Machine Learning and Having It Deep and Structured

HOMEWORK #2

Structured Learning

廖宜修 沈昇勳
r03921048@ntu.edu.tw
r03942071@ntu.edu.tw

Outline

- From HW1 to HW2
- Structured SVM
- Feature Extraction
- Structural SVM Training
- Viterbi Algorithm
- SVM^{struct}
- Data format
- Homework Requirement

From HW1 to HW2

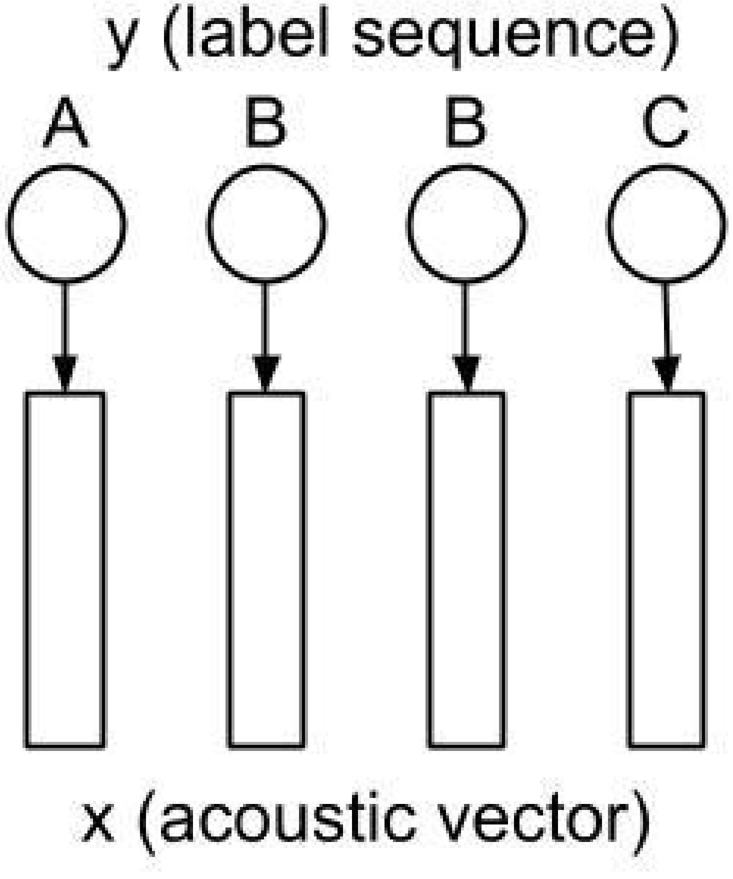
- some label sequence is impossible.
  - {..., a, a, b, a, a, a,...}

- How to model it?

- Consider all labels as a whole.
  - Structured Learning.

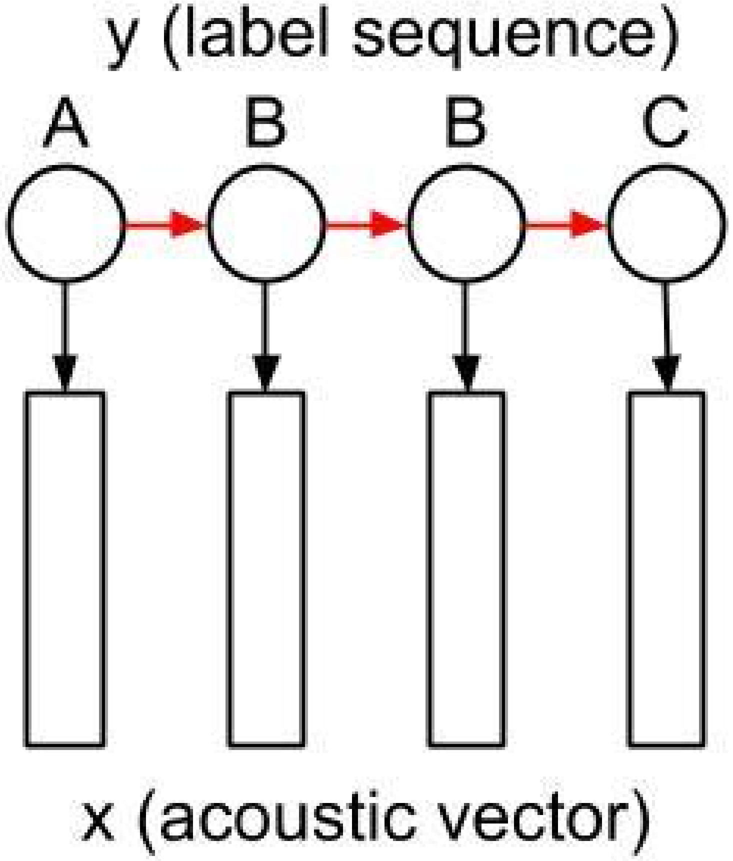
- In this homework, we are going to implement Structured SVM.

From HW1 to HW2



HW1

From HW1 to HW2



HW2

Structured SVM

- Given M frames acoustic vectors and label sequence, we can generate one feature vector

$$\mathbf{x} = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^M\}$$
$$\mathbf{y} = \{y^1, y^2, ..., y^M\}$$
$$\Psi(\mathbf{x}, \mathbf{y})$$

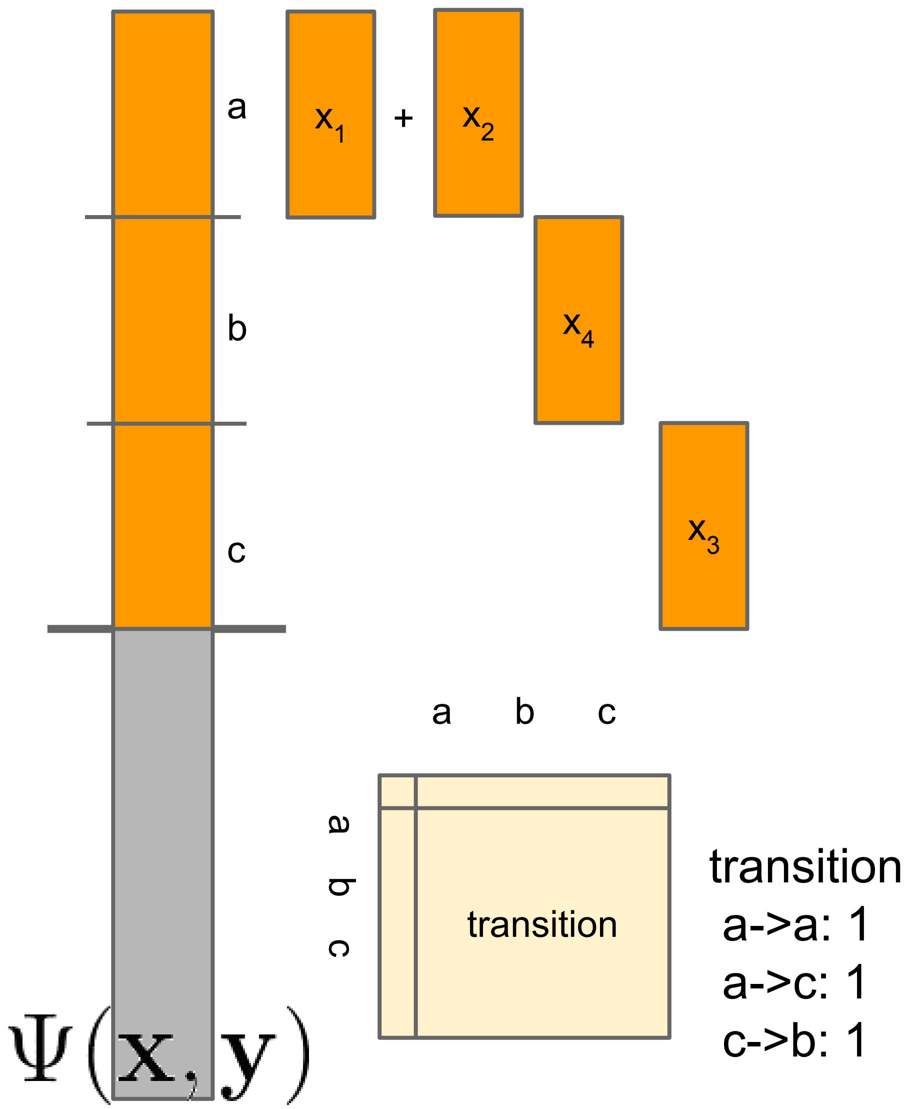
- Feature vector consists of observation and transition.

