Language, Dialect, and Speaker Recognition Using Gaussian Mixture Models on the Cell Processor

Nicolas Malyska, Sanjeev Mohindra, Karen Lauro, Douglas Reynolds, and Jeremy Kepner

{nmalyska, smohindra, karen.lauro, reynolds, kepner}@ll.mit.edu

This work is sponsored by the United States Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Outline

- Introduction
- Recognition for speech applications using GMMs
- Parallel implementation of the GMM
- Performance model
- Conclusions and future work

Introduction

Automatic Recognition Systems

- **In this presentation, we will discuss technology that can be applied to different kinds of recognition systems**
  - **Language recognition**
  - **Dialect recognition**
  - **Speaker recognition**

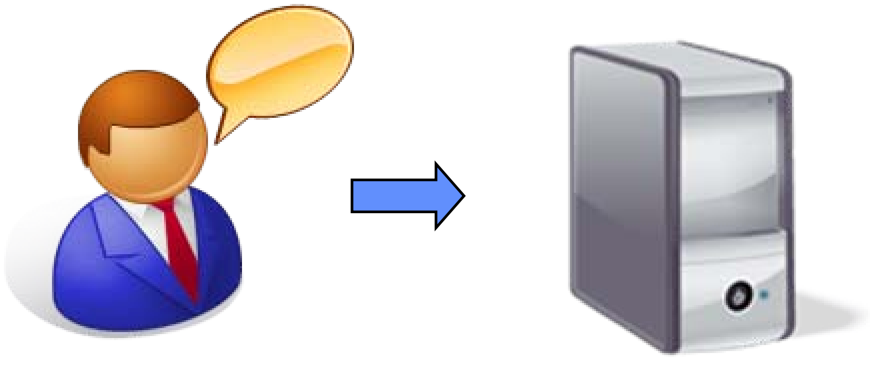
Who is the speaker?

What language are they speaking?

What dialect are they using?

Introduction
The Scale Challenge

- **Speech processing problems are often described as one person interacting with a single computer system and receiving a response**



Introduction
The Scale Challenge

- **Real speech applications, however, often involve data from multiple talkers and use multiple networked multicore machines**
  - **Interactive voice response systems**
  - **Voice portals**
  - **Large corpus evaluations with hundreds of hours of data**

Information
About Speaker,
Dialect, or Language

Introduction
The Computational Challenge

- **Speech-processing algorithms are computationally expensive**
- **Large amounts of data need to be available for these applications**
  - **Must cache required data efficiently so that it is quickly available**
- **Algorithms must be parallelized to maximize throughput**
  - **Conventional approaches focus on parallel solutions over multiple networked computers**
  - **Existing packages not optimized for high-performance-per-watt machines with multiple cores, required in embedded systems with power, thermal, and size constraints**
  - **Want highly-responsive "real-time" systems in many applications, including in embedded systems**

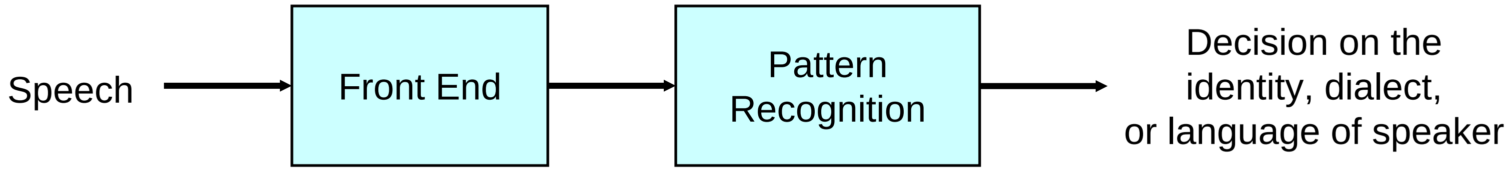
Outline

- Introduction
- Recognition for speech applications using GMMs
- Parallel implementation of the GMM
- Performance model
- Conclusions and future work

Recognition Systems

Summary

- **A modern language, dialect, or speaker recognition system is composed of two main stages**
  - **Front-end processing**
  - **Pattern recognition**

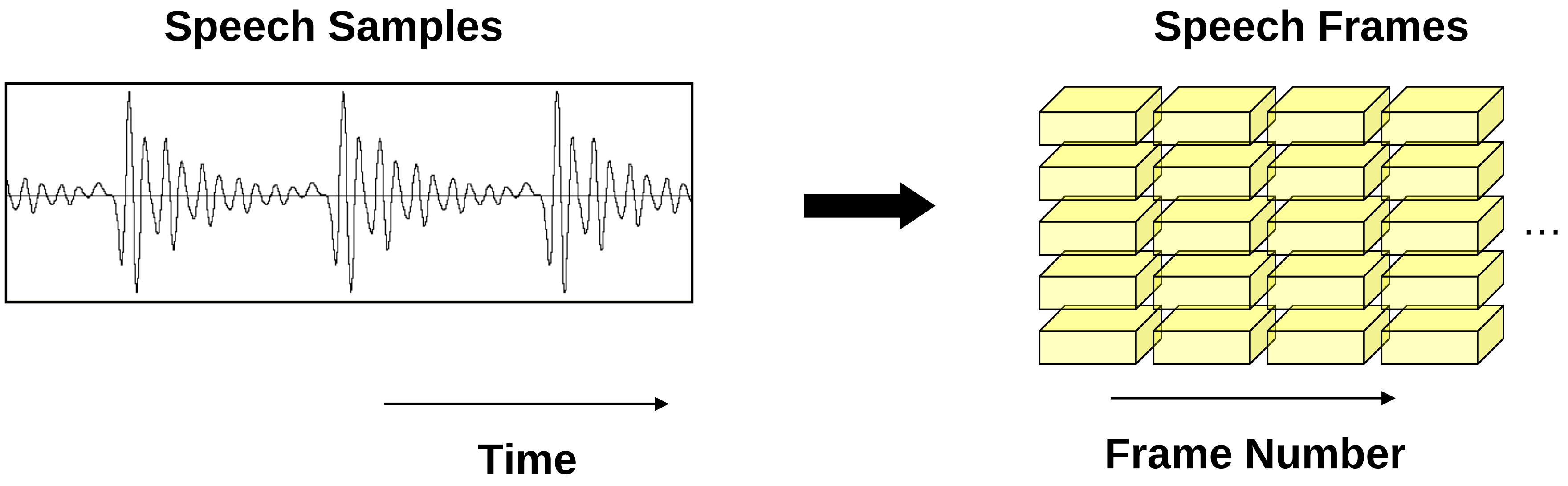


- **We will show how a speech signal is processed by modern recognition systems**
  - **Focus on a recognition technology called Gaussian mixture models**

Recognition Systems

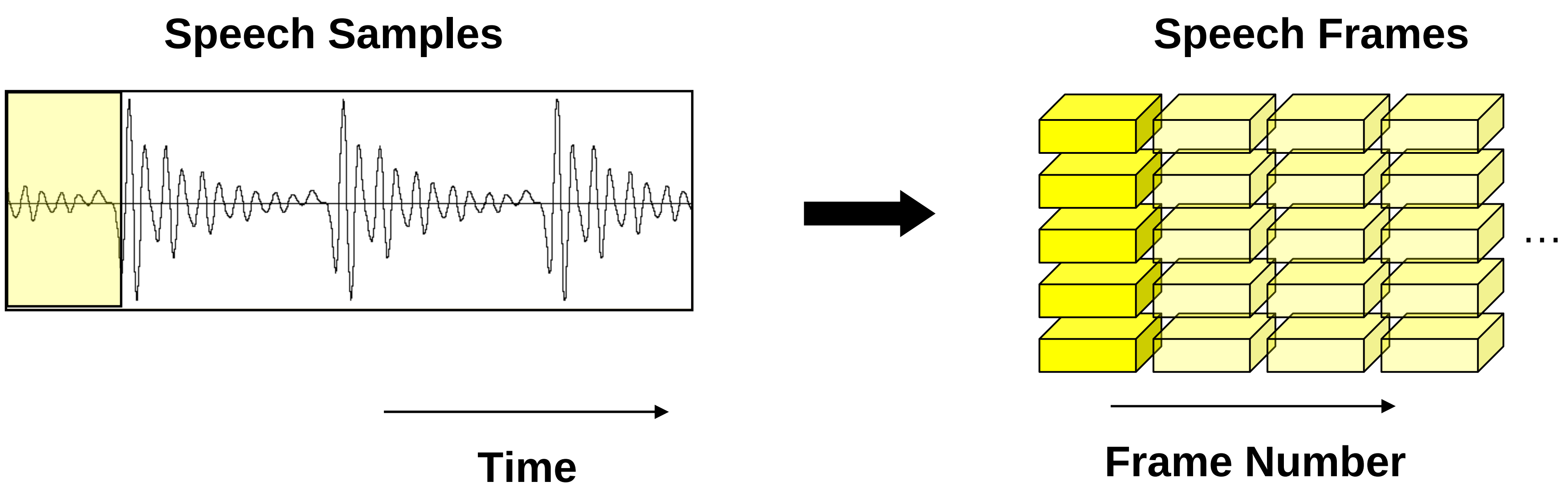
Frame-Based Processing

- **The first step in modern speech systems is to convert incoming speech samples into *frames***
- **A typical frame rate for a speech stream is 100 frames per second**



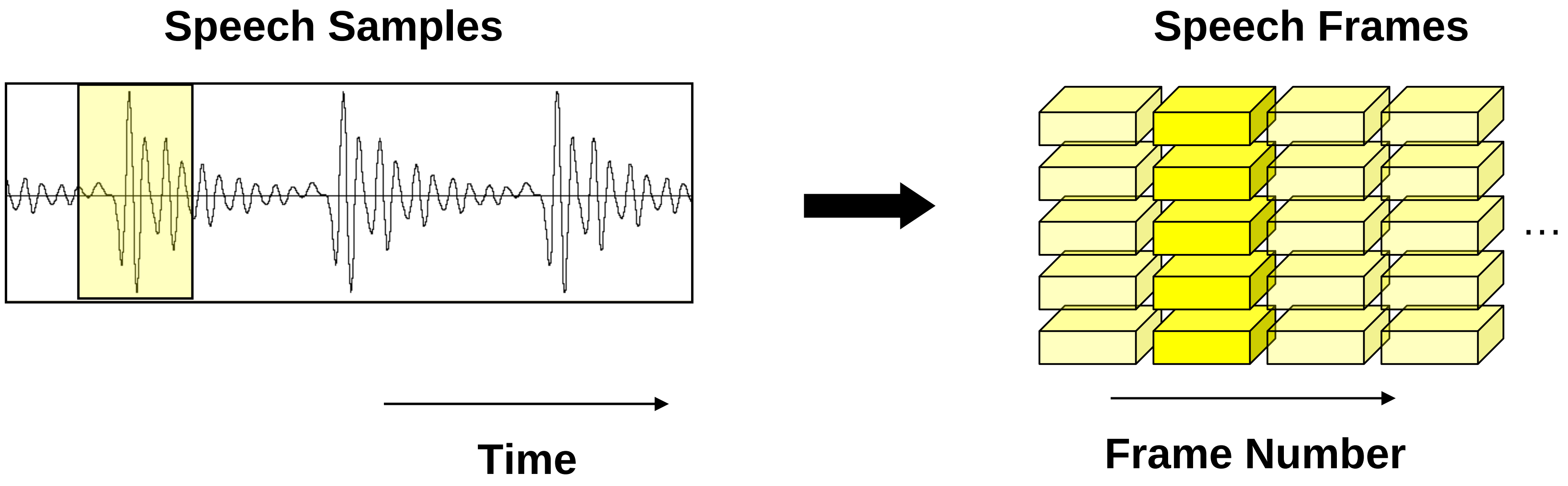
Recognition Systems
Frame-Based Processing

- The first step in modern speech systems is to convert incoming speech samples into *frames*
- A typical frame rate for a speech stream is 100 frames per second



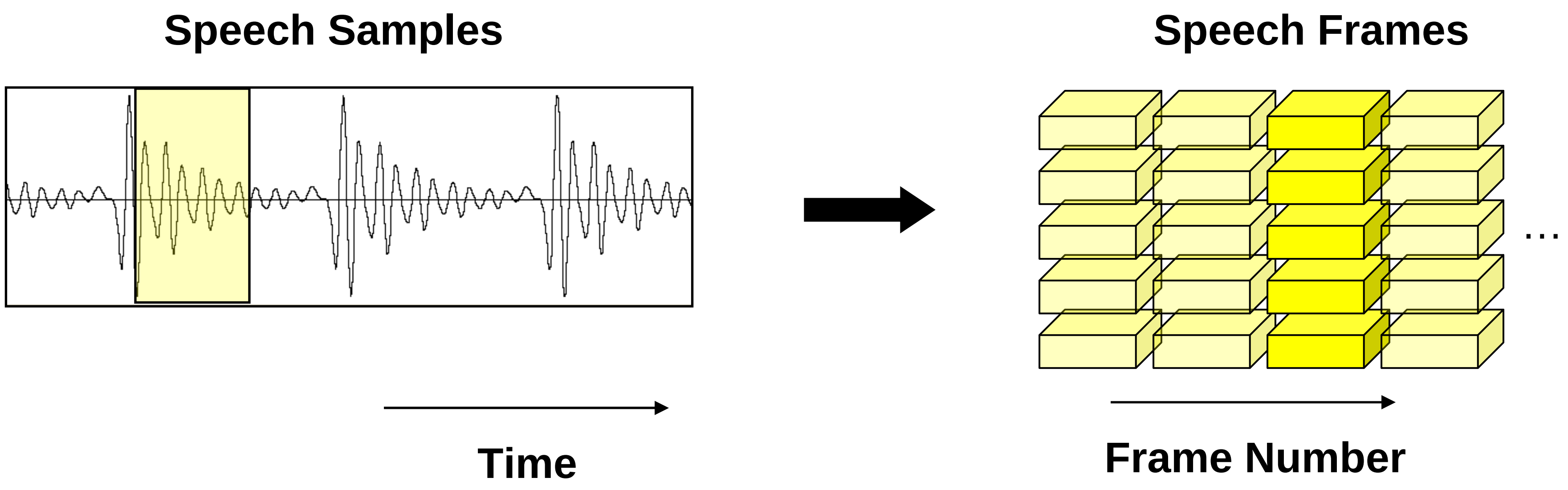
Recognition Systems
Frame-Based Processing

