The Target Cost Formulation in Unit Selection Speech Synthesis

Paul Taylor

Engineering Department, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK

Abstract

We review the various approaches that have been used to define the target cost in unit selection speech synthesis and show that there are a number of different and sometimes incompatible ways of defining this. We propose that this cost should be thought of as a measure of how similar two units sound to a human listener. We discuss the issue of what features should be used in unit selection and the pros and cons of using derived features such as F0. We then explore some algorithms used to calculate target costs and show that none are really ideal for the problem. Finally, we propose a new solution to this that uses a neural network to synthesise points in acoustic space around which we can build new clusters of units at run time.

Index terms speech synthesis, unit selection, target cost, decision trees, neural networks

1. Introduction

The now standard formulation of unit selection described in Hunt and Black [4] defines unit selection as a process whereby we examine various sequences of units from a database, and choose the one which gives the lowest total cost for the sentence we wish to synthesise. This total cost is calculated from separate *join costs*, which measure how well two adjacent units join, and *target costs*, which we will now discuss. In the original paper, Hunt and Black describe this as "an estimate of the difference between a database unit and a target unit". Most other papers use a similar definition, using terms such as "the degree to which the database unit matches the target", or "a distance measuring the similarity" between the database and target unit.

The main issues with the target cost are that we know that simply measuring the "objective" distance between two sets of features rarely results in distance that corresponds with human perception.

2. Defining the purpose of the target cost

First let us consider a very benevolent system which does not suffer does any data sparsity problems. This can either be achieved by having a very large database, or else by only having a small number of features that we use for selection. At run-time, our algorithm iterates through each item in the **specification**, a description of what we require that has been generated by the text analysis module. The algorithm finds the target feature values, and simply finds the units in the database which match this. As there are no sparse data problems there are plenty of examples of units matching the specification. It can be argued that we are now done with respect to the target cost; we have plenty of examples of exact matches, and we will now use the join cost to find final selection. We have two further possibilities. Firstly, we could consider other units; but why should we? By definition none will be exactly what we want, and why would be ever want to choose a unstressed unit when we actually want a stressed one? It can be argued that the other units in the database are simply "wrong" and should not be considered.

In a slightly more common situation, we of course *do* have sparse data problems, which manifest themselves in there being either insufficient units within the desired class, or no units at all within the class. As we increase the number of features we wish to consider, low and zero occupancy classes become the norm, such that often we never have exactly the units we want. In such cases we have to consider non-matching units, and here we hit the nub of the problem - while we have to consider other units to give the join cost a chance, all the other units we consider are simply "wrong"; the listener will hear that the units aren't the ones they are supposed to be and the speech will not sound as it should.

It turns out that the situation is not as bleak as just portrayed; this is because the acoustic space which units from a particular feature combinations lie often overlaps with the acoustic space from other units. This many-to-one mapping is of course what makes speech recognition difficult; it is well known that acoustic spaces associated with one phone overlap with those of other phones, and this extends to different feature definitions within phones as well. This leads us to our first definition of target cost, namely that it should be a measure of *how similar two units sound*. If two units sound the same to a listener, then it is safe to use one in place of the other. Significantly, in such cases the listener will not realise a unit with the "wrong" feature description is being used. We shall call this type of target cost **perceptual cost** as it is purely a measures of how similar two units sound to a listener. Note that this is different from both linguistic cost and acoustic cost, which are discussed below.

Now let us consider a different formulation of target cost, which we shall call **linguistic cost**. This is a cost that operates purely in linguistic terms; acoustics are irrelevant. To take a trivial example, suppose we are required to synthesise the word "big", but find we have no suitable units for that word. We do however have plenty of good units for the word "large", and we could use those instead. The point is that *linguistically* the resultant sentence is very similar, but is obviously quite different acoustically. To consider a slightly more plausible example, consider an input sentence which has a comma, indicating a phrase break, in the middle.

```
''In recent years, the rate of progress
has slowed''.
```