- train W such that

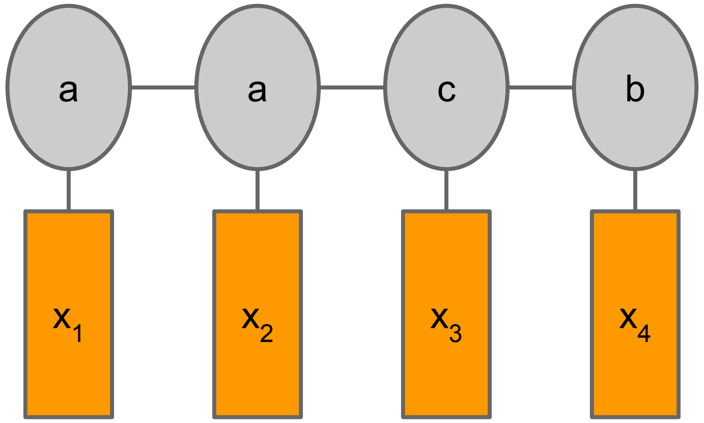
$$log(Prob\{\mathbf{y}|\mathbf{x}\}) \approx W \cdot \Psi(\mathbf{x}, \mathbf{y})$$

- Inference using Viterbi.

Structured SVM - feature exaction



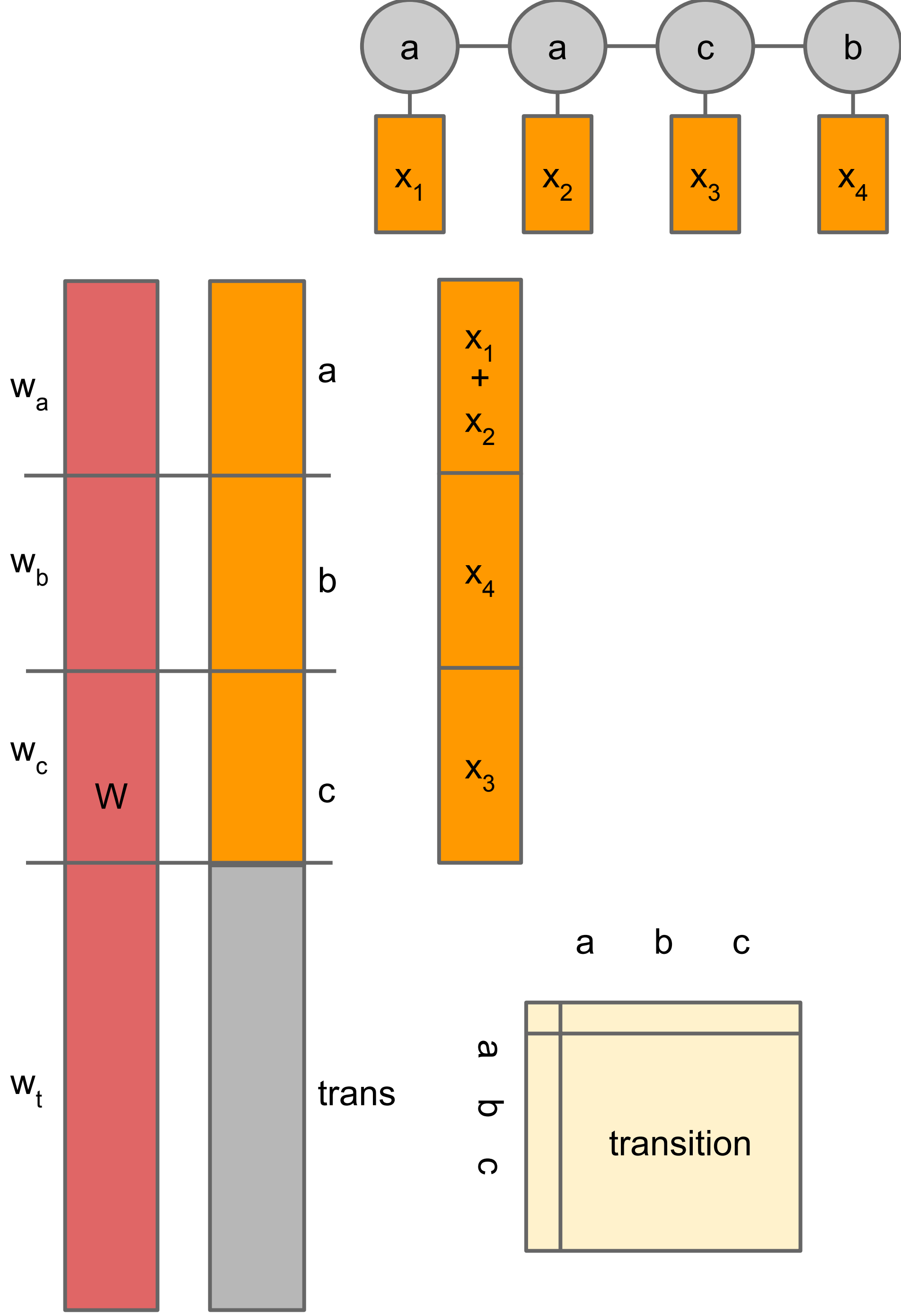
transition
a->a: 1
a->c: 1
c->b: 1



Why addition?

$w^T \cdot \Psi(x, y)$

$= w_a^T (x_1+x_2) +$
$w_b^T x_4 + w_c^T x_3 +$
$w_t^T \text{trans}$



Why addition?