- The first step in modern speech systems is to convert incoming speech samples into *frames*
- A typical frame rate for a speech stream is 100 frames per second



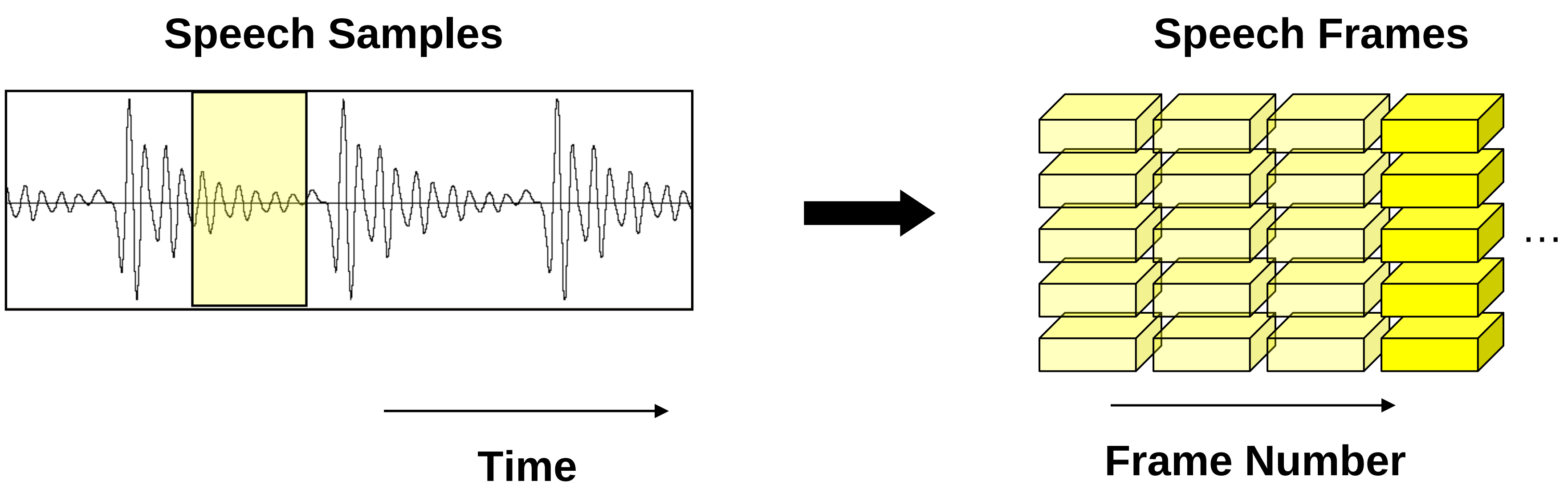
Recognition Systems
Frame-Based Processing

- **The first step in modern speech systems is to convert incoming speech samples into *frames***
- **A typical frame rate for a speech stream is 100 frames per second**



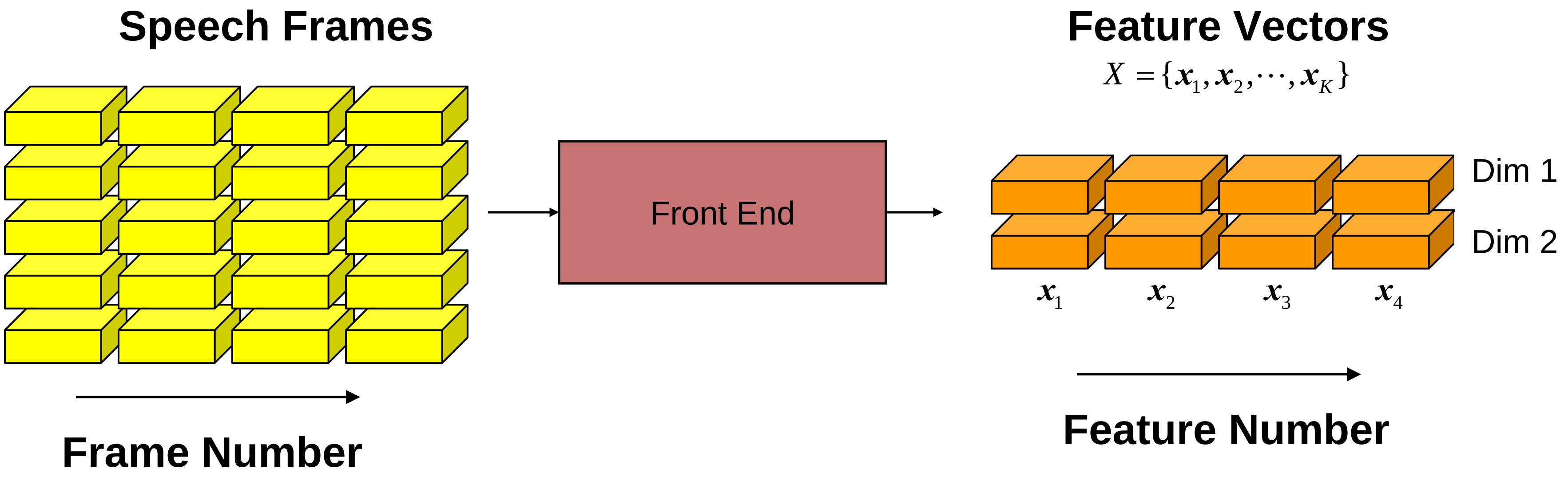
Recognition Systems
Frame-Based Processing

- **The first step in modern speech systems is to convert incoming speech samples into *frames***
- **A typical frame rate for a speech stream is 100 frames per second**



Front-End Processing

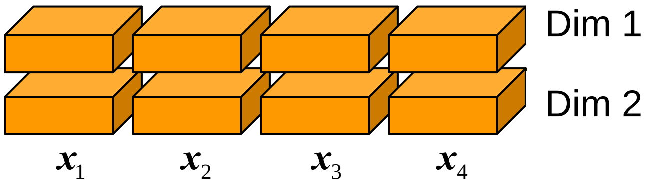
- **Front-end processing converts observed speech frames into an alternative representation, *features***
  - **Lower dimensionality**
  - **Carries information relevant to the problem**



Recognition Systems

Pattern Recognition Training

Training Features



- **A recognition system makes decisions about observed data based on a knowledge of past data**

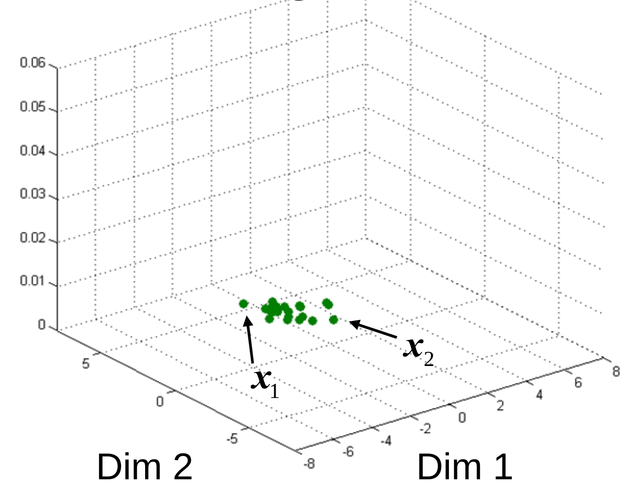
- **During *training*, the system learns about the data it uses to make decisions**
  - **A set of features are collected from a certain language, dialect, or speaker**

Pattern Recognition Training

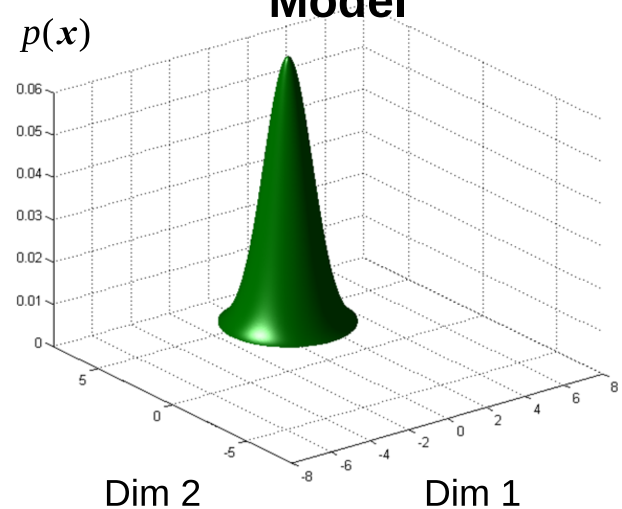
- **A recognition system makes decisions about observed data based on a knowledge of past data**

- **During *training*, the system learns about the data it uses to make decisions**
  - **A set of features are collected from a certain language, dialect, or speaker**
  - **A *model* is generated to represent the data**

