good description of the technique in a chapter of their book. This technique has been rediscovered in speech recognition within the last 4 to 5 years.

One can also use application specific knowledge to reduce the dimensionality of the feature vectors, or just test different reductions by evaluating a pattern recognizer trained on the reduced features on a test set. This is often the best way to get results, although it is more computationally intensive.

30.3.1 Some opinions

Professor Morgan expressed the opinion that it is often better to throw away bad data or features than to reduce its weight. A question was raised about how to determine what data or features are bad. Professor Morgan replied that the best way to make that determination is through analysis of errors in the pattern recognizer. For instance, in ASR, log energy is an unreliable feature, and removing it in his experience improved resultsin realistic conditions.

Another opinion of Professor Morgan's was that dimensionality-reduction based on variance alone is often a bad choice for supervised pattern recognition.

30.4 Pattern Recognition Methods

There are several methods used in pattern recognition; a short list follows.

- Minimum Distance
- Discriminant Functions
  - Linear Discriminant Functions
  - Nonlinear Discriminant Functions
  - Statistical Discriminant Functions

The following discussion will be about deterministic methods only. In the next lecture, statistical methods will be discussed.

30.4.1 Minimum Distance Classifiers

In a minimum distance classifier, we define a distance function which takes the feature vector which represents an new element and a stored vector, and returns the distance between the two vectors. The classifier then chooses the class of the stored element closest to the new input as the class of the input. The choice of the distance function is very important; it is equivalent to implicit statistical assumptions you are making about the data. For speech, this is complicated by the temporal variability within the templates.

A sample minimum distance classifier might have j templates $\mathbf{z}_i$, where $0 \leq i \leq j$. For each input vector $\mathbf{x}$, choose

$$\operatorname*{argmin}_i D_i = \sqrt{(\mathbf{x}-\mathbf{z}_i)^T(\mathbf{x}-\mathbf{z}_i)} \tag{30.1}$$

In other words, choose the i which minimizes the distance D_i. Manipulating the algebra, we get:

$$\operatorname*{argmin}_i D_i^2 = \mathbf{x}^T\mathbf{x} - 2(\mathbf{x}^T\mathbf{z}_i - \frac{1}{2}\mathbf{z}_i^T\mathbf{z}_i) \tag{30.2}$$

Since $\mathbf{x}^T\mathbf{x}$ is constant over i, we can also equivalently choose to maximize the following function:

$$\operatorname*{argmax}_i D_i'^2 = \mathbf{x}^T\mathbf{z}_i - \frac{1}{2}\mathbf{z}_i^T\mathbf{z}_i \tag{30.3}$$

which is a linear function of $\mathbf{x}$. If the templates are all normalized so that $\mathbf{z}_i^T\mathbf{z}_i = 1 \forall i$ then we can just maximize:

$$\operatorname*{argmax}_i D_i''^2 = \mathbf{x}^T\mathbf{z}_i \tag{30.4}$$

The min distance classifier in this case is equivalent to a classifier based on the maximum correlation with the stored vector.

Now that we have a distance function, we can derive a decision rule for classifying new vectors by assigning the vector the class of its nearest neighbor. In the limit as the number of samples goes to infinity, this rule will provably get at most twice the error of the optimum classifier[DH73]. We can also generalize this to a k-nearest neighbors rule, where the class of the new vector is decided by the k nearest neighbors; in a variant of this, the voting weight is decided by the distance of the neighbor.

This technique is problematic in several ways. It requires lots of storage for large problems; and the search over this storage will also be large. Also, every coefficient of the vector is treated equally without scaling. If the variance of coefficient A is large compared to B, then A will be much more influential than B in the distance metric. Finally, for high dimensions, often large portions of the space will be sparsely sampled by the training set.