$= w_a^T (x_1 + x_2) + w_b^T x_4 + w_c^T x_3 + w_t^T \text{trans}$

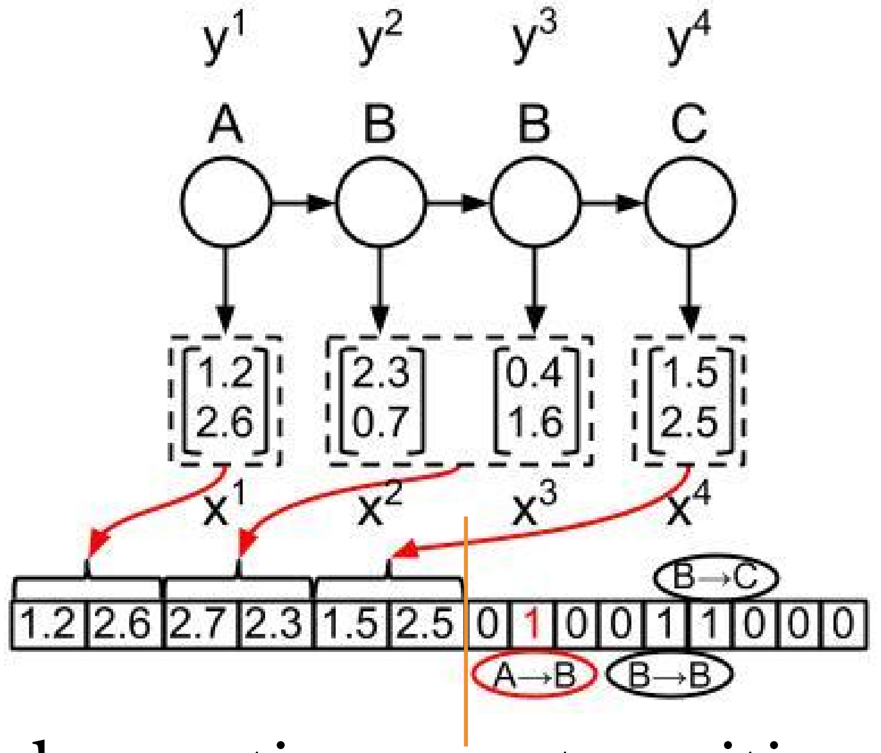
$\approx \log(P\{a|x_1\}) + \log(P\{a|x_2\}) + \log(P\{b|x_4\}) + \log(P\{c|x_3\}) + \log(P\{\text{trans}\})$

$= \log(P\{a|x_1\}\ P\{a|x_2\}\ P\{b|x_4\}\ P\{c|x_3\}\ P\{\text{trans}\})$

$\approx \log(\text{Prob}\{y|x\})$

Structured SVM - feature exaction

- A real number example.

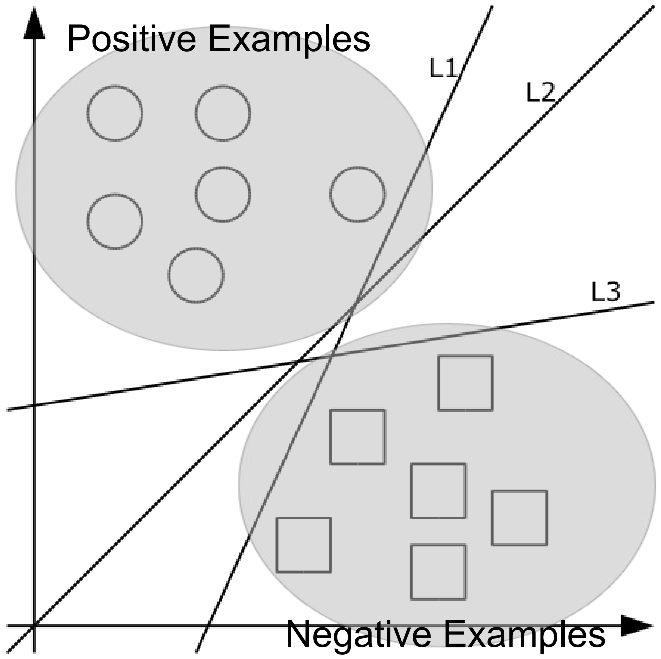


observation transition

Structural SVM Training

Support Vector Machine (SVM) is a linear classifier, and the goal for a classifier is to decide which class a new data will be in while given some data points with class labels.

There are many hyperplanes that might classify the data (Figure below). One reasonable choice as the best hyperplane is the one that represents the largest separation (margin) between the two classes.



Structural SVM Training

•**STEP 1:** Solve the SVM objective function using only the current working set of constraints.

•**STEP 2:** Using the model learned in STEP 1, find the most violated constraint from the global set of constraints.

•**STEP 3:** If the constraint returned in STEP 2 is violated by more than epsilon, add it to the working set.

Repeat STEP 1-3 until no additional constraints are added. Return the most recent model that was trained in STEP 1.

Structural SVM Training

Find Most Violated Constraint –

Similar to SVM, which also tries to maximize the hyperplane margin while given a set of positive examples and negative examples.

- Positive examples : Phone labels
- Negative examples : Solved by **Viterbi Algorithm**

Viterbi Algorithm

In order to generate the phone label sequence according to feature vectors, greedy algorithm may become impossible due to time consuming. As a result, using Viterbi algorithm is the best way to solve this problem.

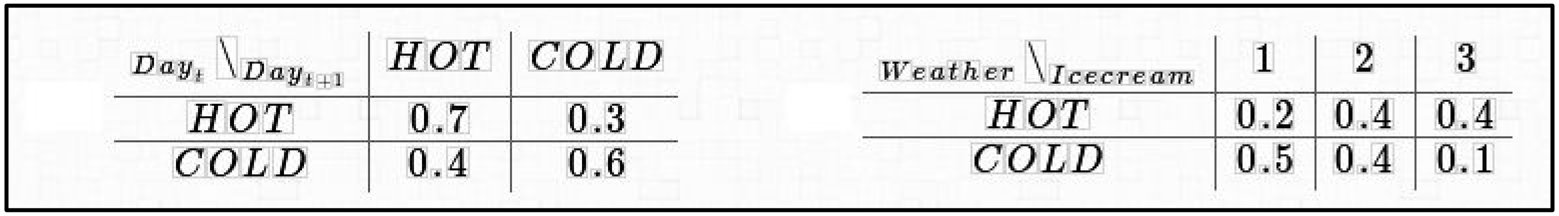
Viterbi Algorithm

For example, a boy buys ice-cream everyday and the amount is correlated to weathers.

Weather = {HOT,COLD}.

Ice-cream = {1,2,3}

The _Transition Matrix_ of ***Weather*** and _Output Matrix_ of ***Ice-cream*** is shown below :



$Day_t \backslash Day_{t+1}$	HOT	$COLD$
HOT	0.7	0.3
$COLD$	0.4	0.6

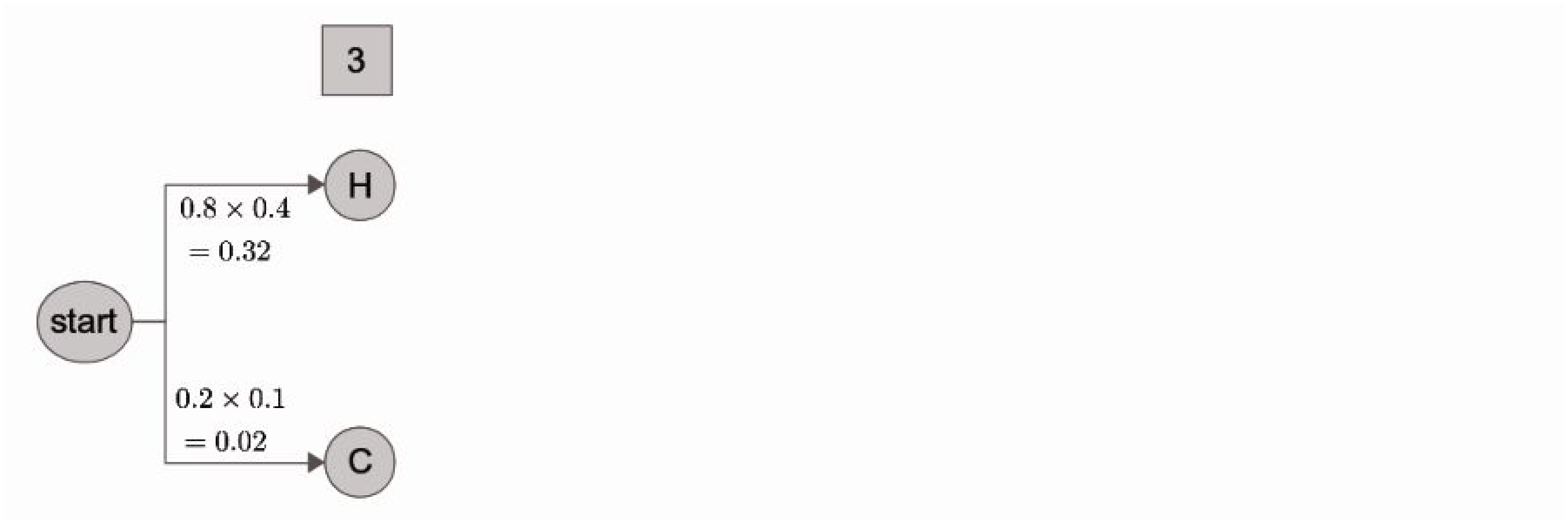
$Weather \backslash Icecream$	1	2	3
HOT	0.2	0.4	0.4
$COLD$	0.5	0.4	0.1

Ice-cream is the observed result, we need to predict the ***Weather*** (hidden state) sequence while given the observed sequence.