Training Features



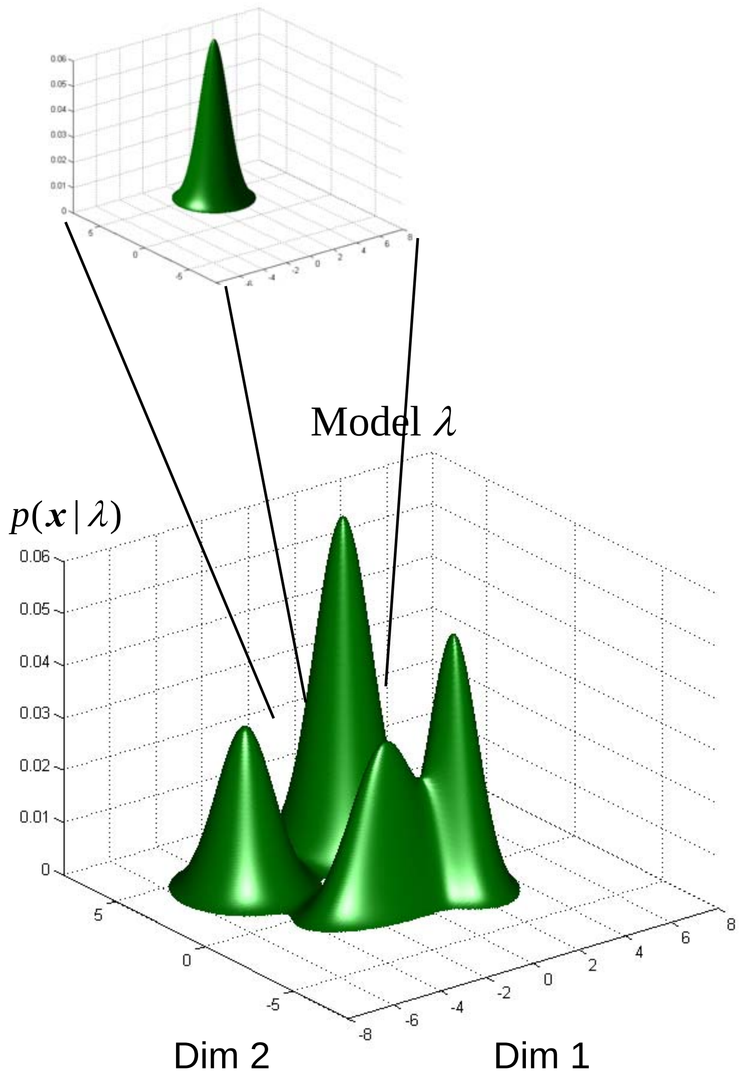
Model



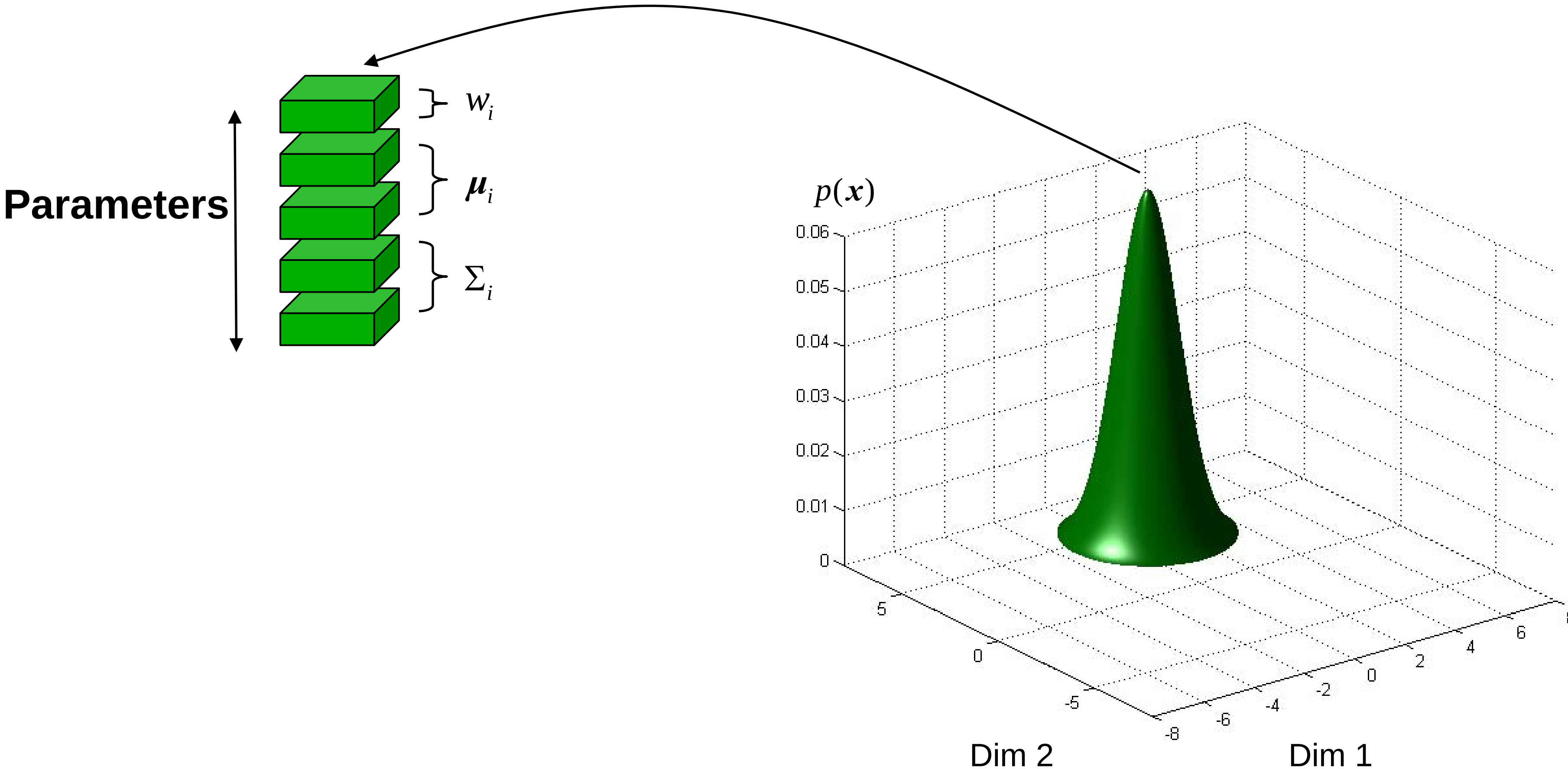
Gaussian Mixture Models

- **A Gaussian mixture model (GMM) represents features as the weighted sum of multiple Gaussian distributions**

- **Each Gaussian *state i* has a**
  - **Mean** $\boldsymbol{\mu}_i$
  - **Covariance** Σ_i
  - **Weight** w_i

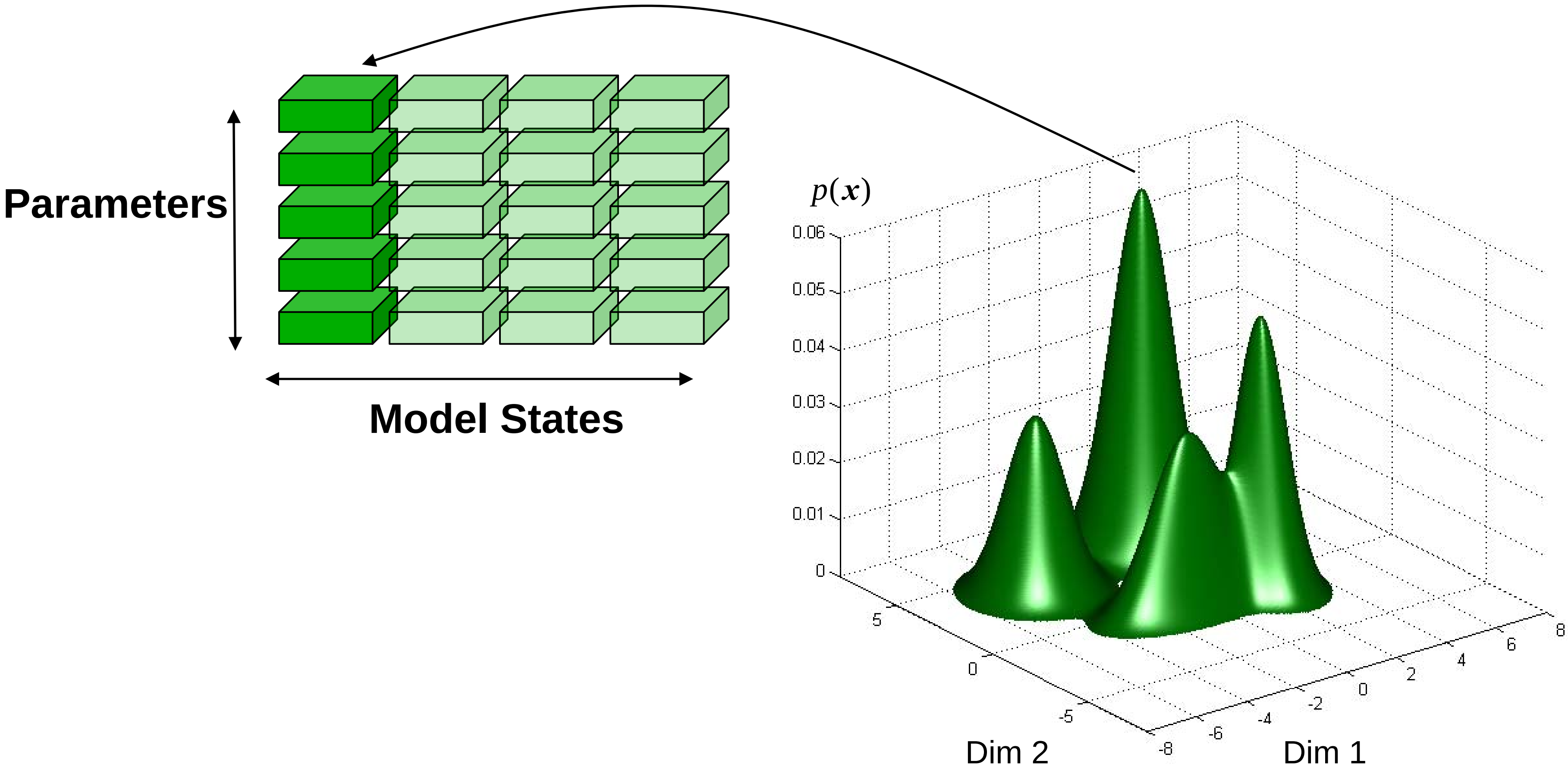


Recognition Systems
Gaussian Mixture Models



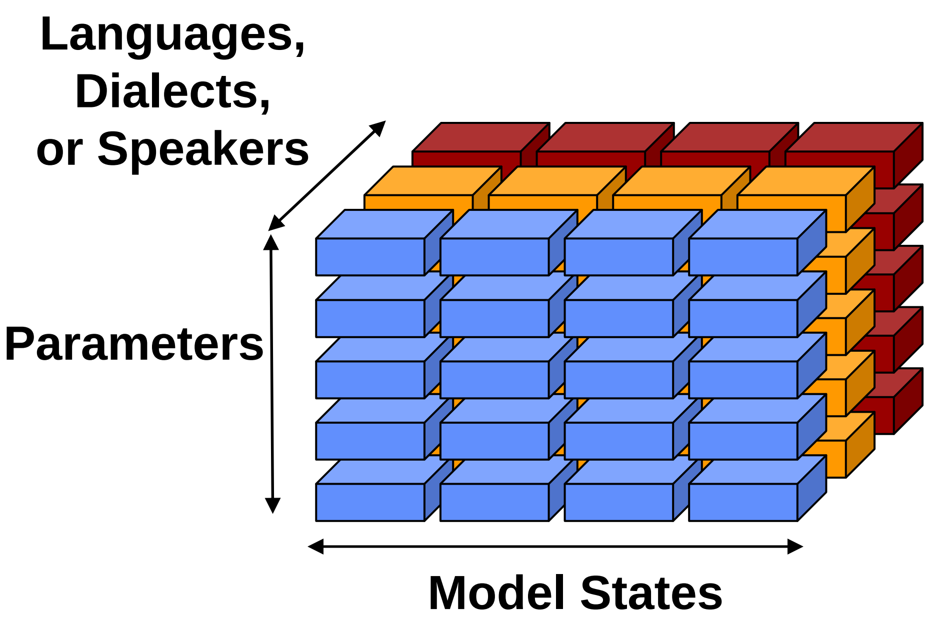
Recognition Systems

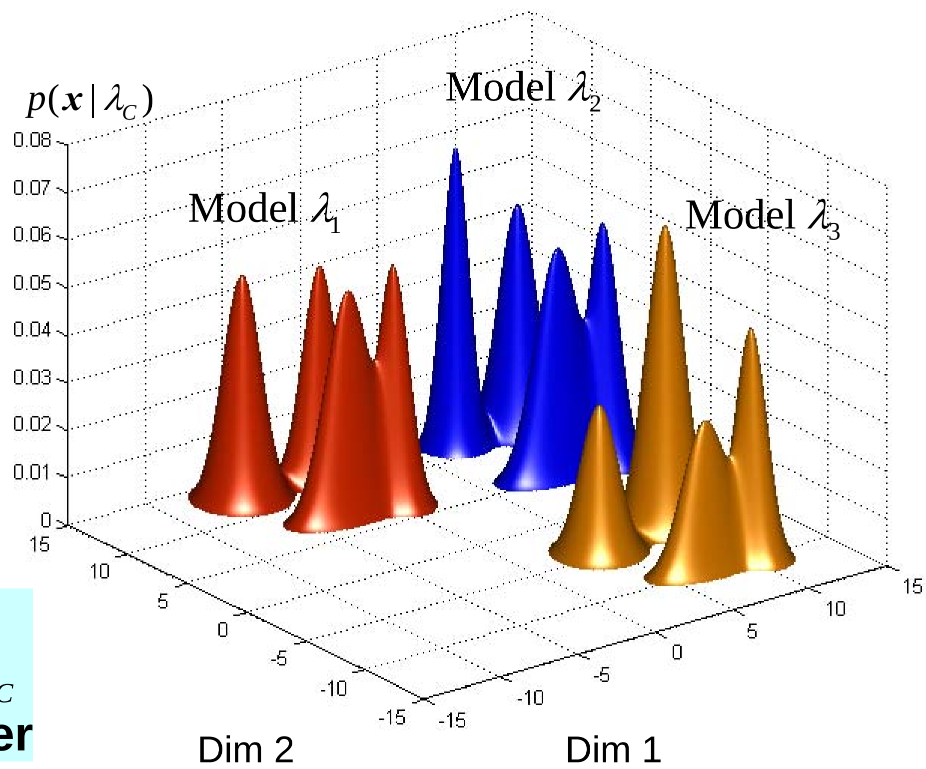
Gaussian Mixture Models



Recognition Systems

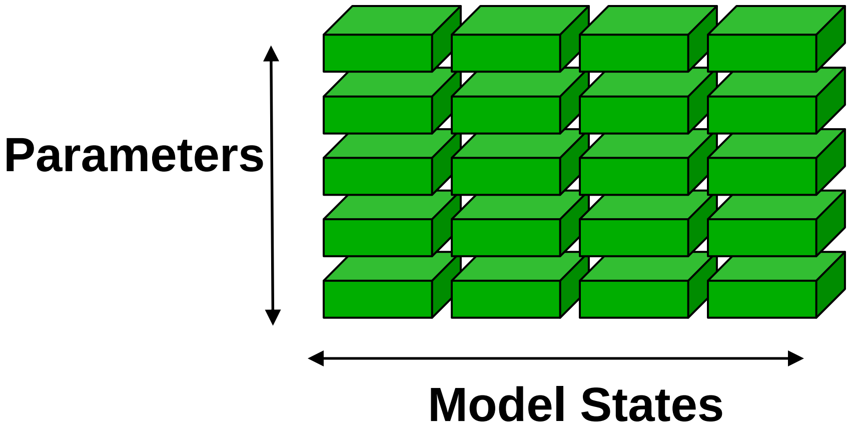
Language, Speaker, and Dialect Models

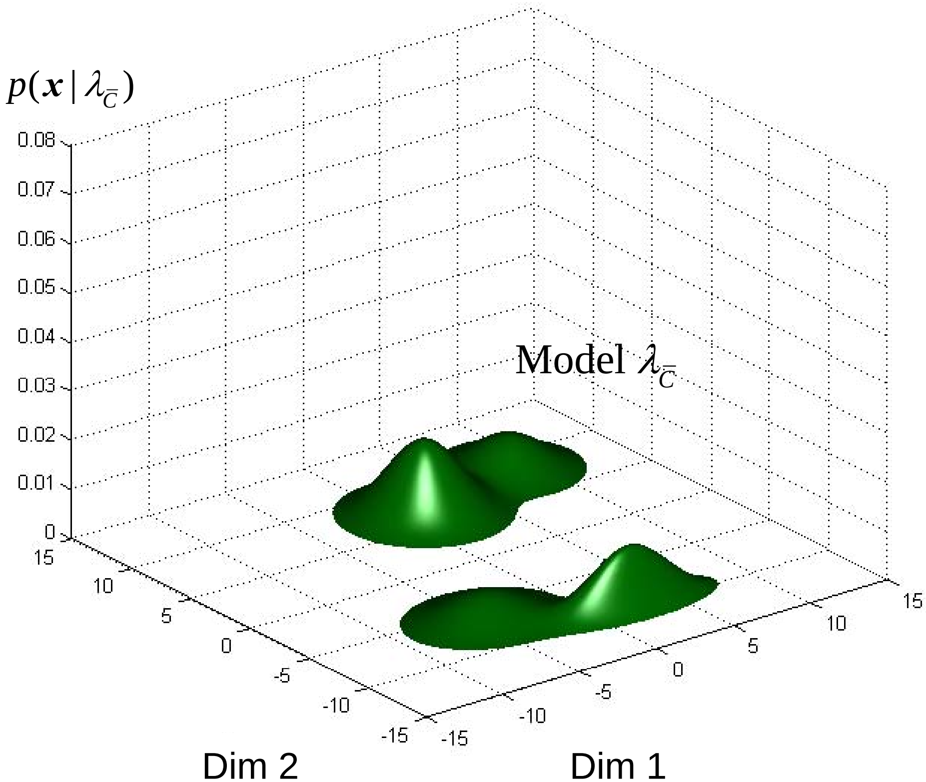




In LID, DID, and SID, we train a set of *target models* λ_C for each dialect, language, or speaker

