2.5. Methodology

Figure 2.2: Example of applying temporal smoothing to LSF parameter using a sliding Hanning window. The width of the window was varied, to impose varying amounts of smoothing. Figure 2.2 shows an example of this process.

2.5.1.2 Variance scaling

Variance adjustment was implemented as a simple scaling of the standard deviation by a fixed factor. For each parameter (i.e., each LSF) in turn, the mean value over the utterance was found and subtracted before multiplying the parameter by a scalar value, and finally adding the mean back in. By altering the scalar value, the standard deviation is correspondingly adjusted, to simulate both reduced variance (which is commonly observed in HMM synthesis) and increased variance (e.g., as may happen if a Gaussian p.d.f. is poorly estimated during training, or when GV fails to re-instate the appropriate amount of variance). This approach of variance scaling is similar to the postfiltering method investigated by Silén and Helander (2012). Figure 2.3 shows an example of this process.
Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network
What you should already know

• Text processing in the front end
  • what the available linguistic features are
  • how they can be flattened on to the phonetic sequence
  • how categorical linguistic features can be treated as binary

• HMM-based speech synthesis
  • how questions in a regression tree use those binary features
  • typical speech parameters used by vocoders

“one-hot” encoding

also known as

1-of-K or 1-of-N
Speech synthesis using Neural Networks

• what is a Neural Network?
• doing Text-to-Speech with a Neural Network
• training a Neural Network
A simple “feed forward” neural network

directed connections, each with a **weight**

a hidden **layer**

units (or “neurons”), each with an **activation function**

input **layer**

**a weight matrix**

output **layer**

information flows in this direction
What is a unit, and what does it do?

A unit sums the inputs and passes them through a non-linear function, usually called the “activation.”
What are all those layers for?

- A representation of the input
- A sequence of non-linear projections
- Learned intermediate representations
- A representation of the output
Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network
Synthesis with a neural network: flatten the linguistic structure

sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao-th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...

the cat sat
DET NN VB

phrase initial
pitch accent
phrase final

sil^dh-ax+k=ae, "phrase initial", "unstressed syllable", ...

((the cat) sat)
Synthesis with a neural network: encode the context-dependent-phones

linguistic timescale  predict durations  fixed framerate

\[
\begin{array}{cccccccccccccc}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0.2 & 0.0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0.2 & 0.1 \\
& & & & & & & & & & & \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0.4 & 0.0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0.4 & 0.5 \\
& & & & & & & & & & & \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1.0 & 1.0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0.2 & 0.0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0.2 & 0.2 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0.2 & 0.4 \\
& & & & & & & & & & & \\
\end{array}
\]

sil-sil-sil+ao=th@x_A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil-sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil-ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao-th+er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th+er+ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er+ah+v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
av+dh=ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v+dh=ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
Synthesis with a neural network: generate
How to construct the sequence of input features

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.2 & 0.0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.2 & 0.1 \\
\vdots \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.2 & 1.0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.4 & 0.0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.4 & 0.5 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0.4 & 1.0 \\
\vdots \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 1.0 & 1.0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & \ldots & 0.2 & 0.0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & \ldots & 0.2 & 0.2 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & \ldots & 0.2 & 0.4 \\
\end{bmatrix}
\]
Preparing the input

- Run the front end
- Obtain linguistic specification

 linguistic timescale

at a fixed framerate

sequence of context-dependent phones

Add duration information

Query context-dependent phones using yes/no questions, to obtain binary features

Frame sequence

Add fine-grained positional information

Sequence of context-dependent phones

• Run the front end

• Obtain linguistic specification

linguistic timescale

at a fixed framerate

Sequence of context-dependent phones

Add duration information

Query context-dependent phones using yes/no questions, to obtain binary features

Frame sequence

Add fine-grained positional information
Preparing the input:
flatten linguistic specification

sil^dh-ax+k=ae, "phrase initial", "unstressed syllable", ...

linguistic timescale: phones

the cat sat
DET NN VB
((the cat) sat)
Preparing the input: a sequence of context-dependent phones

“Please call . . .”

#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4#  . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4#  . . .
l~i-z+k=0:4_1/A/0_0_0/B/1-1-4:1-1&1-4#  . . .
i~z-k+l:w:1_3/A/1_1_4/B/1-1-3:1-1&2-3#  . . .
z~k-0+l:w=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3#  . . .

quinphone positional features POS features
(e.g., position of phone in syllable)

This is the sequence of model names that we would use in HMM-based speech synthesis
Preparing the input:
predict durations at the subphone level

“Please call . . .”

#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .

p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=0:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+0=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-0+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
What is the “subphone”?  