Viterbi Algorithm

If the observed sequence is (3, 1, 1) , we start to calculate the probability of weather (Hot, Cold) from initial state :

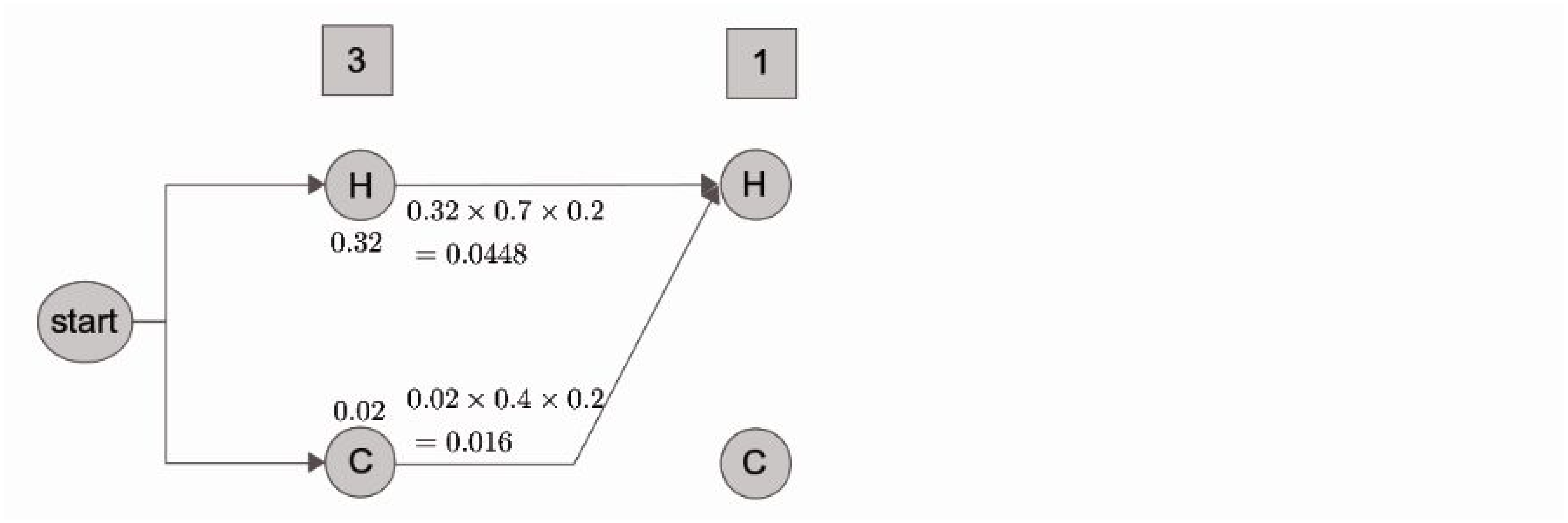
$$P(X_1 = 3, q_1 = H) = P(q_1 = H) \times P(X_1 = 3 \mid q_1 = H) = 0.8 \times 0.4 = 0.32$$
$$P(X_1 = 3, q_1 = C) = P(q_1 = C) \times P(X_1 = 3 \mid q_1 = C) = 0.2 \times 0.1 = 0.02$$



Viterbi Algorithm

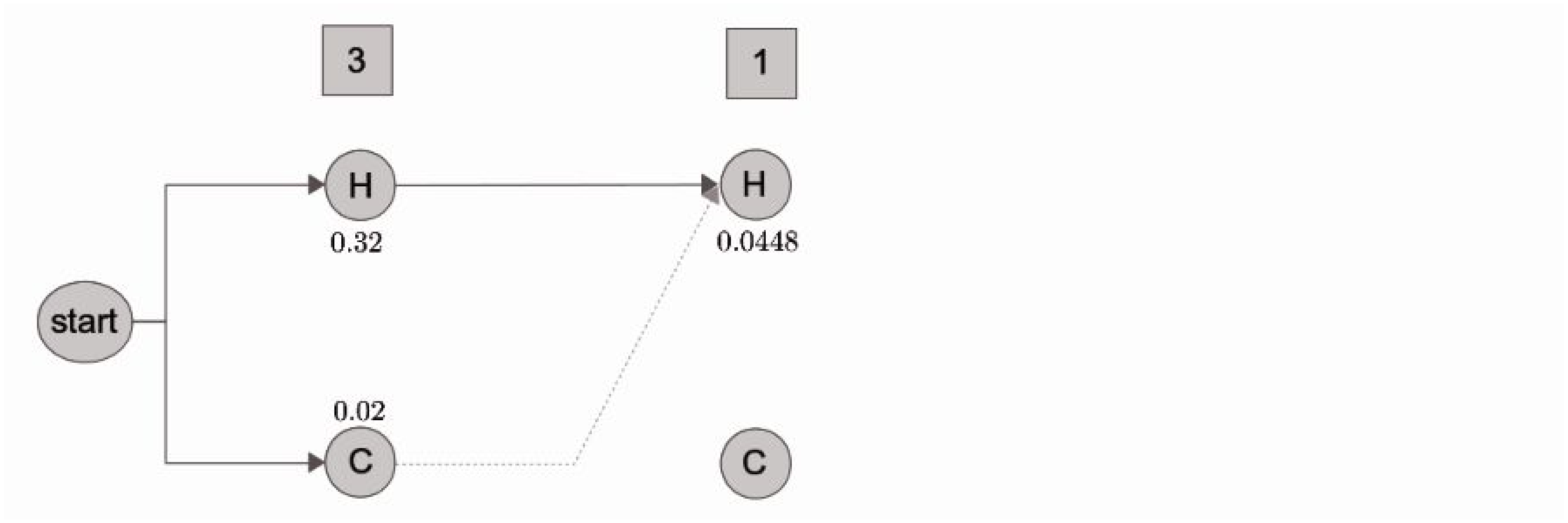
While given the probability of state *H* and *C* in observation 1, we then find the probability of state *H* in observation 2 :

$$P(X_1 = 3, q_1 = H) \times P(q_2 = H \mid q_1 = H) \times P(X_2 = 1\, mid q_2 = H) = 0.32 \times 0.7 \times 0.2 = 0.0448$$
$$P(X_1 = 3, q_1 = C) \times P(q_2 = H \mid q_1 = C) \times P(X_2 = 1\, mid q_2 = H) = 0.02 \times 0.4 \times 0.2 = 0.0016$$