Recognition Systems
Universal Background Model





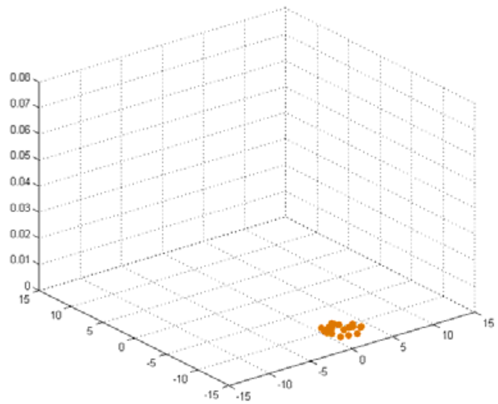
We also train a *universal background model* $\lambda_{\bar{C}}$ representing all speech

Recognition Systems
Hypothesis Test

- **Given a set of *test observations*, we perform a hypothesis test to determine whether a certain class produced it**

$$H_0: \quad X_{test} \text{ is from the hypothesized class}$$
$$H_1: \quad X_{test} \text{ is not from the hypothesized class}$$

$$X_{test} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_K\}$$

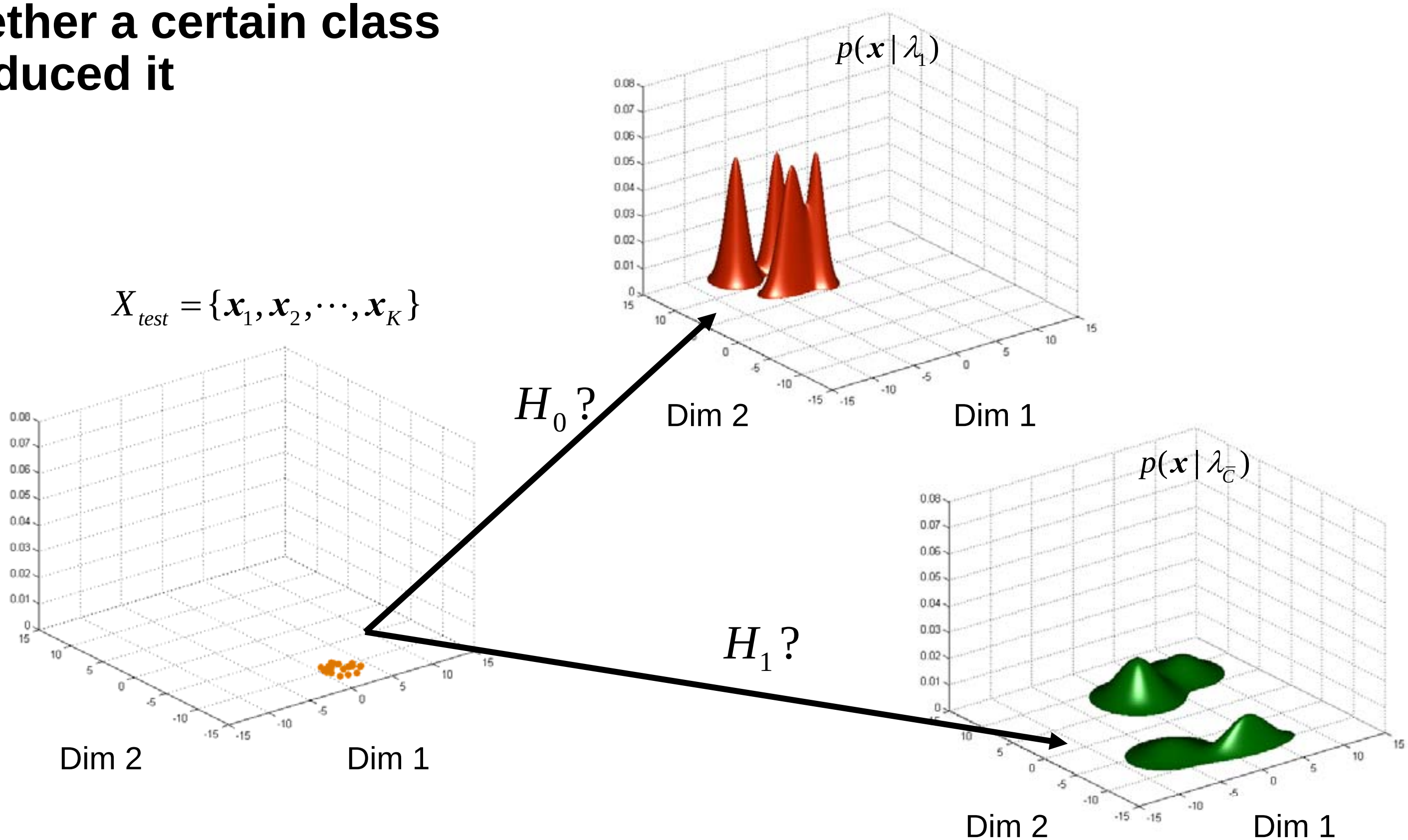


Recognition Systems
Hypothesis Test

$H_0: \quad X_{test}$ is from the hypothesized class

$H_1: \quad X_{test}$ is not from the hypothesized class

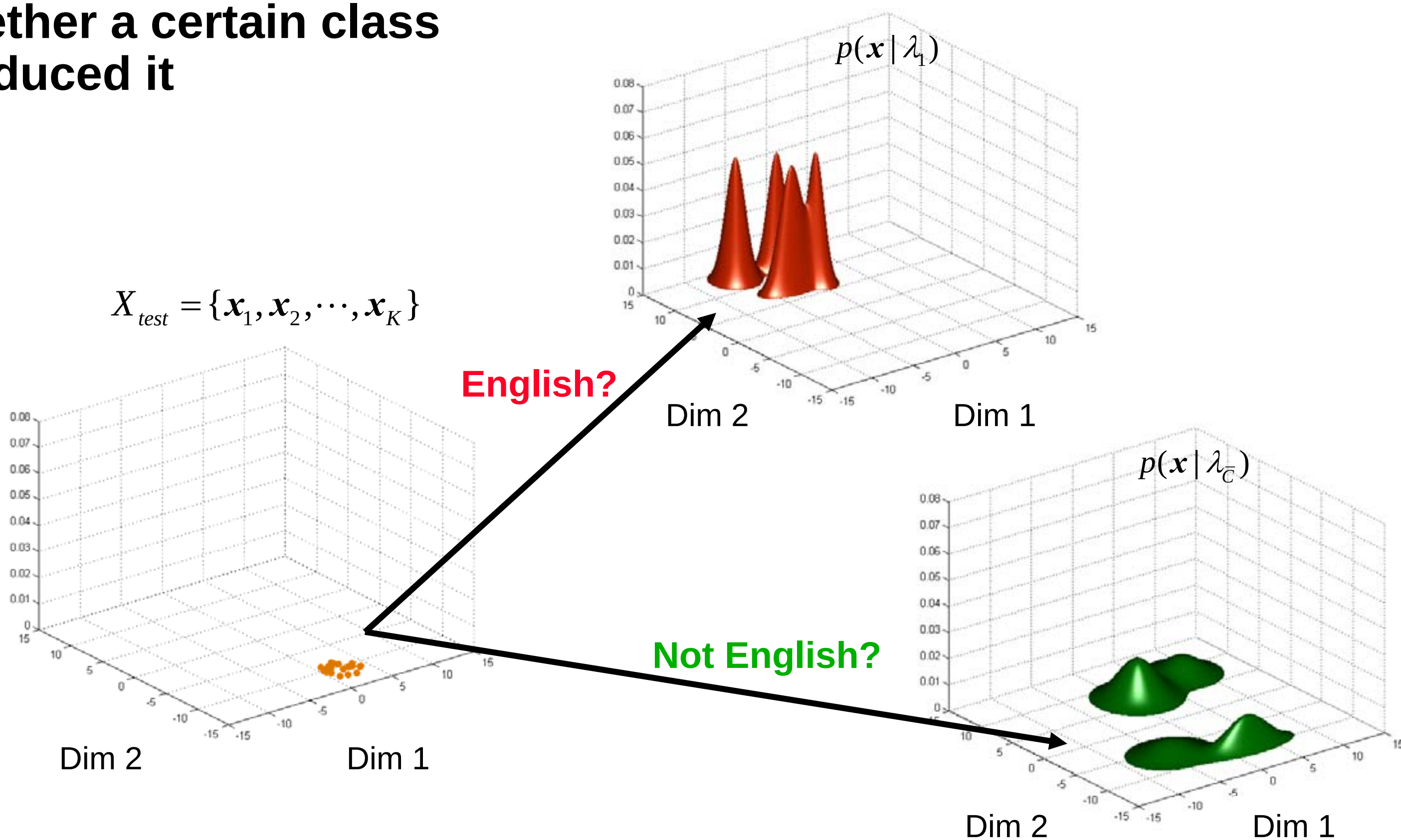
- **Given a set of *test observations*, we perform a hypothesis test to determine whether a certain class produced it**



Recognition Systems

Hypothesis Test

- **Given a set of *test observations*, we perform a hypothesis test to determine whether a certain class produced it**



Log-Likelihood Ratio Score

- **We determine which hypothesis is true using the ratio:**

$$\frac{p(X \mid H_0)}{p(X \mid H_1)} \begin{cases} \geq \text{threshold}, & \text{accept } H_0 \\ \leq \text{threshold}, & \text{reject } H_0 \end{cases}$$

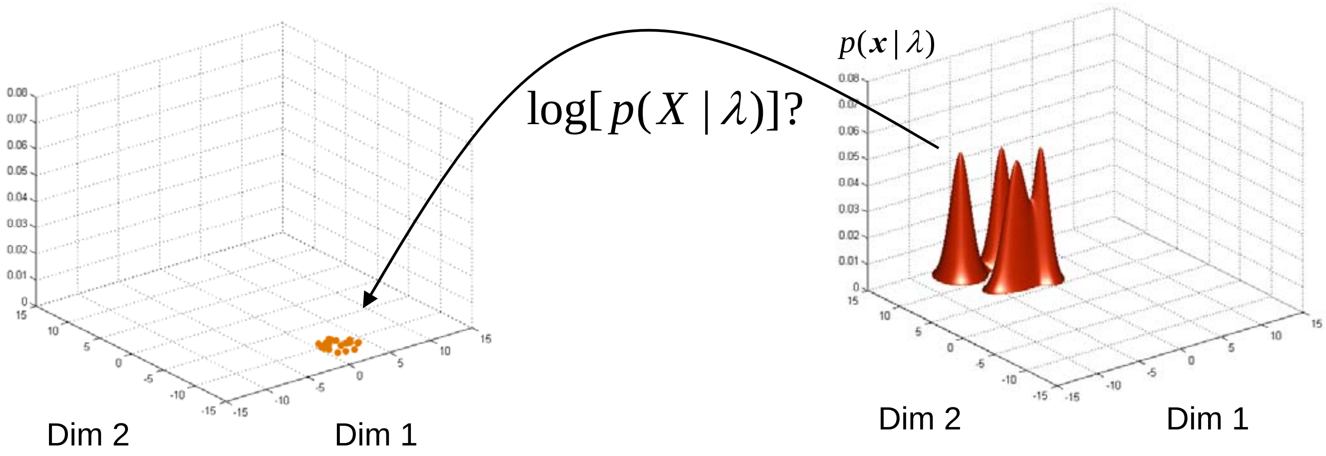
- **We use the *log-likelihood ratio score* to decide whether an observed speaker, language, or dialect is the target**

$$\Lambda(X) = \log[p(X \mid \lambda_C)] - \log[p(X \mid \lambda_{\overline{C}})]$$

$$\Lambda(X) \begin{cases} \geq \text{threshold}, & X \text{ generated by } \lambda_C \\ < \text{threshold}, & X \text{ generated by } \lambda_{\overline{C}} \end{cases}$$

