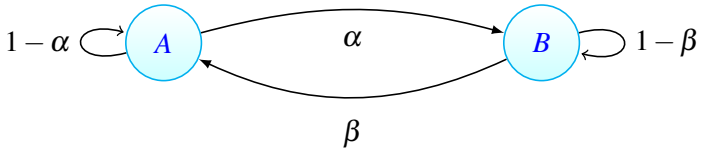
EE 178/278A Probabilistic Systems Analysis

Spring 2014 Tse/Fernandez-Granda/Hussami Lecture 18

Speech Recognition Example Continued

In Lecture 17, we discussed how to use the Viterbi algorithm to find the sequence of states (or X_is) that maximizes the posterior probability. Today, we will move on to learning the parameters of the statistical model that we use in the speech recognition problem: $Q(b|a)$ and $\mathbf{P}(a'|a)$. Example: 40 phonemes, how many parameters for the transition probabilities? 40^2, Q is a 40×40 matrix. Once we have these parameters, we can run Viterbi, but nobody gives this to us in practice! However in general these parameters must also be learnt! We will now consider this more realistic situation.



Let us analyze a very stupid language that consists of two phonemes A and B. We model the transitions between phonemes in a sentence to follow a Markov chain with transition probabilities α and β, as shown in the figure above. However we do not have access to these phonemes directly, but rather to noisy measurements of *features*. In reality the features that we would use in speech recognition would be short-time discrete Fourier transform of the speech signal and hence multidimensional. We can think of them as points in a *feature space* that captures the spectral information of the phonemes, as shown in Figure 1. Because of the noise in our measurements, these points would be randomly distributed around the *true* features corresponding to the different phonemes.

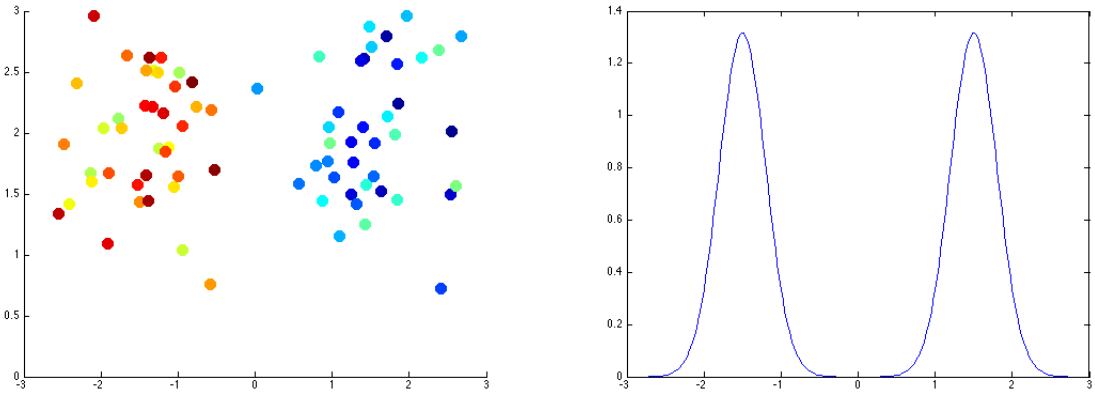


Figure 1: *Sample points in the feature space and their corresponding distribution.*

To simplify matters, we will consider one-dimensional features μ_A and μ_B, corresponding to the phonemes A and B respectively, corrupted with additive Gaussian noise of variance σ^2. Each of our measurements is consequently modeled as a random variables $Y_j \sim N(\mu_A, \sigma^2)$ if the jth phoneme is equal to A or $Y_j \sim N(\mu_B, \sigma^2)$ if the jth phoneme is equal to B.

The user of the speech recognition system says

ABAAABBABAAB

and the system measures

$y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12}$.

Again for simplicity, we will assume that we know the variance of the noise σ^2 and also the transition probabilities α and β. In practice, we could estimate the transition probabilities from any large text in the language of interest by computing the following averages (which in fact correspond to a Maximum Likelihood estimate):

$$\hat{\alpha} = \frac{\text{Number of transitions from A to B}}{\text{Number of occurrences of A}},$$
$$\hat{\beta} = \frac{\text{Number of transitions from B to A}}{\text{Number of occurrences of B}}.$$

However, we cannot do this with the features μ_A and μ_B because they will be specific to the particular person that is talking.

We now would like to estimate the sequence of phonemes uttered by the user from the data $y_1 \dots y_n$. If we knew μ_A and μ_B then we could just apply Viterbi's algorithm as in the last lecture. In the following two sections we will describe two different approaches to estimate μ_A and μ_B.

Supervised learning

Assume for a moment that we knew the sequence of phonemes. This sounds like an absurd assumption given that the sequence of phonemes is exactly what we want to estimate! However we could ask the speaker to say some predetermined words for us at the beginning (in fact most speech recognition software does this). Then we could use this sequence to estimate μ_A and μ_B using the known phonemes. In machine learning and statistics this is known as supervised learning, since we have access to some labeled data that allows us to fit our models in a *supervised* way. If in Figure 1, the points are the features observed over time, then supervised learning corresponds to the case when the points are labeled A or B.