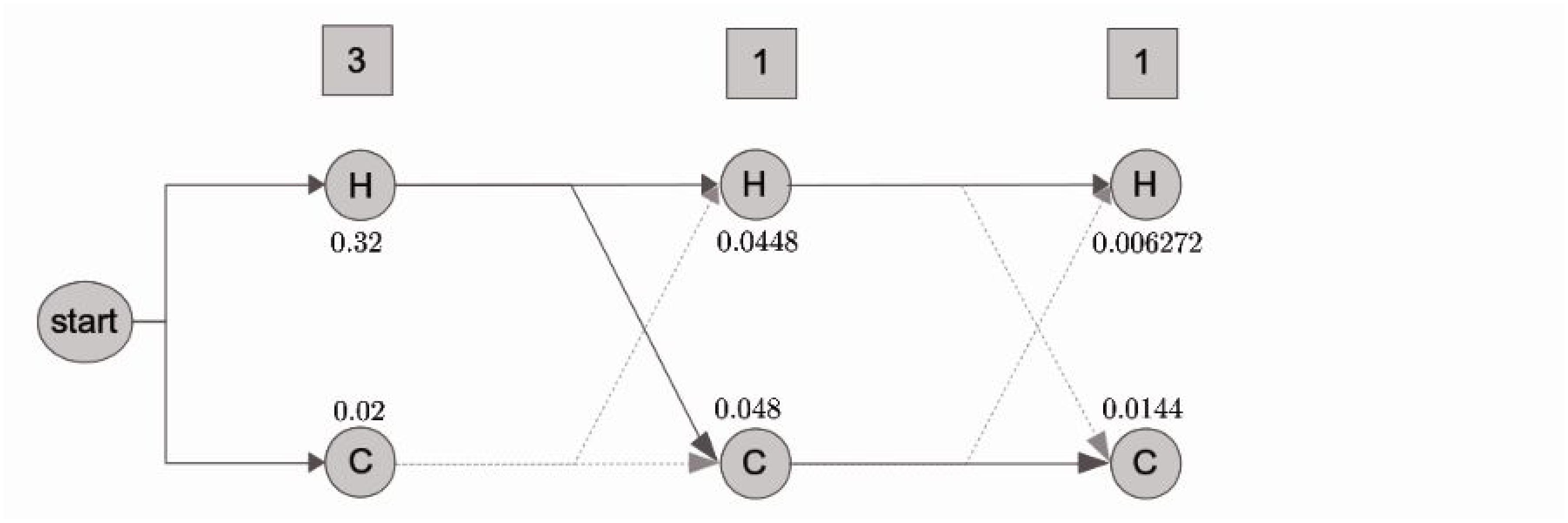
Viterbi Algorithm

The probability of sequence {*HH*} is larger than {*CH*}, so we only keep the best path {*HH*}.



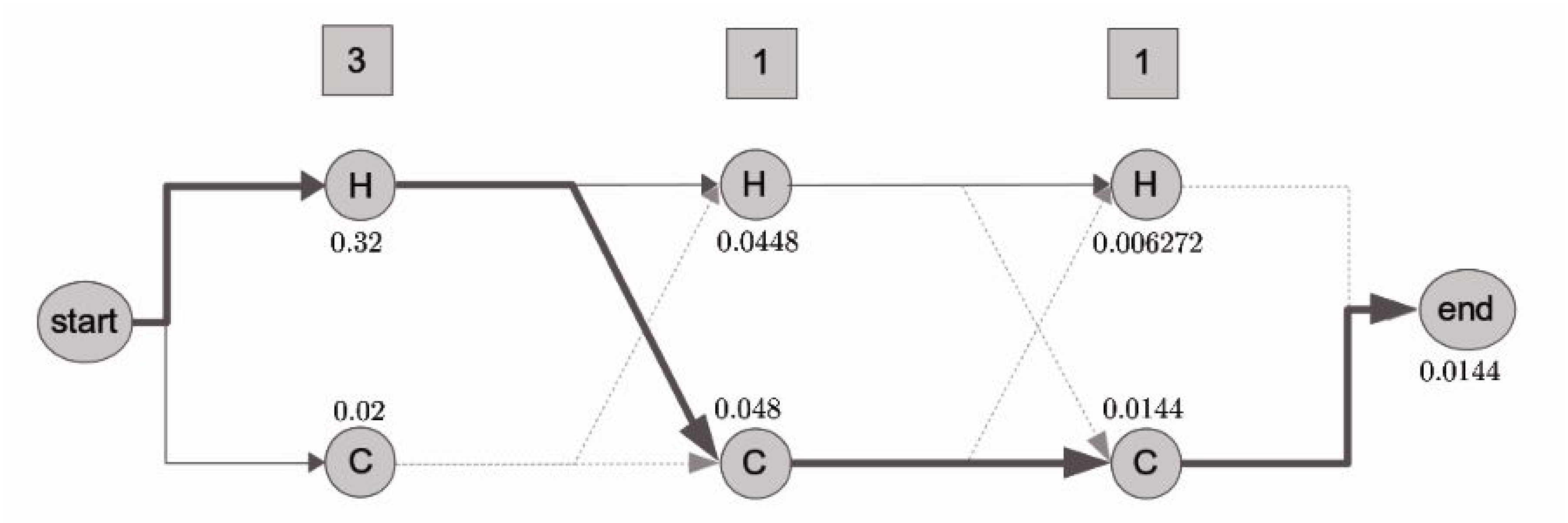
Viterbi Algorithm

Keep going.



Viterbi Algorithm

As a result, the sequence with highest probability is {*H,C,C*}.



Viterbi Algorithm

Animation :

HOMEWORK INSTRUCTIONS

about Implementing SVM^{struct}

SVM^{struct}

SVM^{struct} is a Support Vector Machine (SVM) algorithm for predicting multivariate or structured outputs.

In this homework, you have to implement a SVM^{struct} model. We will give you an API (Application Programming Interface) of SVMstruct so you merely need to provide the code for the following :

(Turn to next page)

SVM$^{\text{struct}}$

1. A function for computing the feature vector Psi.
2. A function for computing the argmax over the (kernelized) linear discriminant function.
3. A function for computing the argmax over the loss-augmented (kernelized) linear discriminant function.
4. A loss function.
5. Some functions for I/O.

SVM^{struct}

Instructions :

1. Download the source code (written in C):
   http://download.joachims.org/svm_struct/current/svm_struct.tar.gz
2. If you are not so eager on C programming, you might want to try the Python and Matlab version API. (See Appendix) However, we only show the hints of C version API.
3. Decompress, and then compile it using the command "***make***". It should produce two executable programs ***svm_empty_learn*** and ***svm_empty_classify***.
4. To complete the functions mentioned in the previous page and make the SVM^{struct} executable, you only need to edit the following files :
   a. svm_struct_api_types.h
   b. svm_struct_api.c

svm_struct_api_types.h

Two structs for you to define :

1. pattern : defines the x-part (observation) of a training example
2. label : defines the y-part (label answer) of a training example

$$\Psi(\mathbf{x}, \mathbf{y})$$

svm_struct_api.c

1. **read_struct_examples** : input for training and classifying
2. **init_struct_model** : define the size for feature vector and weight vector
3. **find_most_violated_constraint_slackrescaling** & **find_most_violated_constraint_marginrescaling** : two different types of function for finding most violated constraint
4. **psi** : feature vector
5. **loss** : define your own loss function
6. **classify_struct_example** : for classification
7. **write_struct_model** : write the weight vector to file
8. **read_struct_model** : read the weight vector from file
9. **write_label** : write the prediction result to file
10. (Optional) **free_pattern** & **free_label** : free the memory

Psi

1. Read the instructions in Psi function.
2. The feature vector is returned as a list of SVECTOR.
3. Initialize feature values using struct WORD with the size [feature_vector.size()+1].
   a. words.wnum : feature number, starts from 1
   b. words.weight : feature value
   c. Set the last feature number to 0 as a terminator.
4. After generating all the feature values, call ***create_svector*** to add feature values into SVECTOR.
5. You can check ***svm_common.c*** and ***svm_common.h*** in folder svm_light to make sure everything.

