Gaussian Mixture Models in Speech

CSC401/2511 2026 A3 Tutorial 01.
Presenter: Jiankun Wei, Slides: Ken Shi

UNIVERSITY OF TORONTO

Agenda

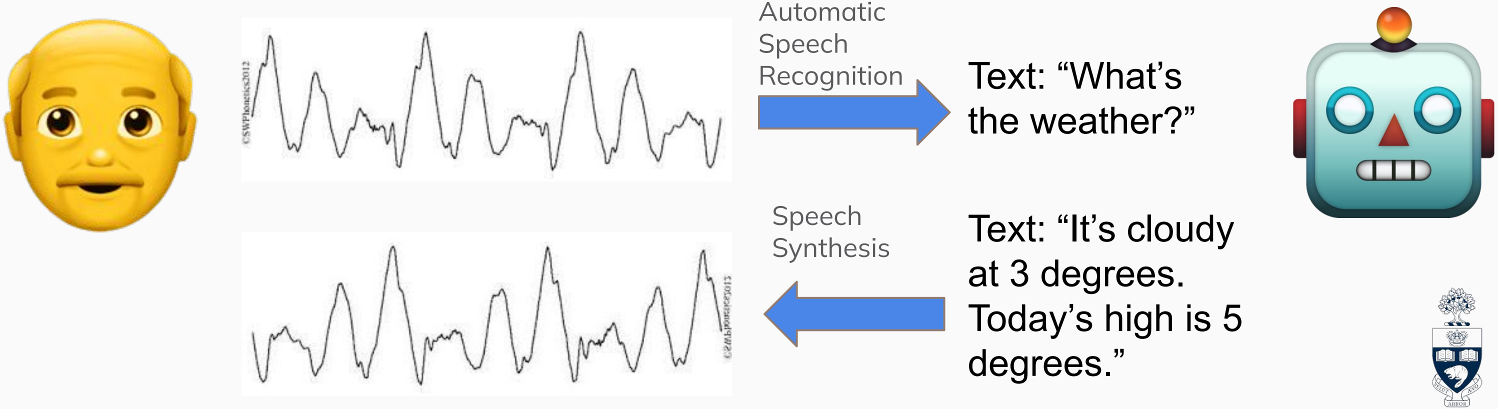
- Assignment 3 Logistics
- A brief summary of Speech stuff you need to know for this assignment
    - Speech technology
    - Speech signal features, MFCC
    - Acoustic phonetics
- Gaussian Mixture Model and GRU for lie detection
- Dynamic Programming in Speech (at a later date)

Assignment Breakdown

- Part 1: Sequence Classifications [35/70]
  - Implement GMMs for speaker identifications [30/35]
  - Implement and train a GRU for lie detection [5/35]
- Part 2: Dynamic Programming in Speech [35/70]
  - Implement Levenshtein Word Error Rate for ASR Evaluation [15/35]
  - Implement Dynamic Time Warping for Speaker Verification [20/35]

Applications of Speech Technology

Speech! Very useful stuff!



Automatic Speech Recognition

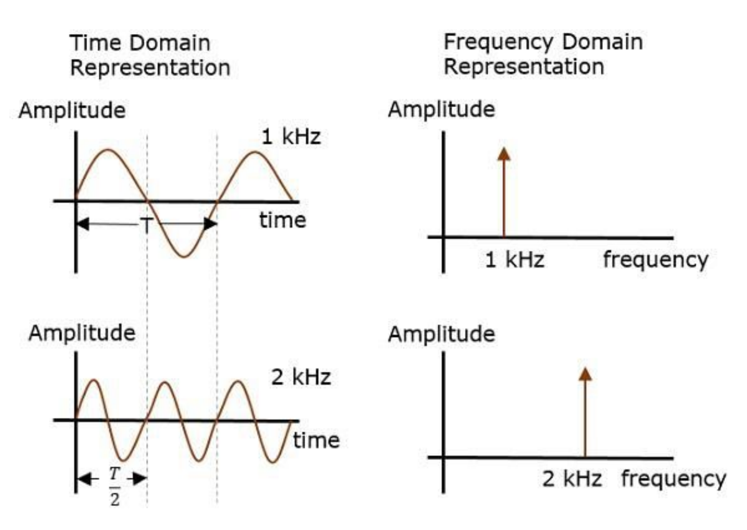
Text: "What's the weather?"