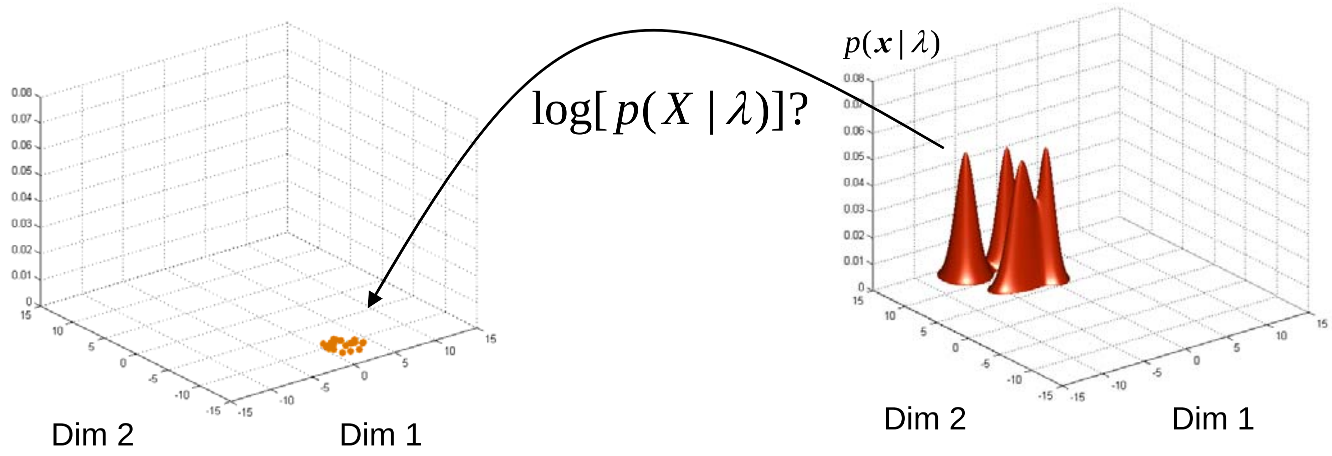
Recognition Systems
Log-Likelihood Computation

- **The observation log-likelihood given a model λ is:**



Log-Likelihood Computation

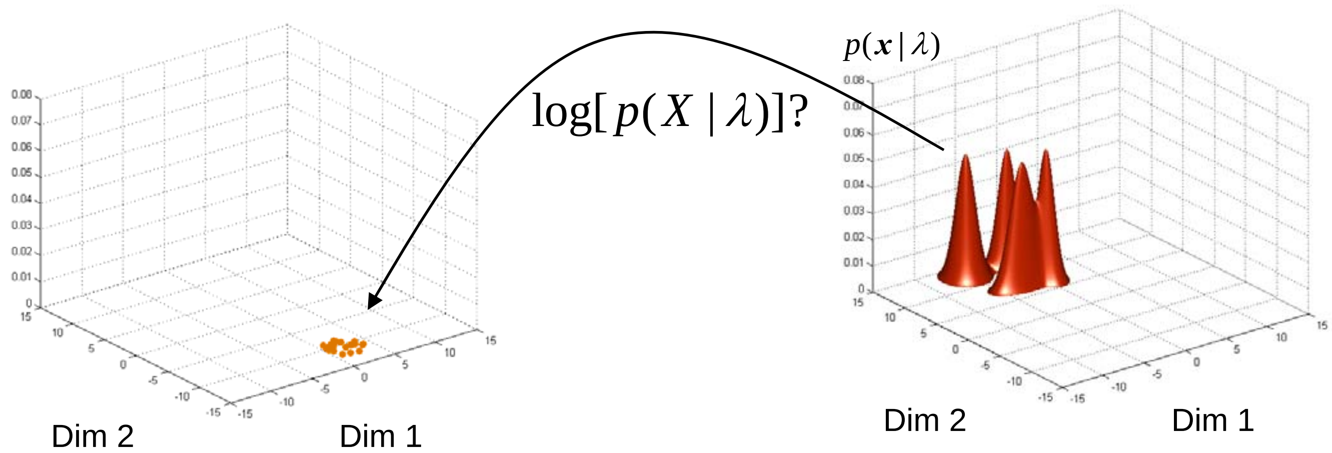
- **The observation log-likelihood given a model λ is:**



$$\log[p(X \mid \lambda)] = \frac{1}{K}\sum_{1}^{K}\left(\log\sum_{i=1}^{M}\exp\left(C_i - \frac{1}{2}\underbrace{(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)}_{\textbf{Dot product}}\right)\right)$$

Recognition Systems
Log-Likelihood Computation

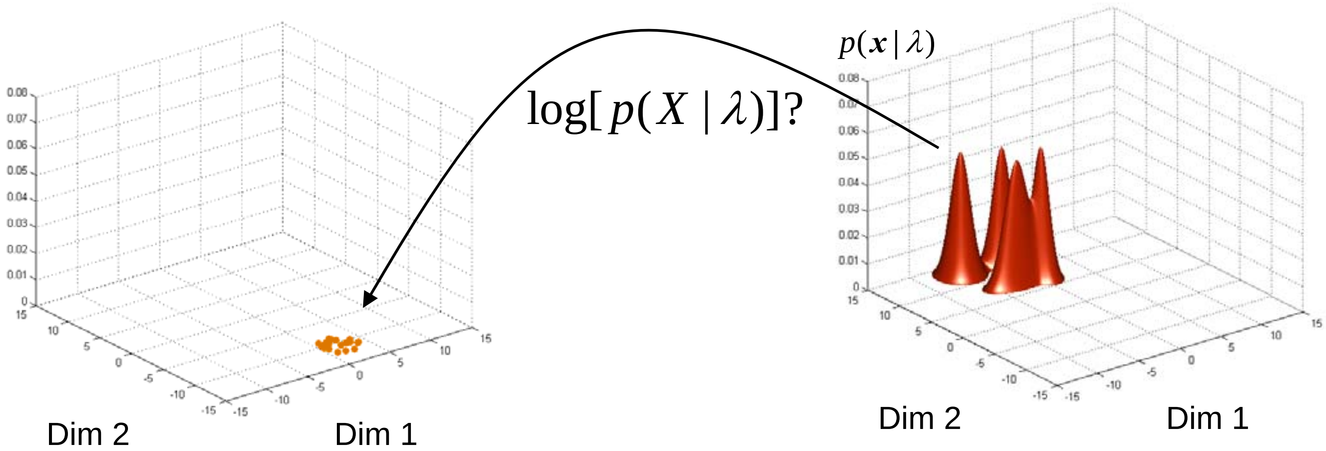
- **The observation log-likelihood given a model λ is:**



$$\log[p(X \mid \lambda)] = \frac{1}{K}\sum_{1}^{K}\left(\log\sum_{i=1}^{M}\exp\left(\underbrace{C_i}_{\text{Constant derived from weight and covariance}} - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)\right)$$

Log-Likelihood Computation

- **The observation log-likelihood given a model λ is:**

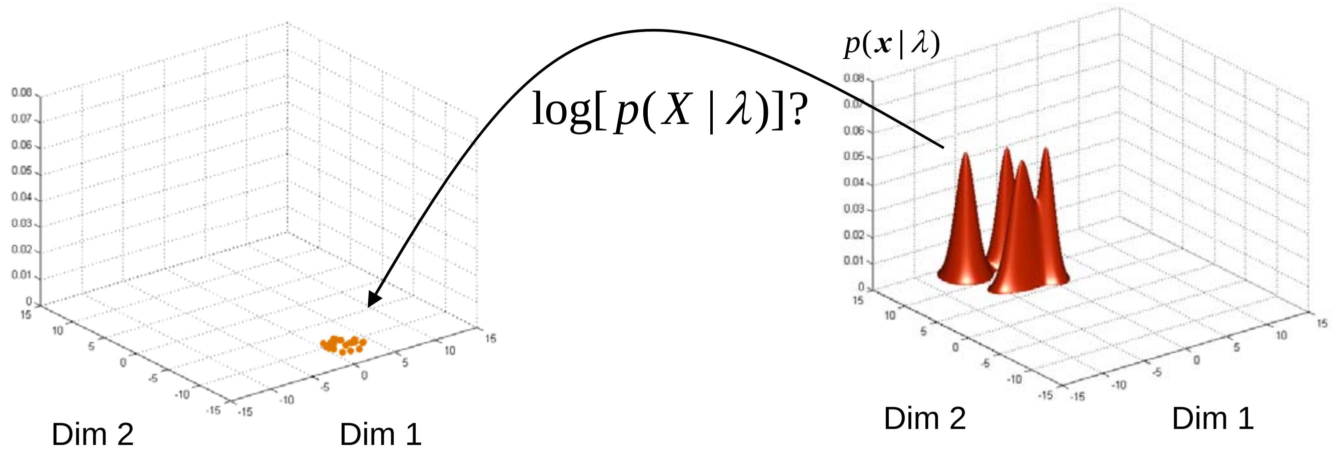


$$\log[p(X \mid \lambda)] = \tfrac{1}{K}\sum_{1}^{K}\left(\underbrace{\log\sum_{i=1}^{M}\exp}_{\textbf{Table lookup used to compute this function}}\left(C_i - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)\right)$$

Recognition Systems

Log-Likelihood Computation

- **The observation log-likelihood given a model λ is:**



$$\log[p(X \mid \lambda)] = \underbrace{\frac{1}{K}\sum_{1}^{K}}\left(\log \sum_{i=1}^{M} \exp\left(C_i - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)\right)$$

Sum over all *K* features

Outline

- Introduction
- Recognition for speech applications using GMMs
- Parallel implementation of the GMM
- Performance model
- Conclusions and future work

Parallel Implementation of the GMM

Summary

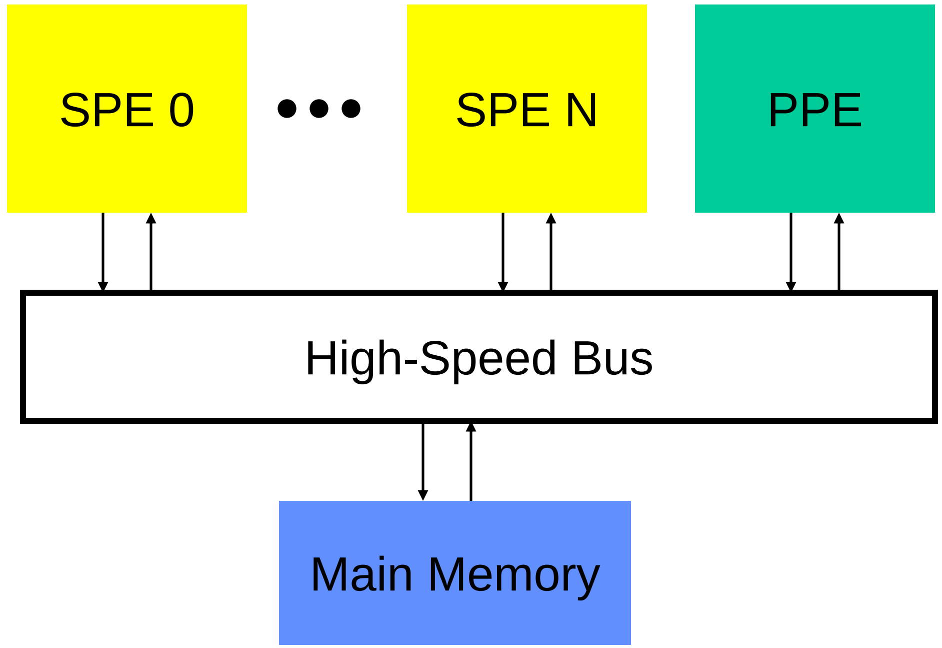
- **We have developed an algorithm to perform GMM scoring on the Cell processor**
- **This scoring stage of pattern recognition is where much of the time is spent in current systems**
- **This section:**
  - **Describes the Cell Broadband Engine architecture**
  - **Summarizes the strengths and limitations of the Cell**
  - **Discusses step-by-step the algorithm we developed for GMM scoring on the Cell**

Parallel Implementation of the GMM

Cell Architecture

- **The Cell Broadband Engine has leading performance-per-watt specifications in its class**

- **Synergistic processing elements (SPEs)**
  - **256KB of local store memory**
  - **25.6 GFLOPs per SPE**
  - **SIMD instructions**
- **PowerPC processor element (PPE)**
- **PPE and multiple SPEs operate in parallel and communicate via a high-speed bus**
  - **12.8e9 bytes/second (one way)**
- **Each SPE can transfer data from main memory using DMA**
  - **PPE can effectively “send” data to the SPEs using this method**



Parallel Implementation of the GMM
Cell Design Principles

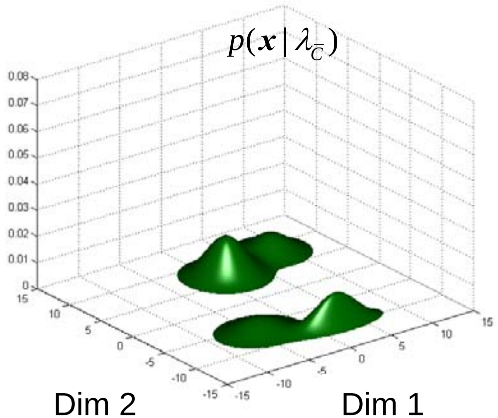
- **Limitations of the Cell processor**
  - Size of local store is small—only 256KB
  - All SPE data must explicitly be transferred in and out of local store
  - The PPE is much slower than the SPEs
- **Solutions to maximize throughput**
  - Do computations on SPEs when possible
  - Minimize time when SPEs are idle
  - Keep commonly-used data on SPEs to avoid cost of transferring to local store