*** If you want to load weight vector in functions, you also need to start from 1.

Trimming

From frame sequence to phone sequence :

frame sequence (prediction result from SVM^{struct})

↓

phone sequence (your final output)

---

frame sequence : {***<u>a a a a a</u> <u>b b</u> <u>c c c</u> <u>d</u>***}

↓

phone sequence : {***a*** ***b*** ***c*** ***d***}

You need to report the result in <u>phone sequence</u>.

Trimming

If silence occurs in the frame sequence :

Eliminate <*sil*> if it occurs at the begin or end of sequences.

While it occurs in the middle of sequences, just regard it as other normal phones.

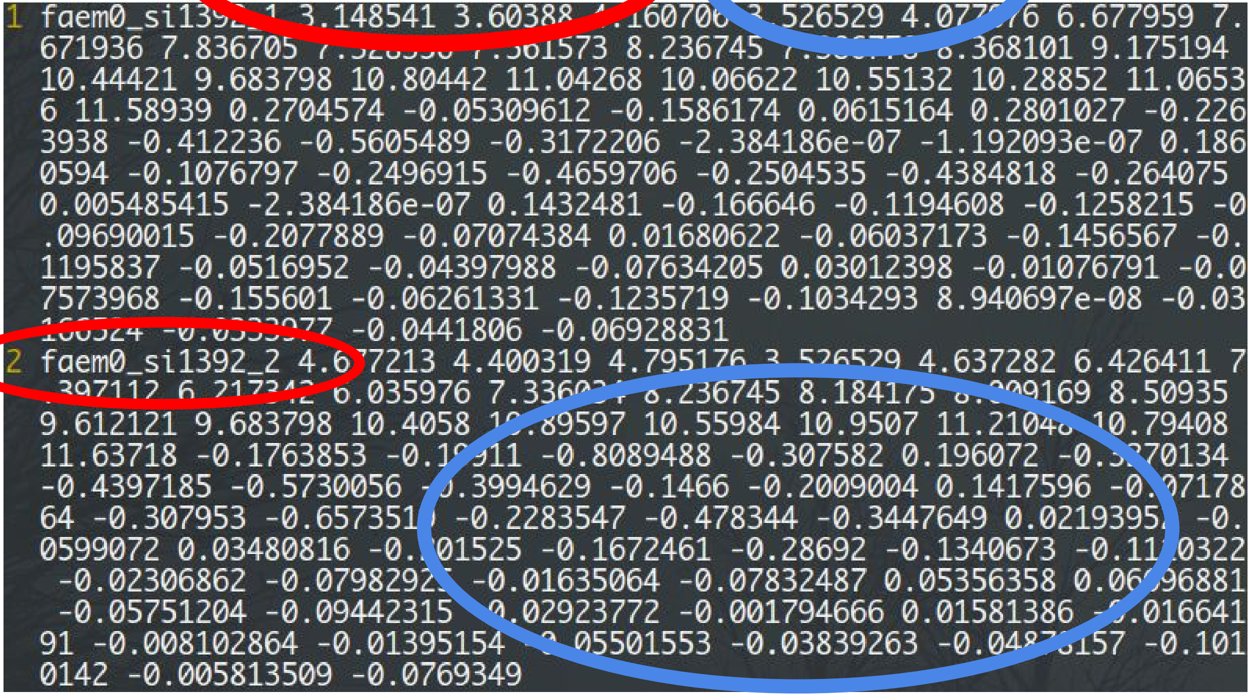
$$\{\underline{<sil>\ <sil>}\ \underline{a\ a\ a\ a}\ \underline{<sil>}\ \underline{c\ c\ c}\ \underline{<sil>}\}$$

$$\downarrow$$

$$\{a\ <sil>\ c\}$$

Data Format

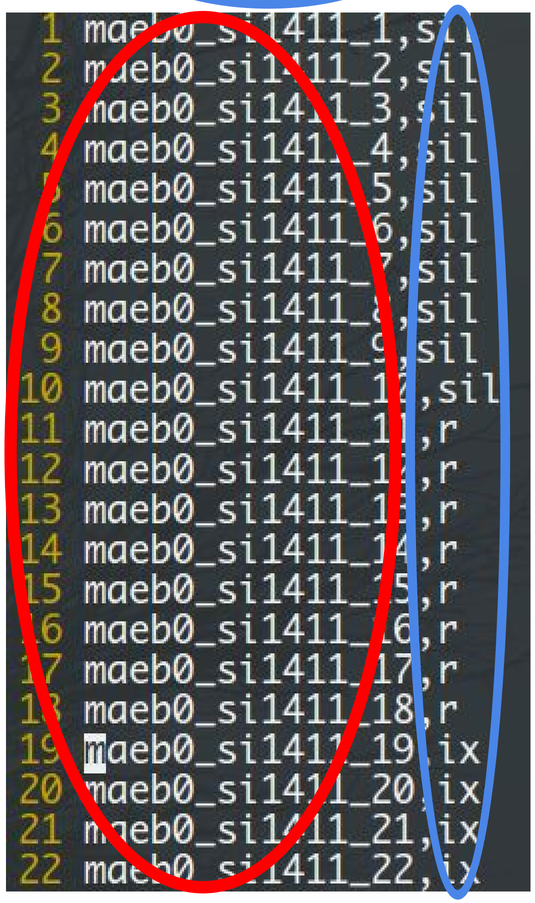
- Same as in HW 1.
- WAV file: Speak-Sentence ID + .wav
  - Check by your ear(s)
- ARK file: Instance ID + features



```
faem0_si1392_1 3.148541 3.60388 4.160706 3.526529 4.077[illegible]76 6.677959 7.
671936 7.836705 7.[illegible] 7.361573 8.236745 7.[illegible] 8.368101 9.175194
10.44421 9.683798 10.80442 11.04268 10.06622 10.55132 10.28852 11.0653
6 11.58939 0.2704574 -0.05309612 -0.1586174 0.0615164 0.2801027 -0.226
3938 -0.412236 -0.5605489 -0.3172206 -2.384186e-07 -1.192093e-07 0.186
0594 -0.1076797 -0.2496915 -0.4659706 -0.2504535 -0.4384818 -0.264075
0.005485415 -2.384186e-07 0.1432481 -0.166646 -0.1194608 -0.1258215 -0
.09690015 -0.2077889 -0.07074384 0.01680622 -0.06037173 -0.1456567 -0.
1195837 -0.0516952 -0.04397988 -0.07634205 0.03012398 -0.01076791 -0.0
7573968 -0.155601 -0.06261331 -0.1235719 -0.1034293 8.940697e-08 -0.03
160324 -0.0[illegible]77 -0.0441806 -0.06928831
faem0_si1392_2 4.677213 4.400319 4.795176 3.526529 4.637282 6.426411 7
.397112 6.217342 6.035976 7.33602[illegible] 8.236745 8.184175 8.[illegible]09169 8.50935
9.612121 9.683798 10.4058 10.89597 10.55984 10.9507 11.21048 10.79408
11.63718 -0.1763853 -0.1[illegible]911 -0.8089488 -0.307582 0.196072 -0.3[illegible]70134
-0.4397185 -0.5730056 -0.3994629 -0.1466 -0.2009004 0.1417596 -0.07178
64 -0.307953 -0.6573510 -0.2283547 -0.478344 -0.3447649 0.0219395[illegible] -0.
0599072 0.03480816 -0.[illegible]01525 -0.1672461 -0.28692 -0.1340673 -0.11[illegible]0322
 -0.02306862 -0.0798292[illegible] -0.01635064 -0.07832487 0.05356358 0.06[illegible]96881
 -0.05751204 -0.09442315 [illegible].02923772 -0.001794666 0.01581386 -0.016641
91 -0.008102864 -0.01395154 -0.05501553 -0.03839263 -0.048[illegible]8157 -0.101
0142 -0.005813509 -0.0769349
```