Speech Synthesis

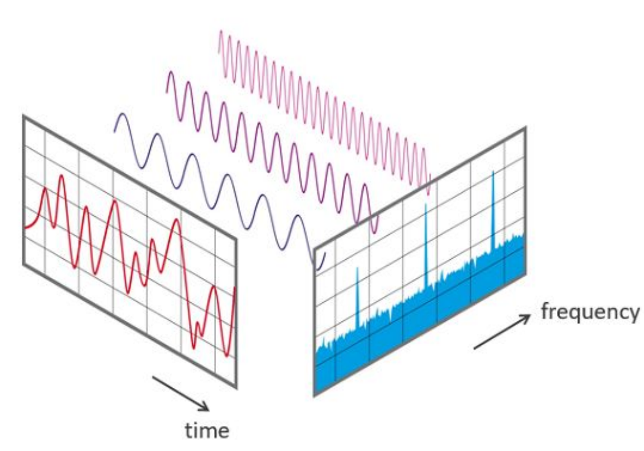
Text: "It's cloudy at 3 degrees. Today's high is 5 degrees."

Automatic Speech Recognition (ASR)

- Input: Speech; Output: Text
- Feature Extraction (Formants, MFCC, etc)



How? Fourier Transform (FT) !



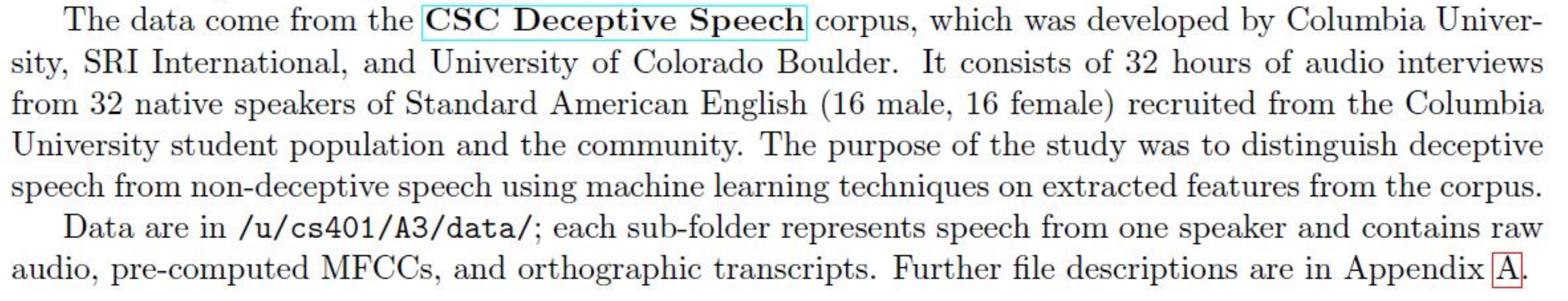
Tasks in A3

- Speaker Identification (via GMM)
  - Speech data -> Speaker Information
- Lie Detection (via GRU)
  - Speech data -> Truth or Lie
- ASR Evaluation (via WER)
  - ASR result -> Evaluation of ASR system
- Speaker Verification (via DTW)
  - Speech data (pairs) -> Same speaker or not

UNIVERSITY OF TORONTO

Dataset: CSC Deceptive Speech

Descriptions can be found in the assignment handout:



The data come from the **CSC Deceptive Speech** corpus, which was developed by Columbia University, SRI International, and University of Colorado Boulder. It consists of 32 hours of audio interviews from 32 native speakers of Standard American English (16 male, 16 female) recruited from the Columbia University student population and the community. The purpose of the study was to distinguish deceptive speech from non-deceptive speech using machine learning techniques on extracted features from the corpus.

Data are in `/u/cs401/A3/data/`; each sub-folder represents speech from one speaker and contains raw audio, pre-computed MFCCs, and orthographic transcripts. Further file descriptions are in Appendix A.

Speaker Data

- 32 speakers, labeled as “S-{number}{group}”
- Each speaker has up to 12 utterances
- Each utterances are represented by 3 files:
    - .wav file: raw audio
    - .mfcc.npy: MFCCs in numpy format
    - .txt: transcripts
- For the purpose of this assignment, you are also given 2 transcripts from two different APIs: Kaldi and Google

Speaker Recognition

with GMMs

Speaker Recognition: Task Description

- The data is randomly split into training and testing utterances. We don’t know which speaker produced which test utterance.
- Every speaker occupies a characteristic part of the acoustic space.

↓

Learn a probability distribution for each speaker that describes their acoustic behaviour!