Parallel Implementation of the GMM

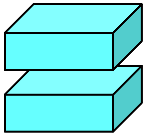
Algorithm: Background Scoring

- **Begin with a background model and a single feature vector**

On PPE



On PPE

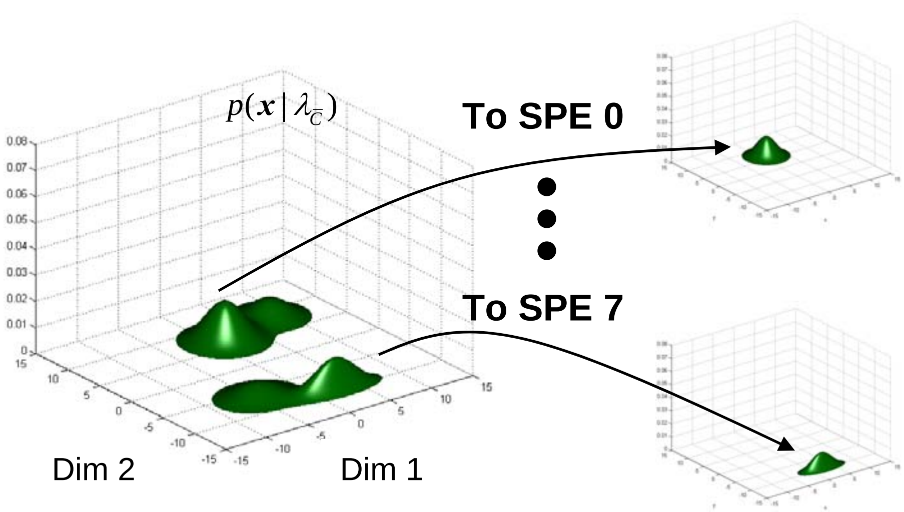


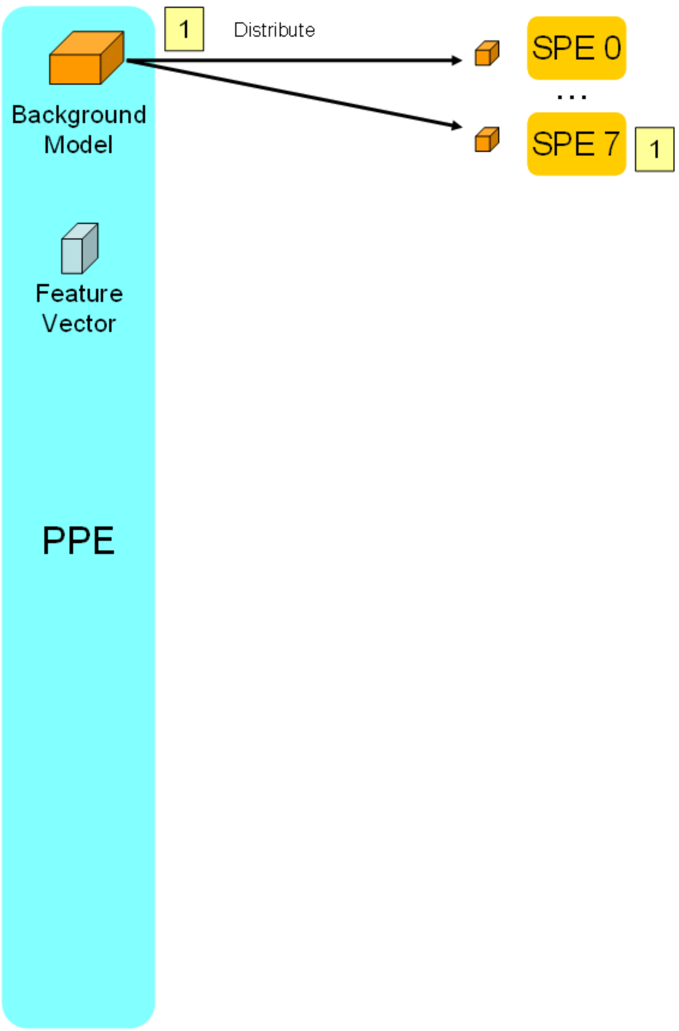
$\boldsymbol{x}_1$

Parallel Implementation of the GMM

Algorithm Step 1

- **Broadcast the background model to the SPEs**
  - **616K model is split across SPEs since it will not fit on single SPE**
  - **Kept on SPEs throughout scoring procedure**

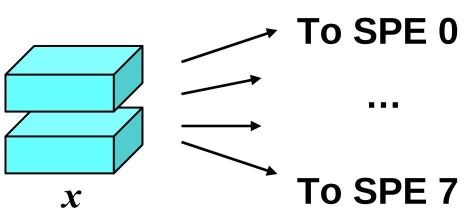


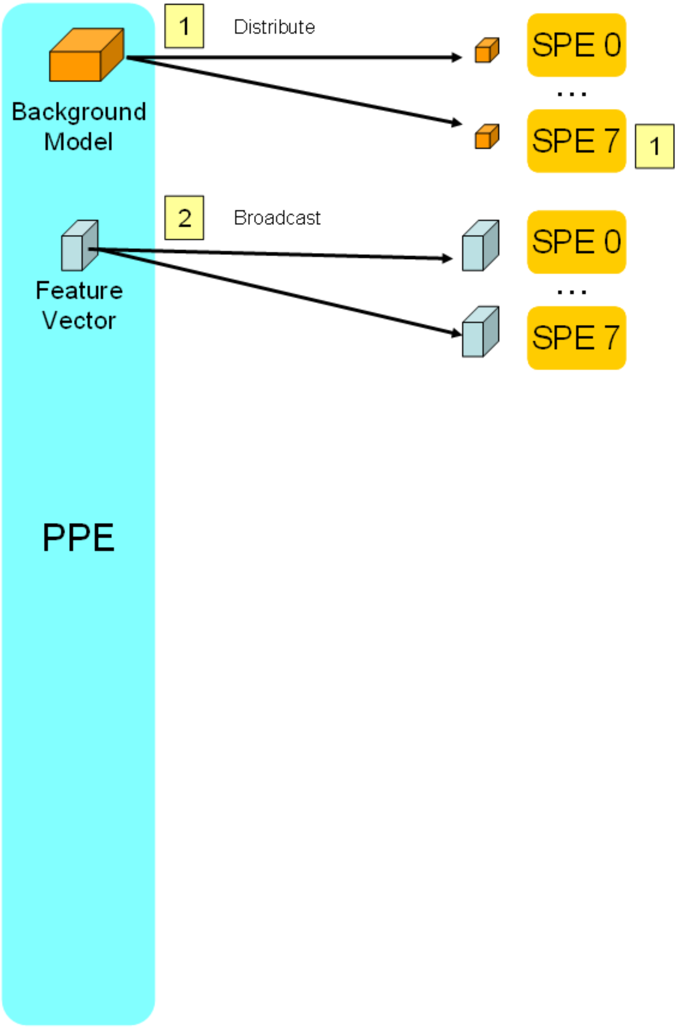


Parallel Implementation of the GMM

Algorithm Step 2

- **Broadcast copy of the feature vector to the SPEs**

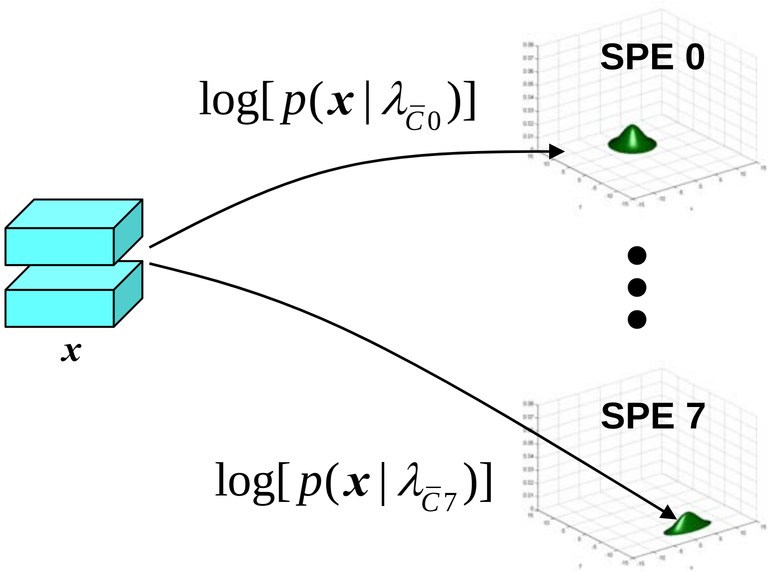


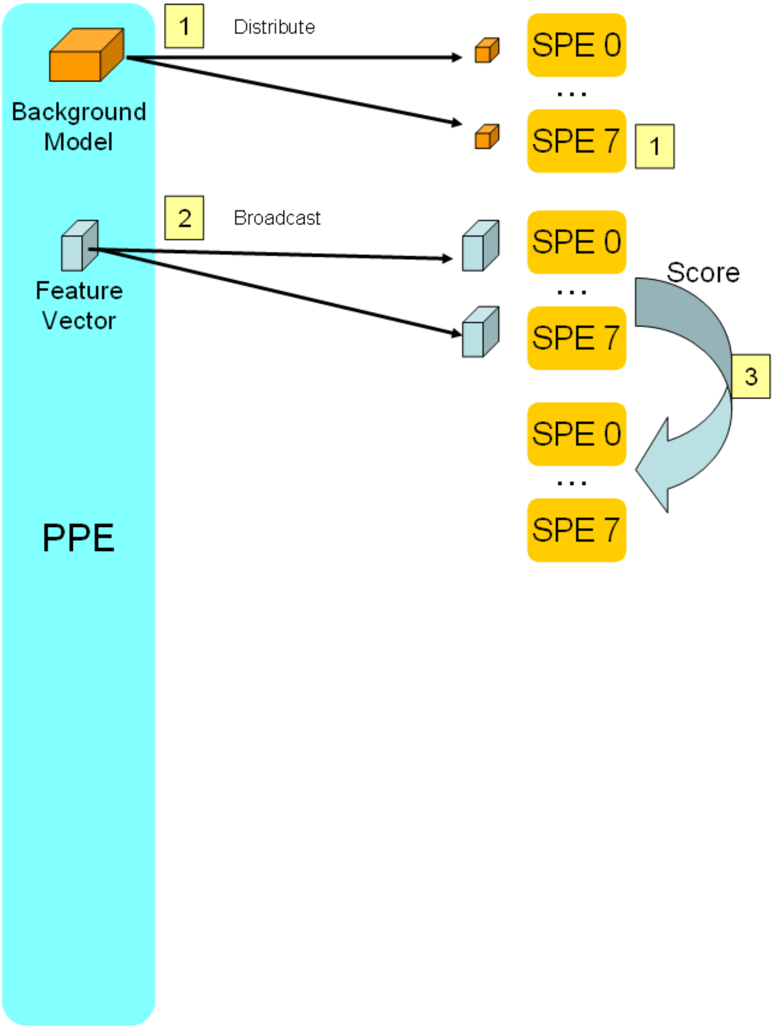


Parallel Implementation of the GMM

Algorithm Step 3

- **Score the feature vector against the background model states on each SPE**





Parallel Implementation of the GMM
Algorithm Step 4

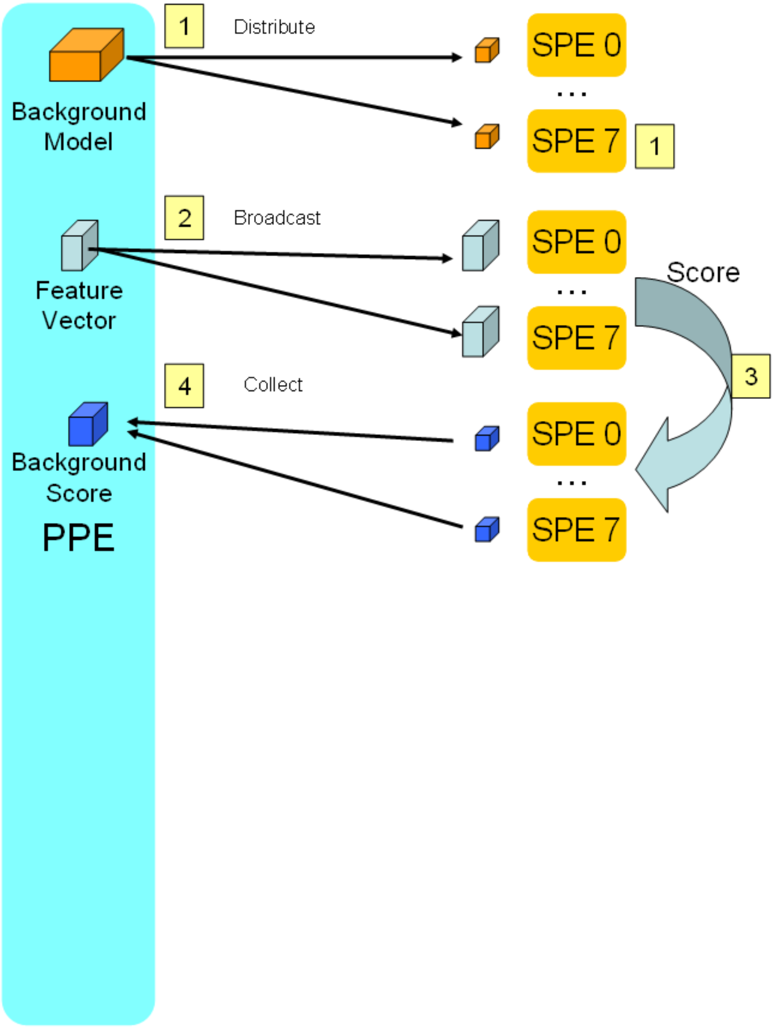
- **Move the background scores to the PPE and *aggregate***