Data Format

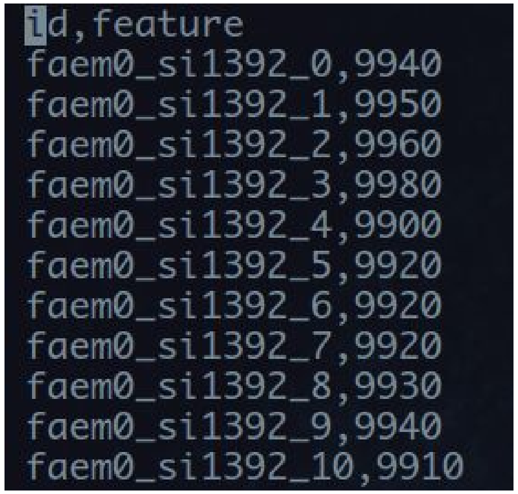
- LAB file: Instance ID + comma + label



```
maeb0_si1411_1,sil
maeb0_si1411_2,sil
maeb0_si1411_3,sil
maeb0_si1411_4,sil
maeb0_si1411_5,sil
maeb0_si1411_6,sil
maeb0_si1411_7,sil
maeb0_si1411_8,sil
maeb0_si1411_9,sil
maeb0_si1411_10,sil
maeb0_si1411_11,r
maeb0_si1411_12,r
maeb0_si1411_13,r
maeb0_si1411_14,r
maeb0_si1411_15,r
maeb0_si1411_16,r
maeb0_si1411_17,r
maeb0_si1411_18,r
maeb0_si1411_19,ix
maeb0_si1411_20,ix
maeb0_si1411_21,ix
maeb0_si1411_22,ix
```

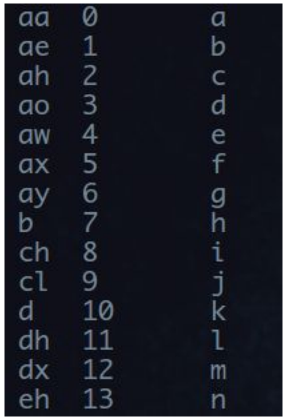
Data Format (a) (Upload)

- **CSV file**: Your prediction when submission.
- With header row: “id,feature”
- Only fbank feature is allowed
- You only need to upload the utterance “faem0_si1392”, and print every feature values in a line to make sure you know how to generate the feature vector
- Feature ID + comma + feature values



```
id,feature
faem0_si1392_0,9940
faem0_si1392_1,9950
faem0_si1392_2,9960
faem0_si1392_3,9980
faem0_si1392_4,9900
faem0_si1392_5,9920
faem0_si1392_6,9920
faem0_si1392_7,9920
faem0_si1392_8,9930
faem0_si1392_9,9940
faem0_si1392_10,9910
```

Data Format (a) (Upload)



```
aa  0       a
ae  1       b
ah  2       c
ao  3       d
aw  4       e
ax  5       f
ay  6       g
b   7       h
ch  8       i
cl  9       j
d   10      k
dh  11      l
dx  12      m
eh  13      n
```

- Feature vector format
- 69*48 dimensions for observation values
- 48*48 dimensions for transition values
- **observation values:** for fbank vector of phoneme “aa”, you should put it to the feature vector at dim#0~dim#68
- **transition values:**
  a. transition from “aa” to “aa”, then add 1 at dim#69*48
  b. transition from “aa” to “ae”, then add 1 at dim#69*48+1

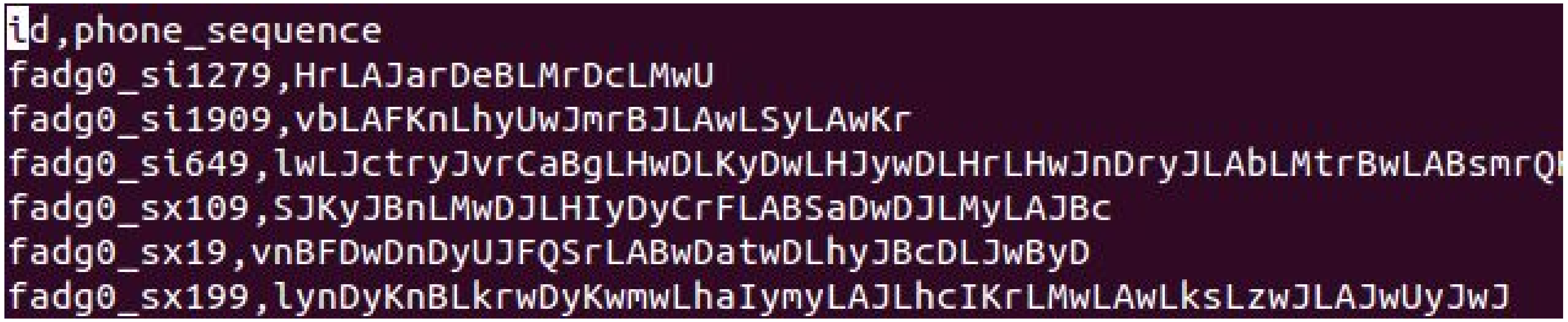
Data Format (a) (Upload)

- Transition value example (Transition table)

	aa	ae	ah	ao				
aa	69*48+0	69*48+1	69*48+2	69*48+3	...	...	...	...
ae	69*48+48	69*48+49	...	...	...	...	...	...
ah	69*48+96	...						
	...							
	...							
	...							
	...							
	...							

Data Format (b) (Upload)

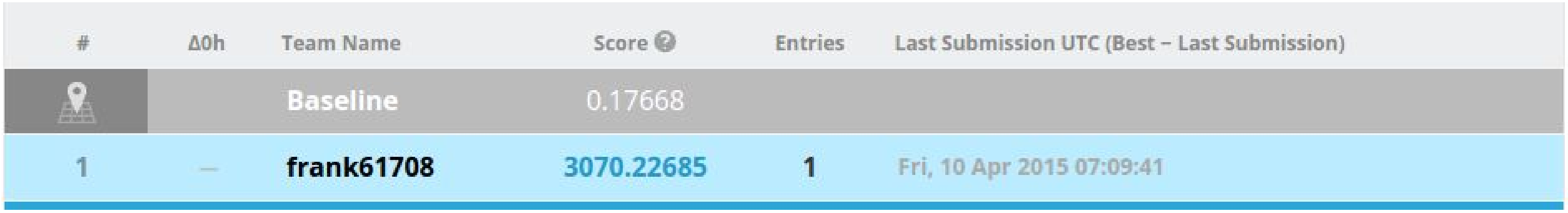
- **CSV file**: Your prediction when submission.
- **Must mapped to 39-phonemes(in english characters without space)**
- **With header row: “id,phone_sequence”**
- **utterance ID + comma + 39-phonemes sequence**



```
id,phone_sequence
fadg0_si1279,HrLAJarDeBLMrDcLMwU
fadg0_si1909,vbLAFKnLhyUwJmrBJLAwLSyLAwKr
fadg0_si649,lwLJctryJvrCaBgLHwDLKyDwLHJywDLHrLHwJnDryJLAbLMtrBwLABsmrQ
fadg0_sx109,SJKyJBnLMwDJLHIyDyCrFLABSaDwDJLMyLAJBc
fadg0_sx19,vnBFDwDnDyUJFQSrLABwDatwDLhyJBcDLJwByD
fadg0_sx199,lynDyKnBLkrwDyKwmwLhaIymyLAJLhcIKrLMwLAwLksLzwJLAJwUyJwJ
```