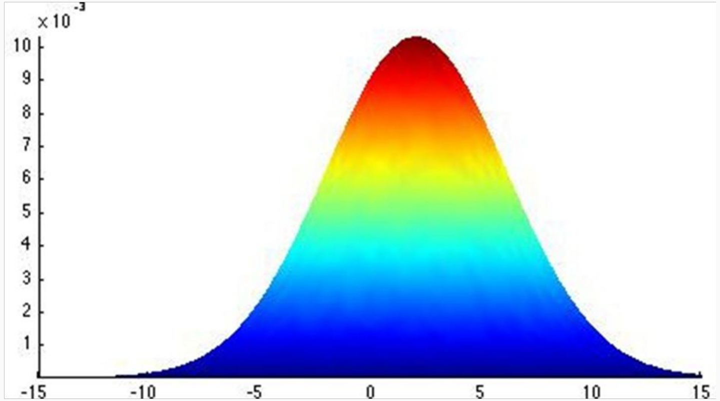
↓

Gaussian Mixture Model (GMM) Distribution!

GMM: 1D Gaussians

$$p(x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$N(\mu, \sigma)$$



$$\mu = E(x) = \int xp(x)dx$$

$$\sigma^2 = E((x-\mu)^2) = \int (x-\mu)^2 p(x)dx$$

$$\theta = \langle \mu, \sigma \rangle$$

UNIVERSITY OF TORONTO

GMM: Maximum Likelihood Estimation (MLE)

- Given: $X = \{x_1, x_2, \ldots, x_n\} \quad \theta = \langle \mu, \sigma \rangle$

- Want: Find optimal set θ that maximizes the likelihood of the data:

$$L(X,\theta) = p(X \mid \theta) = p(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta)$$

- We will do so by making the derivative of this likelihood function 0:

$$\frac{\partial}{\partial \theta} L(X,\theta) = 0$$

GMM: MLE cont'd

- Estimate optimal average $\hat{\mu}$:

$$L(X, \mu) = p(X \mid \mu) = \prod_{i=1}^{n} p(x_i \mid \mu) = \prod_{i=1}^{n} \frac{\exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$\log L(X, \mu) = -\frac{\sum_i (x_i - \mu)^2}{2\sigma^2} - n \log \sqrt{2\pi}\sigma$$

$$\frac{\partial}{\partial \mu} \log L(X, \mu) = \frac{\sum_i (x_i - \mu)}{\sigma^2} = 0$$

$$\boxed{\hat{\mu} = \frac{\sum_i x_i}{n}}$$

TL;DR: Similarly, $\boxed{\hat{\sigma}^2 = \frac{\sum_i (x_i - \hat{\mu})^2}{n}}$

UNIVERSITY OF TORONTO

GMM: Multidimensional Gaussians

How about Gaussian of higher dimensions? $\vec{x} = \langle x[1], x[2], \ldots, x[d] \rangle$

$$p(x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$p(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{d/2}|\Sigma|^{1/2}}$$

$$\mu = E(x) = \int x p(x) dx$$

→

$$\vec{\mu} = E(\vec{x}) = \langle \mu[1], \mu[2], \ldots, \mu[d] \rangle$$

$$\sigma^2 = E((x-\mu)^2) = \int (x-\mu)^2 p(x) dx$$

$$\Sigma = E((\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T)$$

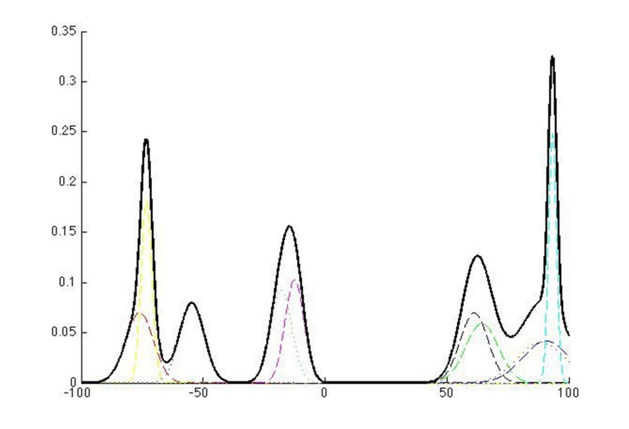
with $\Sigma[i,j] = E(x[i]x[j]) - \mu[i]\mu[j]$

UNIVERSITY OF TORONTO

GMM: Gaussian Mixtures

- Weighted Linear Combination of M component Gaussians: $\langle \Gamma_1, \ldots, \Gamma_M \rangle$

$$p(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{d/2} |\Sigma|^{1/2}} \rightarrow p(\vec{x}) = \sum_{j=1}^{M} p(\Gamma_j) p(\vec{x} \mid \Gamma_j)$$