Let us assume that the text analysis module decides to place a phrase break after the word "years". During synthesis we find that we have no units of that type but we do have plenty of good units with a non-phrase final feature. We choose them instead and effectively synthesise

```
''In recent years the rate of progress
has slowed''.
```

Note that this will not sound the same as the version with

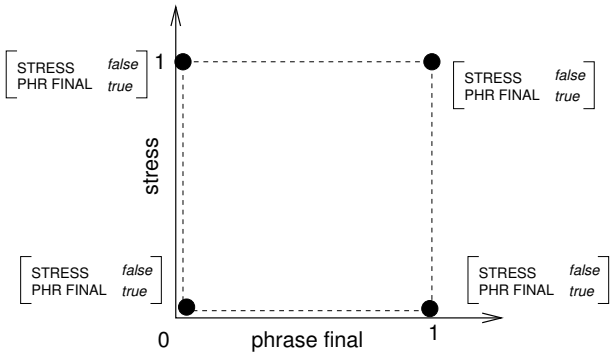

Figure 1: The standard target-cost formulation projected onto a perceptual space. Each unique feature combination lies at the corner of a hyper-cube, which is a square in this case as we are only considering two features.

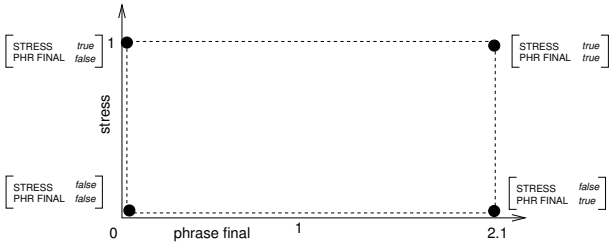

Figure 2: The effect of applying weights to the perceptual space is to scale each axis. Here phrasing has a higher weight than stress which means that axis is scaled and any difference in phrase values will result in a higher overall cots.

the phrase break: a listener could easily tell the difference. None the less this choice of unit is perfectly acceptable in this situation.

The issue of linguistic costs arises when we have a considerable number of features in our system. Imagine in addition to the above costs, we have features describing emotion and other effects. Now we can, as system designers, rank the importance of the features. We might decide for instance that making the message intelligible is more important that generating any specific emotional effect. Again, we are not trying to bluff the listener into thinking that they really did hear the original specification synthesised; rather we are prioritising what we can synthesise and hoping that will do. These priorities can be reflected in the weights the linguistic features are given in the target cost calculation.

3. High-lever versus low level-features

We now turn to the issue of what features we should use for out specification and units. In older diphone synthesis systems, the norm was that the specification consisted of a list of diphones each with a duration and an F0 value or values. There was no need for a target cost as often there was no choice of unit; the single instance was used and signal processing was used to adjust the pitch and timing. Hence the use of F0 and duration is in line with the operation of the previous generation of systems. The point of note is that it is often the case that the F0 and duration values are generated by algorithms which make use of high level features as input. Furthermore, these high level features are often exactly the same as those now used in the unit selection. This shows the proper way that we should then think about these values. In a sense, they are a low level transformation of the same information that the high level linguistic features hold. So the case can be made that it is not necessary to have F0 and duration generation algorithms as the information required to generate these values is available to the unit selection algorithm.

This illustrates the basic dilemma when choosing features. Either we can use high level features which we have a high degree of confidence in, but which can be highly redundant, leading to spare data issues. Or, we can use lower level features such as F0 values, which while compact, are often inaccurately calculated. There is no real right and wrong with regard to this issue, it very much depends on the database, quality of the F0 algorithm and so on. Informally however, we believe that given the above points there is very little to gain from the explicit generation of F0 and duration and that systems which use just high level features perform better.

4. Calculating target costs with weights

In the Hunt and Black formulation, the target cost is calculated by summing a number of weighted sub-costs:

$$T = \sum_{j=1}^{P} w_j C_j(s_j, u_{ij}) \quad (1)$$

where P is the number of features and sub costs, s_j is the value of the specification for feature j, u_{ij} is the value of unit i for feature j and w_j is the weight for that sub cost. While a variety of means have been proposed for training these weights, it is worth noting that setting these weights simply by hand seems to produce very good synthesis [2]. It is possible that this is because weights set by hand incorporate the notions of acoustic and linguistic cost.