Submission 1(Kaggle)

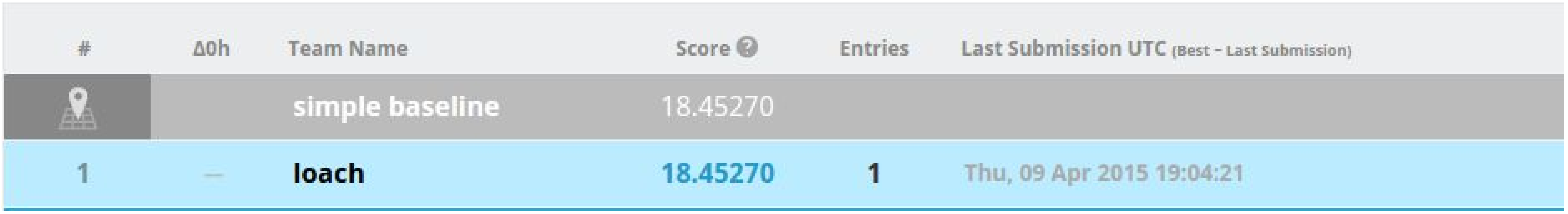
- Feature vector from original acoustic feature
- almost 100% correctness
- https://inclass.kaggle.com/c/mlds-hw2-a3



#	Δ0h	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)
		Baseline	0.17668		
1	—	frank61708	3070.22685	1	Fri, 10 Apr 2015 07:09:41

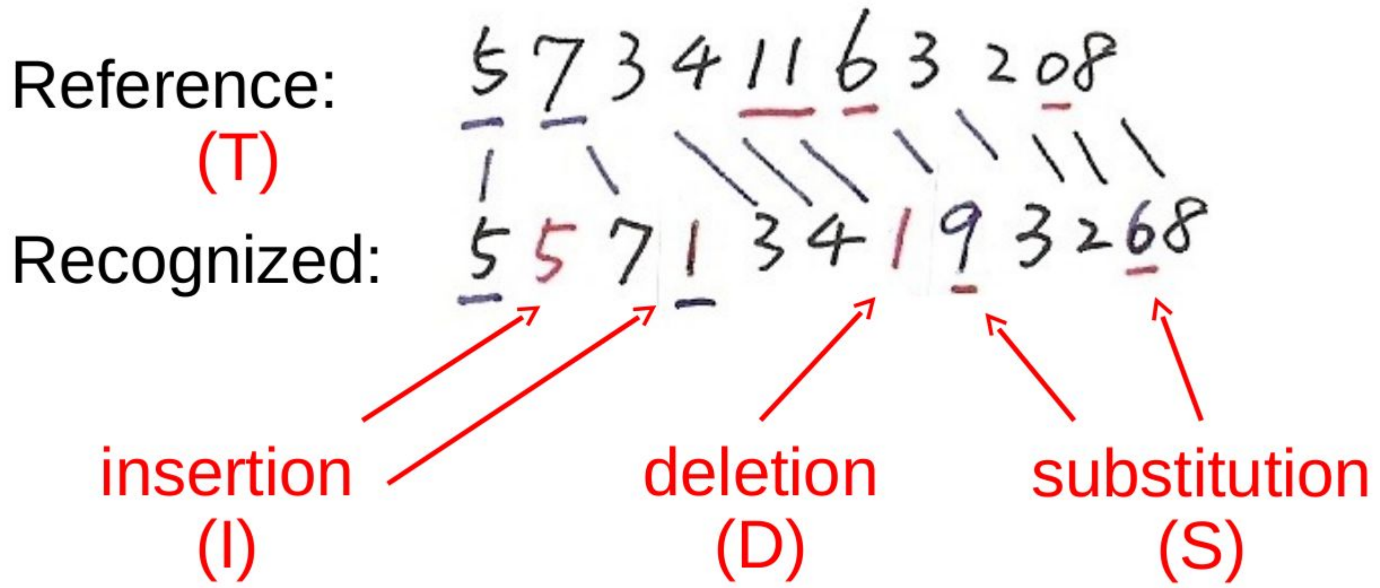
Submission 2(Kaggle)

- Your predict.csv.
- Twice a day.
- 50 % public score and 50% private score
- Evaluate by Edit distance.
- https://inclass.kaggle.com/c/mlds-hw2-b



#	Δ0h	Team Name	Score	Entries	Last Submission UTC (Best − Last Submission)
		simple baseline	18.45270		
1	—	loach	18.45270	1	Thu, 09 Apr 2015 19:04:21

Phone Error Rate



$$\frac{T-D-S-I}{T} \times 100\% = \text{Accuracy}$$

D+S+I = Edit distance

Submission (Ceiba)

- **Your source code and documentation**
  - Usage and environment Setting
  - Package dependency
  - Only **SVM**struct API is allowed
- **Report**
- Upload 1 copy per group.

New Rules

https://docs.google.com/presentation/d/1Crz4_rX5XaqtXEsWTfbTaX3QVpCQzuqwSjt2P21sRZQ/edit

Report

- Group Information 10%
  - Group Name, Student Name, Student ID
  - Member Contribution
- What have you done? 40%
  - Variations of acoustic feature
  - Algorithm to find negative examples.
  - Other algorithms to boost performance.
  - Implementation tips and obstacles
  - Bug(s) and how to solve it(them).

New Rules

https://docs.google.com/presentation/d/1Crz4_rX5XaqtXEsWTfbTaX3QVpCQzuqwSjt2P21sRZQ/edit

Report

- Experiment Setting and Results 50%
  - How you design your experiments
  - Compare different acoustic features and finding negative examples algorithms
- No more than 4 pages with font 12, A4 size.

New Rules

https://docs.google.com/presentation/d/1Crz4_rX5XaqtXEsWTfbTaX3QVpCQzuqwSjt2P21sRZQ/edit

Grading Policy

- Accuracy 60%
  - Submission 1 (30%)
  - Submission 2 (30%)
    - Simple Baseline in Kaggle (released day 7)
    - Once achieve the baseline, you can get the full credit.
    - 1 % less absolute accuracy = 1% absolute credit loss
- Report 40%
- Implementation 20%.
- Bonus
  - First Place 15%
  - First Runner-up 10%
  - Second Runner-up 5%

New Rules

https://docs.google.com/presentation/d/1Crz4_rX5XaqtXEsWTfbTaX3QVpCQzuqwSjt2P21sRZQ/edit

Penalty

- Latency
  - Half-life = 24 hrs
  - The excess time will round up to hours.
- Usage on other toolkit excluding SVM^{struct}.
  - You shall get 0 credit in both implementation and accuracy parts.
  - Please ask TAs whether you could use the package or not.

New Rules

https://docs.google.com/presentation/d/1Crz4_rX5XaqtXEsWTfbTaX3QVpCQzuqwSjt2P21sRZQ/edit

Appendix

- **SVM^{struct} for Python :**

  http://tfinley.net/software/svmpython2/

- **SVM^{struct} for Matlab :**

  http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html

Reference

- http://www.yisongyue.com/talks/svm_struct_intro.ppt
- http://cpmarkchang.logdown.com/posts/192522-natural-language-processing-viterbi-algorithm