Let's denote: $\omega_m = p(\Gamma_m),\ b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$

GMM: MLE for GMM

Follow the convention from last page: $\omega_m = p(\Gamma_m),\ b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$

$$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t}),\ \Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle,\ m = 1, \ldots, M$$

$$\boxed{b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d} \sigma_m^2[i]\right)^{1/2}}}$$

, Estimate $\hat{\Theta}$ for $\nabla_\Theta \log L(X, \Theta) = 0$ where:

$$\log L(X, \Theta) = \sum_{t=1}^{N} \log p_\Theta(\vec{x_t}) = \sum_{t=1}^{N} \log\left(\sum_{m=1}^{M} \omega_m b_m(\vec{x_t})\right)$$

GMM: MLE for GMM cont'd

We know that $$\frac{\partial \log L(X, \Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{1}{p_\Theta(\vec{x_t})} \left[\frac{\partial}{\partial \mu_m[n]} \omega_m b_m(\vec{x_t}) \right]$$

$$\frac{\partial}{\partial \mu_m[n]} b_m(\vec{x_t}) = b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]}$$

$$\frac{\partial \log L(X, \Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]} = 0$$

$$\boxed{\hat{\mu_m}[n] = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta) x_t[n]}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}}$$

$$\boxed{\begin{gathered} b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m) \\ p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t}) \end{gathered}}$$

GMM: Recipe for MLE Training:

- For each speaker:
  - **<u>1. Initialize</u>**: Guess $\Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle, \ m = 1, \ldots, M$ with M random vectors in the data, or by performing M-means clustering.
  - **<u>2. Compute likelihood</u>**: Compute $b_m(\vec{x_t})$ and $P(\Gamma_m \mid \vec{x_t}, \Theta)$
  - **<u>3. Update parameters</u>**:

$$\boxed{\hat{\omega_m} = \frac{1}{T} \sum_{t=1}^{T} p(\Gamma_m \mid \vec{x_t}, \Theta)} \quad \boxed{\hat{\vec{\mu_m}} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta) \vec{x_t}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}} \quad \boxed{\hat{\vec{\sigma}}_m^{\,2} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta) \vec{x_t}^{\,2}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)} - \hat{\vec{\mu}}_m^{\,2}}$$

$$\log p(X \mid \hat{\Theta}_{i+1}) - \log p(X \mid \hat{\Theta}_i) < \epsilon$$

  - Repeat 2&3 until converges

GMM: Takeaways!!

- Probability of observing x_t in the mth Gaussian: $b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$

$$b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d}\sigma_m^2[i]\right)^{1/2}}$$

- Prior probability of the mth Gaussian: $\omega_m = p(\Gamma_m)$

- Probability of the mth Gaussian, given x_t : $p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t})$

- Probability of x_t in the GMM: $$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t})$$

GMM: Initialization

We want to estimate the parameter set:

$$\Theta = \langle \omega_1, \mu_1, \Sigma_1, \omega_2, \mu_2, \Sigma_2, \ldots, \omega_M, \mu_M, \Sigma_M \rangle$$

μ_m : Random vector from data

Σ_m : Random diagonal matrix or identity matrix

ω_m : Random value with following constraints:

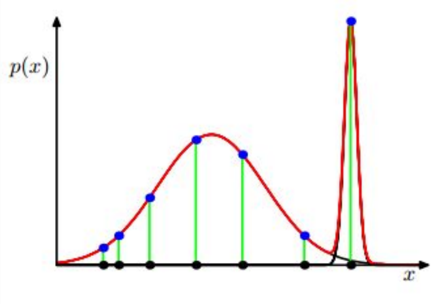
$$0 \leq \omega_m \leq 1$$

$$\sum_m \omega_m = 1$$

… or simply use 1 / m

GMM: Overfitting

When one of your GM is centred at a data point…



Consider $b_m(\vec{x_t}) = \dfrac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d}\sigma_m^2[i]\right)^{1/2}}$

$$\log L(X,\Theta) = \sum_{t=1}^{N}\log p_{\Theta}(\vec{x_t}) = \sum_{t=1}^{N}\log\left(\sum_{m=1}^{M}\omega_m b_m(\vec{x_t})\right)$$

$$\mu_j = x_n \quad \mathcal{N}(x_n|x_n,\ \sigma_j^2\mathbf{I}) = \frac{1}{(2\pi)^{d/2}}\frac{1}{\sigma_j^d} \qquad \sigma_j \longrightarrow 0$$

Solution: use a smaller M so that it's impossible to have spare clusters!