On SPEs

$$\log[p(\boldsymbol{x} \mid \lambda_{\bar{C}0})] \quad \bullet\bullet\bullet \quad \log[p(\boldsymbol{x} \mid \lambda_{\bar{C}7})]$$

On PPE

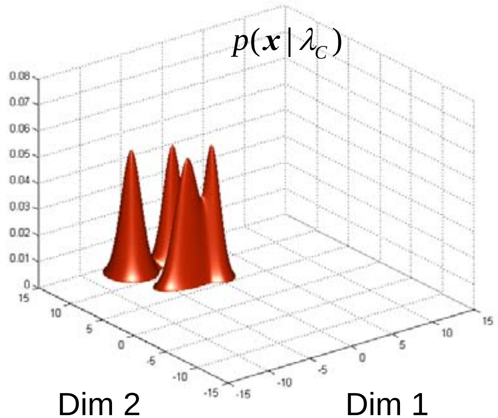
$$\log[p(\boldsymbol{x} \mid \lambda_{\bar{C}})] = \log \sum_{r=0}^{7} \exp\left(\log[p(\boldsymbol{x} \mid \lambda_{\bar{C}r})]\right)$$



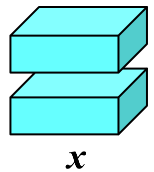
Parallel Implementation of the GMM
Algorithm: Target Scoring

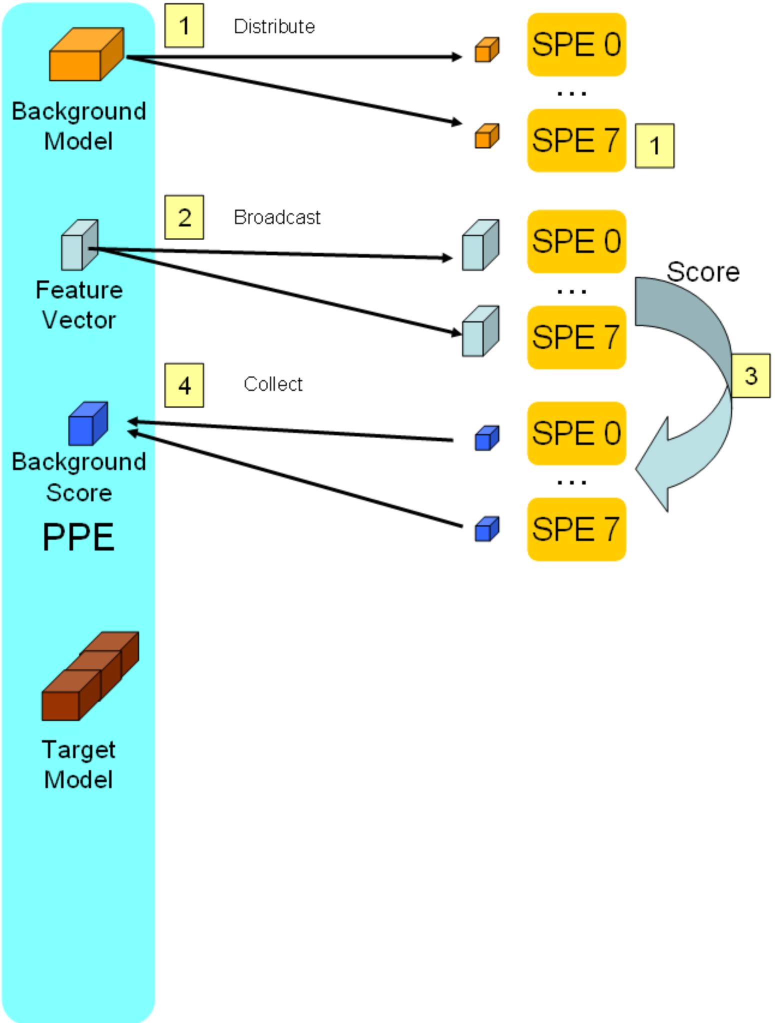
- **Begin with a target model and keep the single feature vector on the SPEs**

On PPE



On SPEs

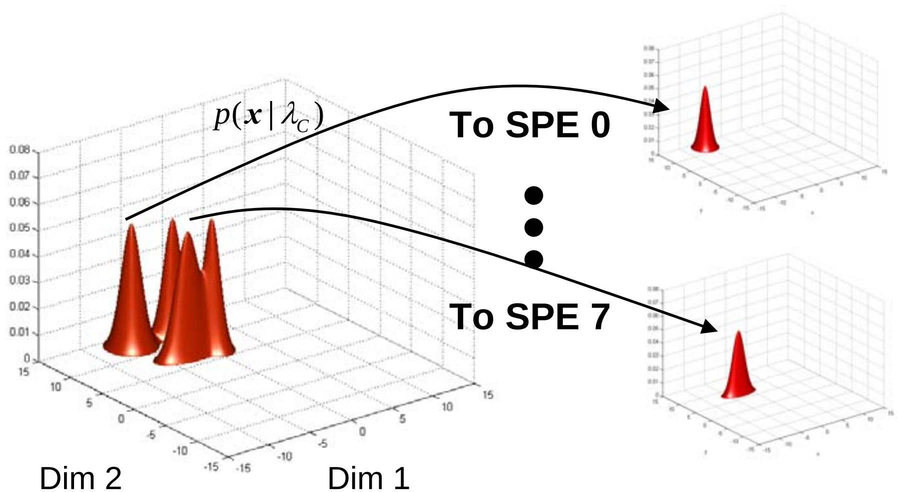


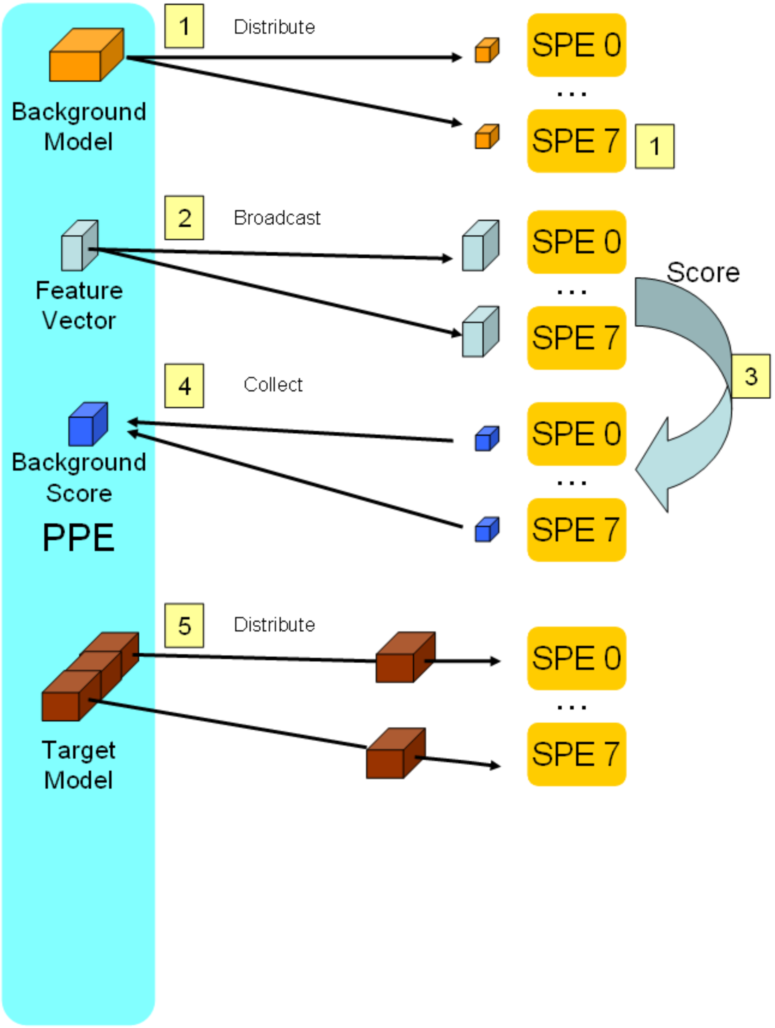


Parallel Implementation of the GMM

Algorithm Step 5

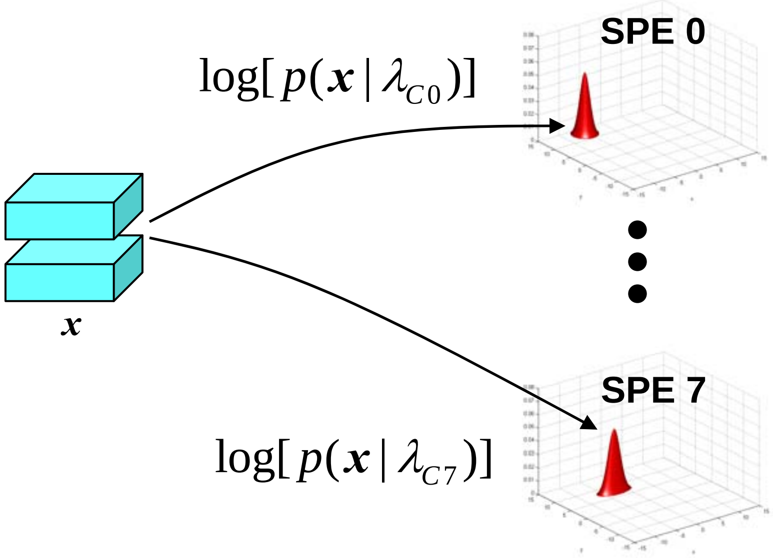
- **Distribute target model states to the SPEs**
  - **Only a subset of states need to be scored (called *Gaussian short-lists*)**

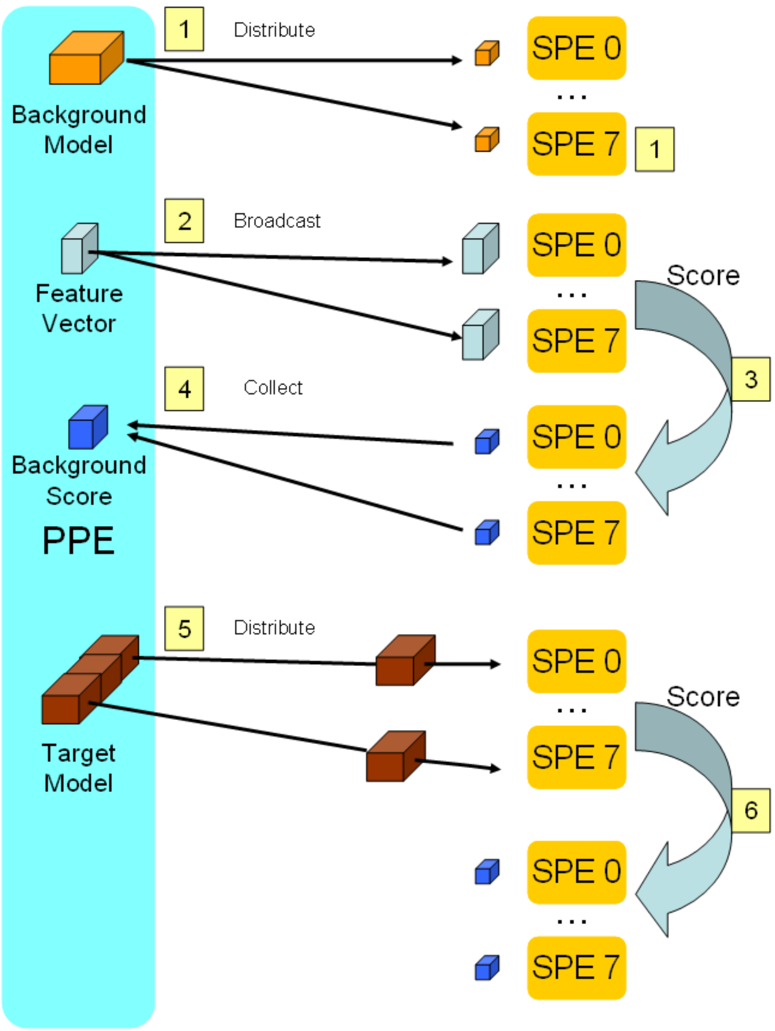




Parallel Implementation of the GMM
Algorithm Step 6

- **Score feature vectors against target models**





Parallel Implementation of the GMM

Algorithm Step 7

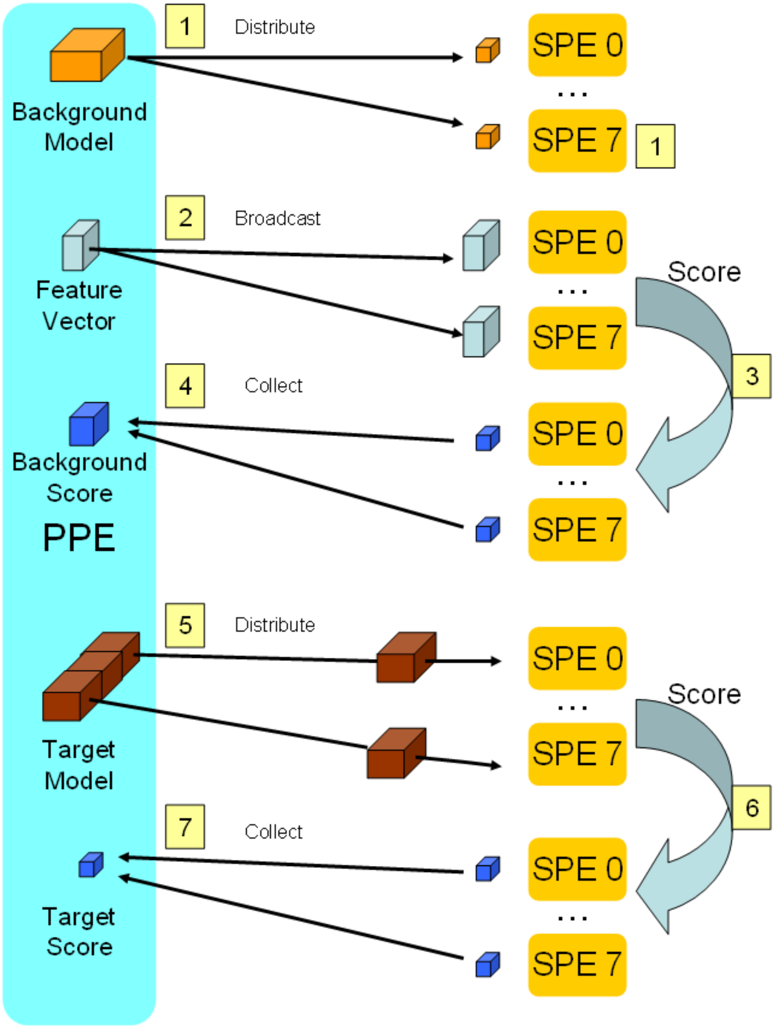
- **Collect target scores from SPEs and aggregate**