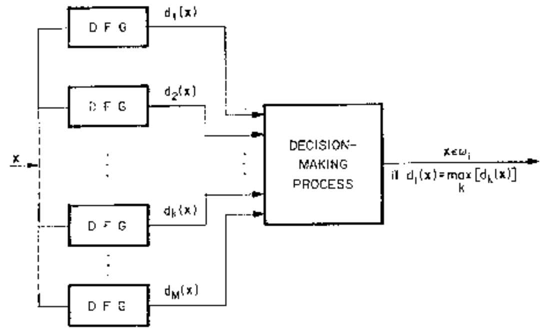

Figure 30.3: A pattern classifier using discriminant functions

30.4.2 Discriminant Analysis

In discriminant analysis, the main objective is to develop a function which takes a vector and a class and returns a scalar which is maximal if the vector belongs to the class, and minimal otherwise.[1] This function will determine *decision surfaces* between classes, where the value of the function is the same for two classes for the vector. In general, we can have many discriminant function generators (DFGs) which can then be used by a decision making process to determine the best class for the pattern (as in figure 30.3).

Linear discriminant analysis gives rise to linear decision surfaces; in 2 dimensions, the surface is a line, in 3 dimensions, a plane. In general, a linear decision surface is called a hyperplane. Figure 30.4 gives an example for the 2 dimensional case. The hyperplane decision surface lies at $\mathbf{w}^T\mathbf{x} + w_0 = 0$. In the quadratic case, the equation of the decision surface is at $\mathbf{x}^T W\mathbf{x} + \mathbf{w}^T\mathbf{x} + w_0 = 0$.

30.4.3 Generalized Discriminators

Often, one may want decision surfaces which are non-linear. While simple cases like quadratic surfaces can be modeled as above, artificial neural networks can be utilized to get quite general surfaces. McCulloch and

[1] Alternatively, the function could take a vector and return a vector of scalars, one for each class, such that the number for the appropriate class is maximized.

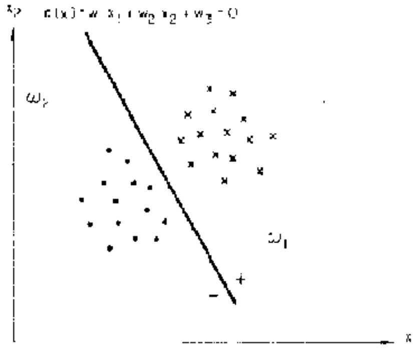

Figure 30.4: A linear discriminant surface for two dimensional space

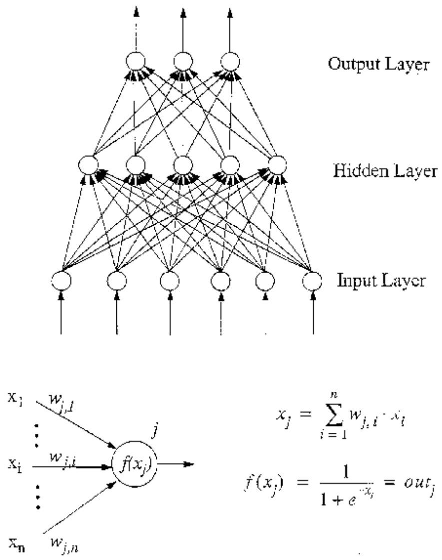

Figure 30.5: A typical MLP

Pitts first modeled neurons in 1943, using a threshold logic model. Rosenblatt developed perceptrons in 1962, which organized McCulloch-Pitts neurons into a single layer, receiving inputs from one set of nodes, and producing output on a disjunct set of nodes. Multiple layer perceptron (MLP) systems were also developed around that time by Rosenblatt's students; in a later development the thresholding function was replaced with a smoother (differentiable) nonlinear function. It can be proved that a two layer MLP with enough internal nodes (or hidden nodes) can learn any arbitrary mapping from input to output; hence, they make good general discriminators. Figure 30.5 shows a typical MLP with two layers. A unit in an MLP typically computes a nonlinear function (e.g. $out_j = f(x_j)$ in the diagram) of the weighted sum of its inputs (i.e. x_j). No computation takes place at the input layer, only at the hidden and output layers.

References

[DH73] Duda and Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1973.