- All early DNN systems employ HMMs as a sub-phonetic “clock” 
  - duration is then modelled at the **state** (i.e, subphone) level 

```plaintext
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
```

```
duration (in frames)  2  1  3  1  3
```
Preparing the input:
predict durations at the subphone level

"Please call . . ."

```
3900000 4000000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4000000 4050000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4050000 4100000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4100000 4150000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4150000 4200000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
```

```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
```

```
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=0:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+0=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-0+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```
Preparing the input: convert each state of each context-dependent phone to a vector of binary features

“Please call . . .”

QS "C-OI" {-OI+}
QS "C-i" {-i+}
QS "C-aU" {-aU+}
QS "C-aI" {-aI+}
QS "C-a" {-a+}
QS "C-Q" {-Q+}
QS "C-@@" {-@@+}
QS "C-I@" {-I@+}
QS "C-U@" {-U@+}
QS "C-E@" {-E@+}
QS "C-E" {-E+}
QS "C-A" {-A+}
QS "C-eI" {-eI+}
QS "C-b" {-b+}

"P l e a s e c a l l . . .”

#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
3900000 4000000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4000000 4050000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4050000 4200000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4200000 4250000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
4250000 4400000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=0:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+0=1w:1 3/A/1 1 4/B/1-1-3:1-1&2-3# . . .
z~k-0.
Position-within-phone and position-within-state features

```
00000100000100010000 . . .
```

time is now at a fixed framerate
Position-within-phone  =  state counter

000001000001000100000 . . . 2
000001000001000100000 . . . 3
000001000001000100000 . . . 4
000001000001000100000 . . . 5
000001000001000100000 . . . 6

time is now at a fixed framerate
Position-within-state feature

[Diagram]

0000010000010001000000... 2 0.50
0000010000010001000000... 2 1.00
0000010000010001000000... 3 0.00
0000010000010001000000... 2 1.00
0000010000010001000000... 3 1.00
0000010000010001000000... 4 0.33
0000010000010001000000... 4 0.66
0000010000010001000000... 4 1.00
0000010000010001000000... 5 1.00
0000010000010001000000... 6 0.33
0000010000010001000000... 5 0.00
0000010000010001000000... 6 1.00
0000010000010001000000... 6 0.33
0000010000010001000000... 6 0.66
0000010000010001000000... 6 1.00

[l] in the context #~p-l+i=z:2_3/....

with a duration of 10 frames (50ms)
Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network
Preparing the inputs and outputs for training

- **Inputs**
  - linguistic features
  - plus positional features (‘counters’)
  - re-write as vectors
    - \([0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ \ldots \ 0.2 \ 0.1]\)

- **Outputs**
  - same speech features (vocoder parameters) used in HMM synthesis

- Form input/output pairs, one pair per frame (e.g., every 5 msec)
  - how to get the alignment?
Training a neural network: pairs of input/output vectors

```
[0 0 1 0 0 1 0 1 1 0 ... 0.2 0.0] [0.12 2.33 2.01 0.32 6.33 ... ]
[0 0 1 0 0 1 0 1 1 0 ... 0.2 0.1] [0.43 2.11 1.99 0.39 4.83 ... ]
...
[0 0 1 0 0 1 0 1 1 0 ... 0.2 1.0] [1.11 2.01 1.87 0.36 2.14 ... ]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.0] [1.52 1.82 1.89 0.34 1.04 ... ]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.5] [1.79 1.74 2.21 0.33 0.65 ... ]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 1.0] [1.65 1.58 2.68 0.31 0.73 ... ]
...
[0 0 1 0 0 1 0 1 1 0 ... 1.0 1.0] [1.55 1.03 3.44 0.30 1.07 ... ]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.0] [1.92 0.99 3.89 0.29 1.45 ... ]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.2] [2.38 1.13 4.02 0.28 1.98 ... ]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.4] [2.65 1.98 3.94 0.29 2.16 ... ]
...```

○○○○ ○○
Training a neural network: back-propagation

![Diagram of a neural network with input, hidden, and output layers. The input layer has three nodes with values [0, 0, 1]. The output layer has two nodes with target values [0.12, 2.33].]
Orientation

• Simple neural networks
  • feed-forward architecture

• Constructing the input features
  • converting categorical features to binary
  • mapping linguistic timescale to fixed frame rate using the duration model

Early work borrowed a duration model from an HMM system. Later work uses a better duration model,
a straightforward replacement for the regression tree.
What next?

• Neural networks for speech synthesis
• from this point onwards, we must rely on reading the literature

• Hybrid synthesis
• in the next module, we’ll use our Neural Network as the basis for an **ASF target cost function**