Assume that we ask the user to say ABABBBAAB. Since μ_A is the mean of the samples where the subject says A and μ_B is the mean of the samples where the subject says B, a reasonable approach is to estimate the parameters using the sample means

$$\hat{\mu}_A = \frac{y_1+y_3+y_7+y_8}{4} = \frac{\text{Sum of samples where A is said}}{\text{Number of times subject says A}}, \quad (1)$$
$$\hat{\mu}_B = \frac{y_2+y_4+y_5+y_6+y_9}{5} = \frac{\text{Sum of samples where B is said}}{\text{Number of times subject says B}}. \quad (2)$$

This turns out to be a maximum likelihood estimator (MLE). This can be shown by considering the posterior pdf of the measurements corresponding to the phoneme A given μ_A:

$$\hat{\mu}_A = \arg\max_{\mu_A} f(y_1, y_3, y_7, y_8; \mu_A) = \arg\max_{\mu_A} \prod_i \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y_j - \mu_A\right)^2}{\sigma}}.$$

It can be easily checked that the solution to this optimization problem is precisely (1).

Notice two things in the above derivation. First that we have assumed that the data are independent given the phoneme that was uttered. Second we are applying maximum likelihood estimation to a pdf instead of to a pmf! As you saw in the homework this is fine. Let us give an intuitive explanation. If we want to estimate a parameter θ from a realization of a continuous random variable by solving

$$\max_{\theta} \mathbf{P}(Y = y; \theta)$$

in analogy with the discrete case, then we have the problem that the probability of any particular realization is zero! However let us consider a small interval around y:

$$\max_{\theta} \mathbf{P}(y < Y \leq y + \Delta; \theta) \approx \Delta \max_{\theta} f(y; \theta)$$

where Δ is a very small constant. When it is sufficiently small (in particular in the limit where $\Delta \to 0$) the approximation is precise and since Δ is a constant we can just maximize the pdf.

Unsupervised learning

Now let us take a look at the unsupervised learning approach: no known sentence! Again, we want to estimate μ_A and μ_B from the samples $y_1, \ldots, y_n$. However, our data is not labeled as before. Therefore, we cannot just group the samples corresponding to phoneme A and average them as we did in the supervised learning case. This is a common situation in practical applications. It is similar to a clustering problem, which is a fundamental problem in machine learning and statistics. We need to estimate 2 types of unknowns, the hidden variables and the parameters, jointly.

Using the law of total probability

$$\max_{\mu_A, \mu_B} f(y_1, \cdots, y_n; \mu_A, \mu_B) = \max_{\mu_A, \mu_B} \sum_{a_1, \ldots, a_n} f(y_1, \cdots, y_n | X_1 = a_1, \cdots, X_n = a_n; \mu_A, \mu_B) \mathbf{P}(X_1 = a_1, \cdots, X_n = a_n). \tag{3}$$

Even evaluating the objective function is computationally intractable as there are 2^n terms in the objective function. Therefore, optimizing this objective function over the parameters is impossible in practice. Since we are engineers, we don't give up. Let us abandon the math for a second and go back to the picture in Figure 1. We are given the points (the sequence of feature values $y_1, \ldots, y_n$) and we want to estimate the labels (A or B) of the points as well as the means μ_A and μ_B . We are trying to fight two battles at a time. We will assume one guy is known, and try to estimate the other guy. While trying to train Siri on a particular speaker, we usually have some idea that the feature corresponding to A should be around a certain value, and the feature corresponding to B around another value. (Maybe averaged values across the whole population of speakers of this weird language.) We can use these as our initial estimates which we call $\mu_A^{(1)}$ and $\mu_B^{(1)}$. We freeze these estimate and then try to infer the value of the hidden random variables X_i's, i.e. the labeling of the points. Naively, we can estimate X_i to A if y_i is closer to $\mu_A^{(1)}$ than to $\mu_B^{(1)}$, etc. However, this naive method does not take into account the transition probabilities and the fact that we are estimating the phonemes of an entire sentence at the same time rather than estimating each phoneme separately. In fact, given the current estimates of the parameters , we should be seeking the MAP estimate of the sentence given the sequence of feature values, and this problem we already solved: the solution is given by the Viterbi algorithm!

Once we have the label estimates, this becomes a supervised learning problem with training data obtained from the Viterbi algorithm. Overall, we have the following steps:

1. Obtain an initial estimate of the parameters
2. Iterate the following two steps:
   (a) E step: Run Viterbi to get MAP estimate of the sequence of the hidden random variables (the phonemes).
   (b) M step: Generate new estimates of the parameters given the estimated sequence of hidden random variables (phonemes).

This algorithm is an application of the Expectation-Maximization (EM) algorithm. We can prove that it converges because it improves the value of the likelihood function every step, but we cannot prove that it gives the solution of the overall MLE problem (3). The likelihood function is highly non-convex, so the solution can be trapped in a local optimum. Note that in the E-step, we are replacing the sum in the expectation by a single term that is the most likely sequence of hidden random variables given the observations. This is why it's called the expectation step.

A similar approach is used in clustering of points: one can alternate between estimating the centers of the clusters and the labelling of the points. This is called the *k-means* algorithm, another example of an EM algorithm. The main difference between the clustering problem and the speech recognition problem is that in the former problem, the points are assumed to be independently drawn from the population, while in the latter problem, the points correspond to the sequence of feature values from the phonemes uttered and are hence not independent. The implication of this difference is that the E-step in the clustering problem can indeed be achieved by separately labeling each point depending on which center it is closest to, while the E-step in the speech recognition problem requires running Viterbi.