Equation 1 can be viewed as defining a **perceptual space**, and projecting each feature description into that space such that we can measure distances. Each feature is represented by its own dimension in this space, so that the total number of dimensions matches P, the total number of features. Each unique combination of features lies on the corner of a hyper-cube of dimensionality P, and this is shown in Figure 3. By giving a feature a particular weight, we are in effect scaling that axis by that amount. As absolute distances are used, the total distance between two points is calculated by finding the nearest path that is perpendicular to the axes. The effect of adding weights is shown in Figure **??**.

A vital point about this target costs calculation is that it assumes that all features operate independently. That is, the effect that say a phrasing difference has on the overall cost is never affected by any of the other features. This is a very strong claim, and not really supported from experience. A demonstration of this is that in this scheme, two different feature descriptions will always have a non-zero distance. This contradicts what we know from above, namely that ambiguity is rife in speech and that there are frequent cases of different feature descriptions mapping to the same acoustics (and hence same sound). So long as a weight is not zero (which would imply that that feature is irrelevant in every single case) then we can not make use of the power of ambiguity in this weight system.

To show a real example of this, example the simple feature system shown in Figure **??**. It is well known that both phrasing

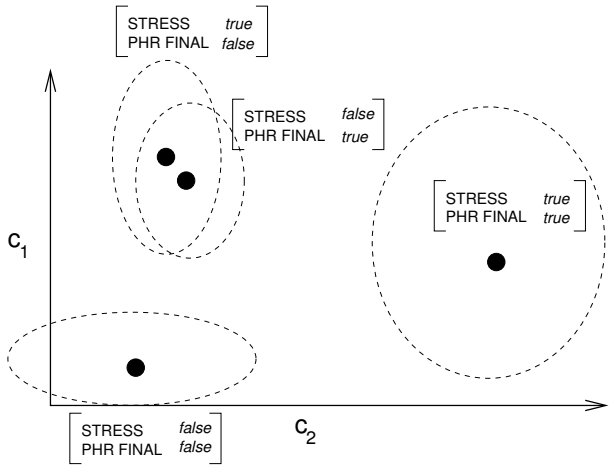

Figure 3: The perceptual space as defined by the decision tree method. Here the axes represent two of the cepstral coefficients. The feature combinations lie in positions defined by their acoustic values.

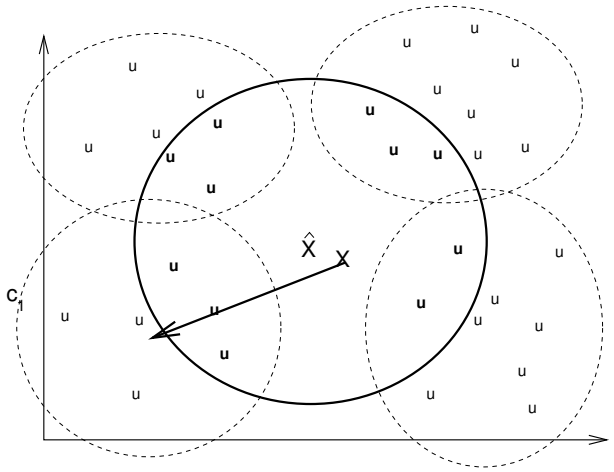

Figure 4: The difference between the decision tree and neural network approach. The observed units sharing the same feature combination are shown in clusters defined by the dotted lines. For an unobserved feature combination whose true value is X, the decision tree assigns a set of units according to the entire cluster from an observed feature combination. This is shown by the arrow. In the neural network approach, a value $\hat{X}$ is estimated by the network and a new cluster, formed from the nearest N points, is drawn around this value, shown by the solid line. This means that the closest N observed units can be used, regardless of whether they share the same features or not.

and stress lengthen a phone and so the effect of add [+phrasing] and [+stress] to a unit can be seen as somewhat similar. We would therefore expect that the target cost function would give a shorter distance between these values than otherwise. However, we can see from Figure **??** that units with [+stress, -phrasing] and [-stress, +phrasing] have in fact equal highest distance, and furthermore this is exactly the same distance as that between the combinations [-stress, -phrasing] and [+stress, +phrasing] which we would expect to sound completely different.

