Paper title: Prediction Interval with Examples of Similar Pattern and Prediction Strength

Introduction:

In early sections, this document describes how to run and modify the code for different signals. In later sections, this documents describes the flow of the code.

How to run the code:

This documents have two master files-

1. Master string file- Master_PI_file_string.m
2. Master point file- Master_file_corr_single_PI.m

The user can run those files at first without any editing.

Running the Master string file, one can observe a portion of Fig. 3 (of paper). In order to get the entire part of fig. 3, the user need to set –

```
prediction_start=4000;
prediction_end=24000;
search_length=20000;
corr_length=20;
```

The simulation of the entire figure will take 5 to 30 minutes in commonly used desktops.

In order to reduce the simulation time, we are setting-

```
prediction_start=19000;
prediction_end=19500;
search_length=2000;
corr_length=20;
```

With this default value the user can see a reasonable output within a few seconds. When the search length is small (2k), the execution becomes faster but the second inauguration of speech needs further training.

Running the Master point file, one can observe a bar chart similar to Fig. 2 (of paper). Before going to the final figure, there are several observation figures-

Figure 1 presents last 2k samples.

Figure 2 presents the value of correlations (blue curve) and the ratio between segments over the search length (green curve).

Figure 3 presents the maximum and sorted correlation values (green curve) and corresponding normalized prediction values (blue curve).

Finally, figure 4 is the bar chart of the probability distribution.

In the command window, it provides length of the signal, the actual value of sample, median of prediction, prediction interval and the Prediction strength.

How to run the code with different signals:

In order to run different signals, the user need to change the following file-

loading_signal.m

After loading the signal the users may crop & down sample the signal. That signal should be a one dimensional array. The user also need to set a sampling frequency (=1, if no relevant information available).

Code flow: Single PI:

1. Loading signal

   ```
   [test_signal,sampling_frequency]=loading_signal;
   ```

2. Correlation of recent samples over certain length

   ```
   [result_corr,ratio_factor] =
   norm_corr_mn(test_signal(1:prediction_point), search_length,
   corr_length);
   ```

3. Finding highest correlation values (up to certain number or higher than certain threshold, whichever is lower)

   Using the values of max_point, we can find and draw similar patterns.

```
[max_value,max_point] =
get_max_value_point(result_corr,best_match_count,0.5);
```

4. Normalizing the signal with ratio and calculating the prediction weight

```
for iter=1:length(max_point)

forecast_point(iter)=test_signal(prediction_point-
max_point(iter))*ratio_factor(max_point(iter));

prediction_weight(iter)=
(max_value(iter)^5)*2/(ratio_factor(max_point(iter))+(1/ratio_fac
tor(max_point(iter))));

end
```

5. Drawing bar chart

```
bar_number=50;
[boundaries,bar_out]=bar_forecast_prediction(forecast_point,predi
ction_weight,bar_number);
```

6. Discarding less relevant regions from corners.

```
bar_number=50;
[boundaries,bar_out]=bar_forecast_prediction(forecast_point,predi
ction_weight,bar_number);
```

7. Calculating the prediction strength.

```
Prediction_Strength = sum(prediction_weight)/best_match_count
```

Code flow: String:

1. Loading signal

```
[test_signal,sampling_frequency]=loading_signal;
```

2. Initialization

3. Calling Point Correlation function in a loop

```
[Strength_PI(index_PI), Prediction_median(index_PI), PI2] =
PI_point_corr(test_signal,
prediction_point,search_length,corr_length,percent);
```

4. Plotting Signals
5. Finding PI Coverage