Gaussian Mixture Models (GMM)
- Probabilistic model for clustering data
- Assumes the distribution of observations follows a mixture of multidimensional Gaussian distributions
- Each Gaussian in the mixture has a mean ($\mu$) and a variance ($\sigma^2$) parameter, as well as a weight ($w$)

Example applications:
- Speech Recognition – speaker classification, acoustic modeling for speech recognition
- Computer Vision – image segmentation, hand writing recognition
- Biology – flow cytometry
- Data mining – topics in documents

GMM Training (EM algorithm)
- Given a set of observations/events: find the maximum likelihood estimates of the set of Gaussian Mixture parameters ($\mu, \sigma, \pi$) and classify observations
- Expectation Maximization (EM) Algorithm
  - E step: Compute probabilities of events given model parameters
  - M step: Compute model parameters given probabilities
    - Weights, mean, covariance matrix
    - Iterate until convergence

Specialization of the covariance matrix computation
- Problem parameters:
  - $N$: number of events, $\sim$10K-100K
  - $D$: event dimension, $\sim$10-40
  - $M$: number of Gaussians (clusters), $\sim$1-128
  - $\Sigma$: symmetric – only compute the lower 2D/3 cells
- Platform parameters (GPU):
  - Number of SMs
  - Number of SIMD vector lanes
  - Size of per-block shared memory
  - Size of global memory

Optimal-performing code variant depends both on the specific platform and the specific problem parameters

Need to develop an automatic selection mechanism that intelligently selects between the code variants based on problem and platform parameters

SEJIT Specializer Framework
- High level goal: automatically transform high-level abstraction of a machine learning algorithm to highly efficient parallel code

General Specializer Setup
- Application code is written in Python
- Specialization is done by:
  - Creating templates for both the host and device (CPU and GPU) code in C and CUDA
  - Filling templates with the correct code variant and associated runtime parameters
- ASP Specializer (Mako, CodePy, PyUBLAS)
  - Takes in the problem and platform parameters
  - Selects appropriate code variant (currently tries all and remembers best-performing one)
  - Pulls in the template for the code variant, parameterizes it and compiles to binary

Code Domains
- Python code handles application
  - Manipulates problem data, determines ML targets
  - C code that runs quickly on the CPU
    - Allocates GPU memory
    - Performs main EM iterative loop
      - Until convergence, call E step(s) and M step(s)
    - Calls variants of mstep covariance(events, GMM_model)
- CUDA code that runs quickly on the GPU
  - Defines GPU kernels and their operation
  - Contains kernel code variants

Data Sharing and Allocation
- To allow trained parameters to be read in Python after training, we pass references to data allocated in Python to C
- C code allocates GPU memory and temporary data structures on the CPU, performs training, and copies the data back
- Allocate new event data on demand – i.e. if we’re training models on the same data in a loop, we do not allocate and copy event data every iteration

Example Application Code
- Agglomerative Hierarchical Clustering
  - Perform GMM training within an outer loop that decreases number of clusters
  - Select best “fitting” GMM – number of clusters that best describes the event data
  - Used in speaker diarization – unsupervised identification of speakers in an audio sample

- Probability Computation
  - Compute the probability of observing an event given the trained model
  - Used in speech recognition to compute the observation probability of an audio sample

Covariance Matrix Computation
- Code Variants

Example of GPU block and thread configuration for different code variants

Future Work
- More intelligent code variant selection mechanism, given platform and problem parameters:
  - Pull from existing database of best-performing code variants
  - Use machine learning to predict the best-performing code variant
- Expand framework to specialize other GMM computations:
  - Probability computation
  - Cluster distance computation functions (BIC, AIC)
- Expand framework to other applications (computer vision, data mining) and architectures (OpenCL, RISC-V)
- Performance improvement of the GMM framework for particular application common use cases to reduce overhead