Trajectory Meeting

Zhang Le

CSTR

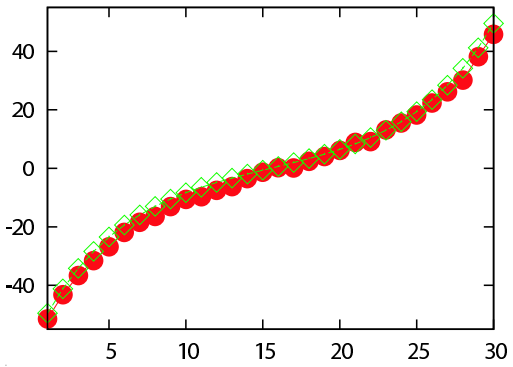
November 2006

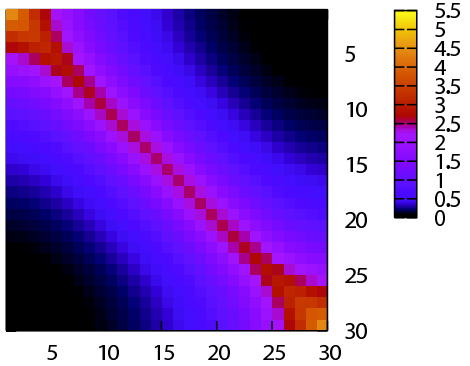
Outline

- Recap
- Implementation
- Command line utilities
- Python API

Trajectory Model

$$\mathbf{p(c \mid q) = \frac{1}{Z_q} \cdot p(o \mid q)}$$





Implementation

1. C implementation with Python binding
2. read/write HTK files (model, dict, lattice, hmmlist etc.)
3. Single Gaussian model with MLE training
4. A trajectory token-passing decoder
5. Command line tools (Condor ready)
6. Python binding

Command line tools

```
export PATH=$PATH:/home/s0450736/opt/bin/
```

Tools availiable:

1. trajectory_train.devel
2. trajectory_decode.devel
3. trajectory_score.devel

Designed as drop-in replacement for HTK (-S -l -i -T).

trajectory_train.devel

Train an MLE trajectory model.

```
trajectory_train.devel -m model/baseline/MMF \
                       -l res/hmmlist \
                       -S res/train.feat \
                       -I trj/train.realign.0.mlf \
                       -w res/dw \
                       -o trj/MMF.trj1 \
                       -i 5 \
                       -T 1
```

trajectory_decode.devel

Token-passing trajectory/HMM decoder.

```
trajectory_decode.devel -m trj/MMF.trj1 \
                        -l res/hmmlist \
                        -S res/test.feat \
                        -w res/dw \
                        -d res/dict.txt \
                        -n res/decode.network \
                        -i trj/test.decode.mlf \
                        -T 1
```

hmm_decode.devel

trajectory_score.devel

```
trajectory_score.devel -m trj/MMF.trj1 \
                       -l res/hmmlist \
                       -S res/train.feat \
                       -w res/dw \
                       -I trj/train.realign.1.mlf \
                       -o trj/score.txt
```

Python Binding

```
export PYTHONPATH=\
/home/s0450736/script/python:/home/s0450736/script:/home/s0450736/prj/l

from trj import *
from numpy import *

m = trj_model_load('res/dw', 'model/baseline/MMF', 'res/hmmlist')
trj_model_load_network_dict(m, 'res/decode.network', 'res/dict.txt')

Oseq, Tseq, Qseq = trj_load_aligned_data(m, 'res/train.feat',
        'trj/train.realign.0.mlf')

trj_train_mle(m, Oseq, Tseq, Qseq, training_type = 0, opt_order = -1,
```

```
        max_iter = 3, trace = 3)
trj_model_save(m, 'trj/MMF.mle')
```

Python API

Wrapped in trj.py, the Python API is enough to do everything from trajectory training to decoding individual utterance.

```
hmm_align
hmm_decode
trj_align
trj_decode
trj_gradient
trj_load_aligned_data
trj_load_data
trj_logprob
trj_smoothed_mean
trj_train_mle
```

Summary

1. Command line tools
2. Python API
3. C source availiable in /home/s0450736/prj/libasr (Un-compilable!)
4. Pre-alpha stage: no doc at all.

Have a look at running example in /home/s0450736/public/trajectory.example/