5. Target costs and decision trees

An alternative to the Hunt and Black target cost formulation is to use decision trees. Here the idea is to cluster units into groups of a minimum size, and then link these to the features by use of a decision tree which asks questions of the features. This technique is inspired by state tying in automatic speech recognition (ASR), and is indeed used by HMM synthesis systems such as Tokuda [5], as well as hybrid HMM/unit selection systems of Donovan and Woodland [3] and pure unit selection systems such as Black and Taylor [1].

In these schemes, there is no target cost per se; given the features the decision tree selects a cluster, and all the units and only the units within that cluster are used in the search. Modifications to this scheme exist, where the units within the cluster are weighted according to how close they are to the centre, or where a back-off strategy is used to go back up the tree to include more units if a particular cluster is somewhat sparse.

It is not straightforward to compare this with the weights method but our view is that in essence this method is effectively equating the acoustic space (usually cepstral) with the perceptual space as previously described. In other words, there is no need to define a separate perceptual space; the cepstral space serves this purpose. As we can measure points in this space directly from data, then we have no training or defining of this space to perform. Figure **??** shows that crucially this formulation does not impose a requirement of feature independence. The diagram shows that as the cost function can freely cluster the data, feature combinations which sound similar can be given low costs, regardless of how different the actual features themselves are.

There are however potential problems with this approach, many of them stemming from the fact that the feature description we require has not been seen during training and so we have to use the decision tree to give us another model instead. It is quite common to end up with many millions of possible models, of which only a few thousand have actually been observed in the training data. A further problem is that the tree partitions the data into clusters once, and these are fixed at run-time. Thus when we need units for a feature combination unobserved in the data, we use the decision tree to give us a whole set of units from a different feature combination.

6. Projecting linguistic descriptions onto an acoustic space

Here we propose a new algorithm that acts as a replacement for the decision tree. As such it does not use a weighted target cost per se, but selects units based on clustering criteria in the acoustic space.

We are trying to solve two problems; first we wish to handle cases where we only have a few units for a particular feature description. Secondly we wish to handle cases where there are no observed units for the feature description.

In both cases we make use of the idea discussed above, namely that the target cost is really a measure of how similar two units sound, and if we have a unit of one feature description that is similar to a unit with a different feature description then we can use that unit when the other set of features are requested. In doing this, we assume that cepstral distance is indeed a good measure of perceptual distance.

For the low occupancy case, we build a similarity table for every unit in the database. We first do a state alignment with an HMM trained on the phones of that unit and then compare and sum the total Euclidean distances for each state. For each base type, we calculate the full matrix of costs between every unit for that base type and every other unit. At run time, we find the feature description we want from the specification and pass the units that have that feature description into the search. If the number of matching units is below a defined minimum threshold, say 50, then we use the distance metric to pad the occupancy of that feature description. So for instance, if we have 10 units matching our feature description, for each of those 10 units we would find the closest 5 other units within the base type, making 50 in total. The distance measures are performed offline, but it is computationally trivial to pad the feature description classes, so this can be done offline or at run time.

For the unseen case, we can't use the distances alone as we have no existing units which we can use as a guide. Our solution here is to create a *function* that maps from the linguistic space to a point in the acoustic space, and use then perform the distance measures to this point to find the nearest N units.

In principle any trainable function approximation algorithm could perform this task. In our system, all the linguistic descriptions are represented as binary features - this in fact is the same formulation required to ask "questions" in decision tree growing. The idea is to learn the general influence of the features on the acoustics with the hope of generalising to the cases where we have unseen feature descriptions. While we cannot of course be sure that an algorithm will do this correctly it is worth pointing out that while the features definitely do not operate independently (as explained above) they do have certain regularities with respect to their acoustic mappings, and this helps greatly in the learnability of this function. For instance stressed units are always longer and louder than their unstressed equivalents and phrase final syllables are always longer than no phrase final ones.