GMM: Practical Tips

- Assumption: Diagonal Covariance Matrices
- For Numerical Stability: Compute likelihoods in the log domain:

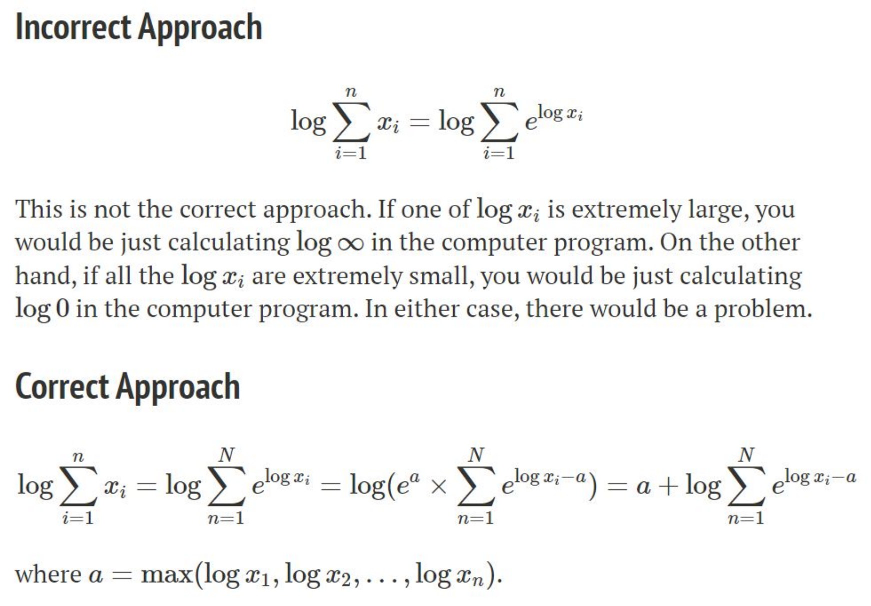
$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d} \frac{(\vec{x_t}[n] - \vec{\mu_m}[n])^2}{2\vec{\sigma_m}^2[n]} - \frac{d}{2}\log 2\pi - \frac{1}{2}\log \prod_{n=1}^{d} \vec{\sigma_m}^2[n]$$

- Pre-compute some terms for efficiency!

$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d} \left(\frac{1}{2}\vec{x_t}[n]^2\vec{\sigma_m}^{-2}[n] - \vec{\mu_m}[n]\vec{x_t}[n]\vec{\sigma_m}^{-2}[n] \right) - \left(\sum_{n=1}^{d} \frac{\vec{\mu_m}[n]^2}{2\vec{\sigma_m}^2[n]} + \frac{d}{2}\log 2\pi + \frac{1}{2}\log \prod_{n=1}^{d} \vec{\sigma_m}^2[n] \right)$$

GMM: Practical Tips cont'd

How do we get the $b_m(\vec{x_t})$ in $p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t})$ from $\log b_m(\vec{x_t})$?



Incorrect Approach

$$\log \sum_{i=1}^{n} x_i = \log \sum_{i=1}^{n} e^{\log x_i}$$

This is not the correct approach. If one of $\log x_i$ is extremely large, you would be just calculating $\log \infty$ in the computer program. On the other hand, if all the $\log x_i$ are extremely small, you would be just calculating $\log 0$ in the computer program. In either case, there would be a problem.

Correct Approach

$$\log \sum_{i=1}^{n} x_i = \log \sum_{n=1}^{N} e^{\log x_i} = \log(e^a \times \sum_{n=1}^{N} e^{\log x_i - a}) = a + \log \sum_{n=1}^{N} e^{\log x_i - a}$$

where $a = \max(\log x_1, \log x_2, \ldots, \log x_n)$.

A good module that does this:

- scipy.special.logsumexp

Lie Detection

with GRU

UNIVERSITY OF TORONTO

Truth/Lie Detection: ToDo List

- Finish the __ init __ method of the class LieDetector by filling in the code to create a unidirectional GRU with one hidden layer
- Also, initialize a linear layer to project the GRU's output to prediction space. What should the number of output features be?
- Following the instructions from the handout, run experiments by varying the size of the hidden layer and record the performance of the model. Comment on these results as asked in the handout.
- Note : the model most likely won't perform very well, that's okay ! Truth/lie detection is a hard task, and we have limited data.