To perform this mapping, we used a standard back-propagation neural network. Despite being somewhat unfashionable these days, a neural network fits our requirements really quite well. In particular it is good at dealing with multi-dimensional binary data, as in our case. Three neural networks were trained, one for each state as determined by the automatic alignment. Each network was trained on all the data for that state, and to avoid the mapping being swamped by the phone identity, the data for each state was normalised by using the mean and variance as measured from all examples with the same phone identity.All three networks had identical topologies, which comprised 38 input nodes (corresponding to the 38 binary features we happened to be using), 20 hidden nodes and 42 output nodes for the 12 cepstral coefficients, energy, F0 and their delta and accelerations.

At time, when we encounter a feature description that was unseen at training time, we use the trained networks to generate three points (one for each state) in acoustic space. We then proceed as if these points represented a real unit, and we find the nearest N units to these points and pass them into the search. To test our system, we built a baseline system using the standard decision tree method, and compared this with a system that was the same in all respects, apart from the two additions just described.

The improvements were slight but significant. Out of 20 test sentences, 14 were judged the same or equally as good, 1 was ranked better with the decision tree method and 5 were ranked better with the neural network method. Significantly, upon examination of the units chose, it was found that in 4 of the 5 better cases, the neural network approach was choosing units which were not in the cluster used by the decision tree. In other words, the decision tree had "pruned" these units and they were no available to the search.

7. Conclusions

In summary:

1. It is important to separate the notions of acoustic, linguistic and perceptual cost.
2. The target cost as traditionally formulated should be thought of as a "sounds similar" measure, assessing whether it is possible to substitute one unit for another with minimal perceptual effect.
3. The weighted sub-cost technique effectively defines a perceptual space, projects feature descriptions into it, and provides a means of measuring distances within it.
4. The weighted sub-cost technique (no matter how the weights are trained) restricts all sub-costs and features to be independent. We believe this is too strong a constraint; as we know that features interact and a consequence of this is that we can not use the ambiguity naturally occurring in the data to our advantage.
5. While the decision tree method can make use of ambiguity (and in fact always does), the number of unseen feature descriptions is very large and the clustering imposed by the tree can be overly crude.
6. Our new technique effectively defines a new cluster at run time for every feature description, meaning that more "fluid" partitioning of the space is possible.

Although our technique has shown improvement we believe there is still significant progress to be made. In particular, we are uneasy about the use of the cepstral space to represent a perceptual space. This may be true in some very low level psychoacoustic sense, but this is not defendable in general. A possible solution is to try to learn an additional hidden perceptual space which has the correct dimensionality and which reflects human judgements.

8. Acknowledgements

The author is supported by the Royal Society and he wishes to gratefully acknowledge that support. Thanks also go to many colleagues in the Machine Intelligence lab, especially Ollie Williams for useful discussions concerning this work.

9. References

[1] BLACK, A., AND TAYLOR, P. Automatically clustering similar units for unit selection in speech synthesis. In *Eurospeech97* (Rhodes, Greece, 1997), vol. 2, pp. 601–604.

[2] COORMAN, G., FACKRELL, J., RUTTEN, P., AND VAN COILE, B. Segment selection in the l&h realspeak laboratory tts system. In *ICSLP* (2000).

[3] DONOVAN, R., AND WOODLAND, P. Improvements in an HMM-based speech synthesiser. In *Eurospeech95* (Madrid, Spain, 1995), vol. 1, pp. 573–576.

[4] HUNT, A. J., AND BLACK, A. W. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP '96, Atlanta* (1996), IEEE, pp. 373–376.

[5] TOKUDA, K., KOBAYASHI, T., AND IMAI, S. Speech parameter generation from hmm using dynamic features. In *ICASSP* (1995).