On SPEs

$$\log[p(\boldsymbol{x}\,|\,\lambda_{C0})] \quad \bullet\bullet\bullet \quad \log[p(\boldsymbol{x}\,|\,\lambda_{C7})]$$

On PPE

$$\log[p(\boldsymbol{x}\,|\,\lambda_{C})] = \log \sum_{r=0}^{7} \exp\left(\log[p(\boldsymbol{x}\,|\,\lambda_{Cr})]\right)$$



Parallel Implementation of the GMM

Implementation Challenges

- **We have begun implementing our algorithm on the Cell processor**

- **Implementing vectorization is a challenge**
  - **Concentrate on optimizing dot product and aggregation algorithms**

$$\log[p(X \mid \lambda)] = \tfrac{1}{K}\sum_{1}^{K}\left(\log\sum_{i=1}^{M}\exp\left(C_i - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)\right)$$

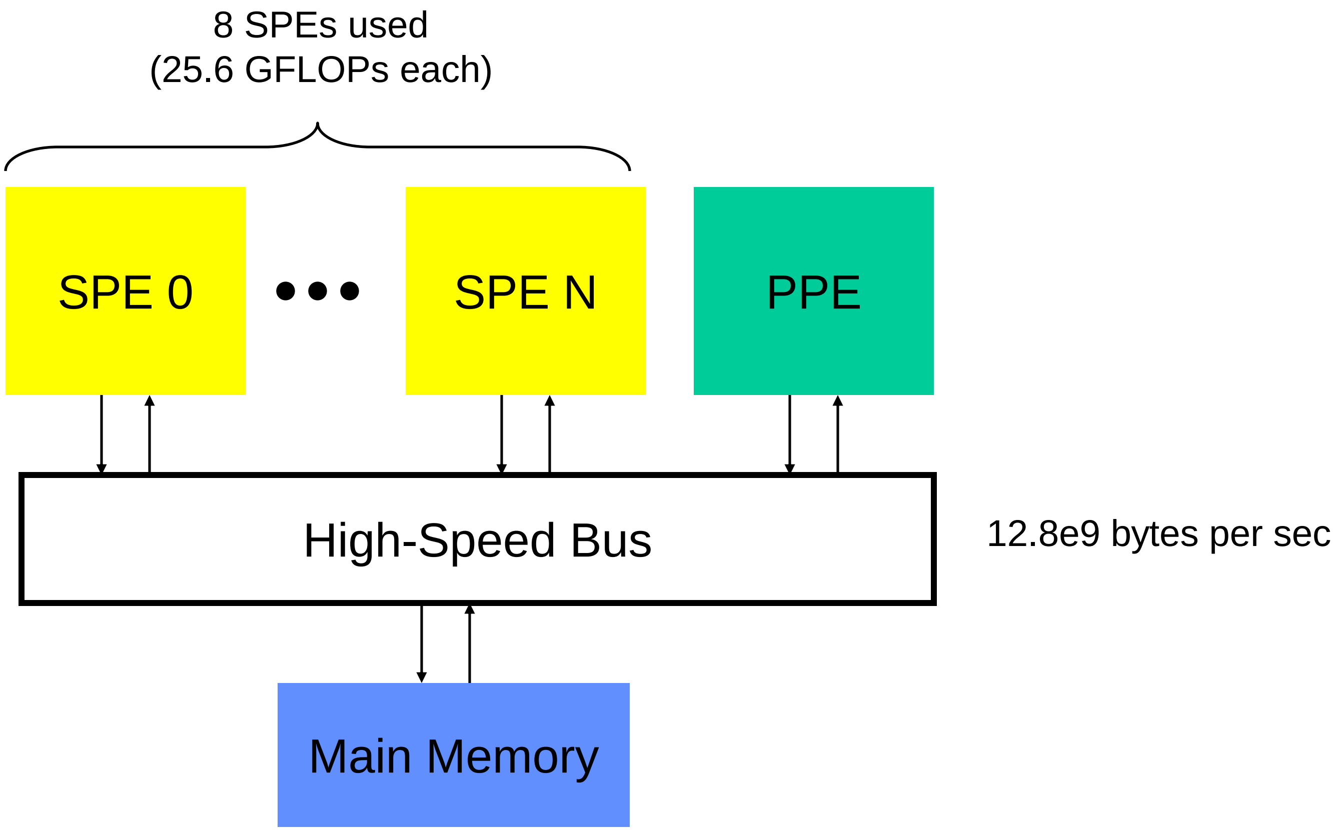
- **Designing data transfers is another challenging problem**
  - **Subdividing and distributing the models to minimize transfer time**
  - **Timing transfers so that they overlap with computation (double buffering)**

Outline

- Introduction
- Recognition for speech applications using GMMs
- Parallel implementation of the GMM
- Performance model
- Conclusions and future work

Performance Model

Cell Resources

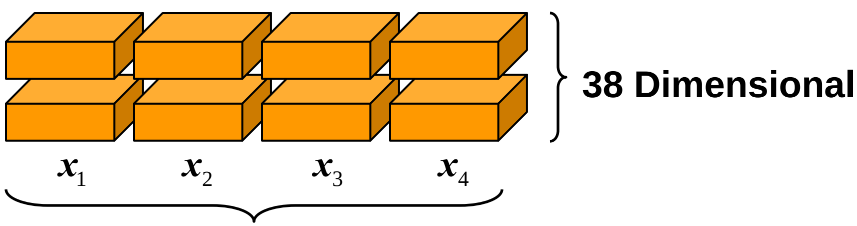


Performance Model

Data Structures

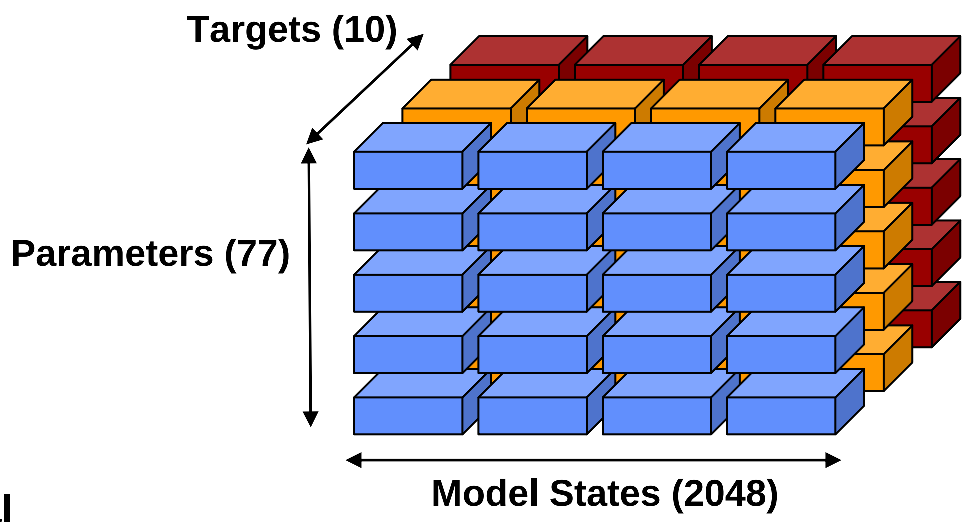
Feature Vectors

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_K\}$$

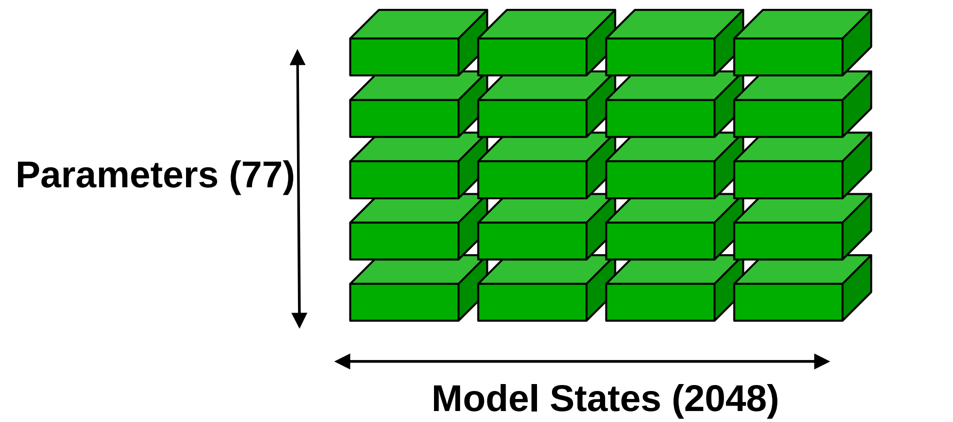


100 Features per Second
For Each Stream

Target Models



Universal Background Model

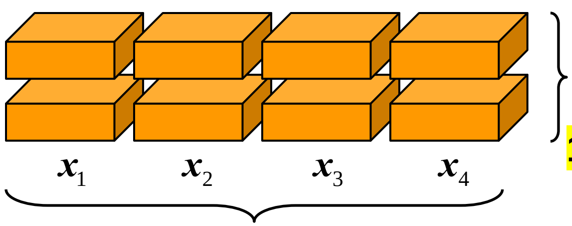


Performance Model

Data Structures

Feature Vectors

$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_K\}$

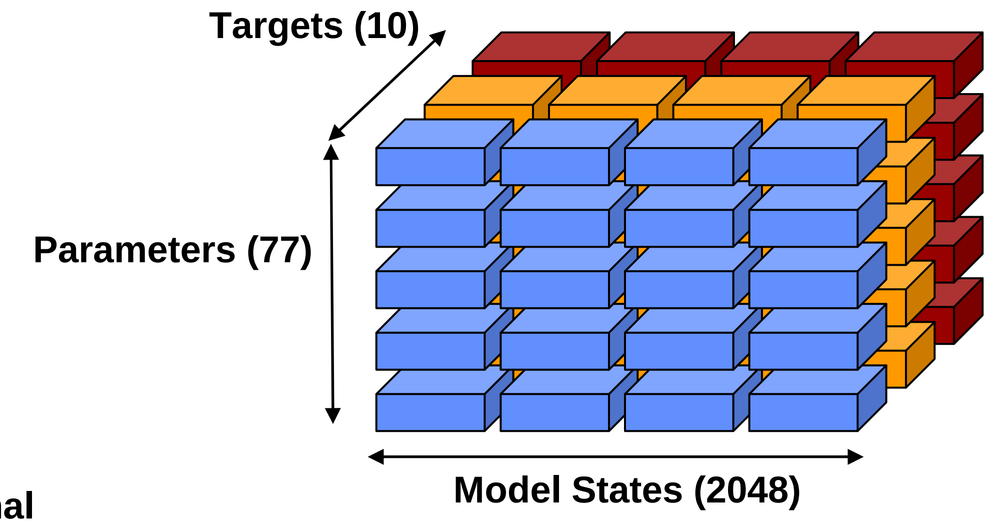


38 Dimensional

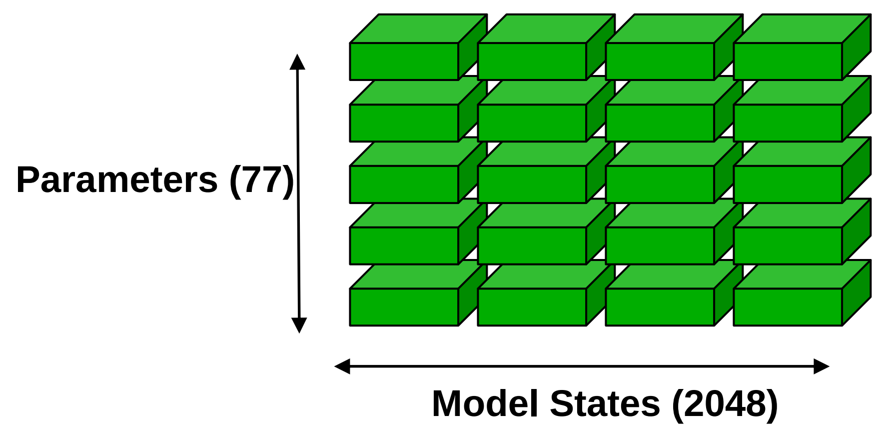
152 Bytes Per Frame

100 Features per Second
For Each Stream

Target Models



Universal Background Model

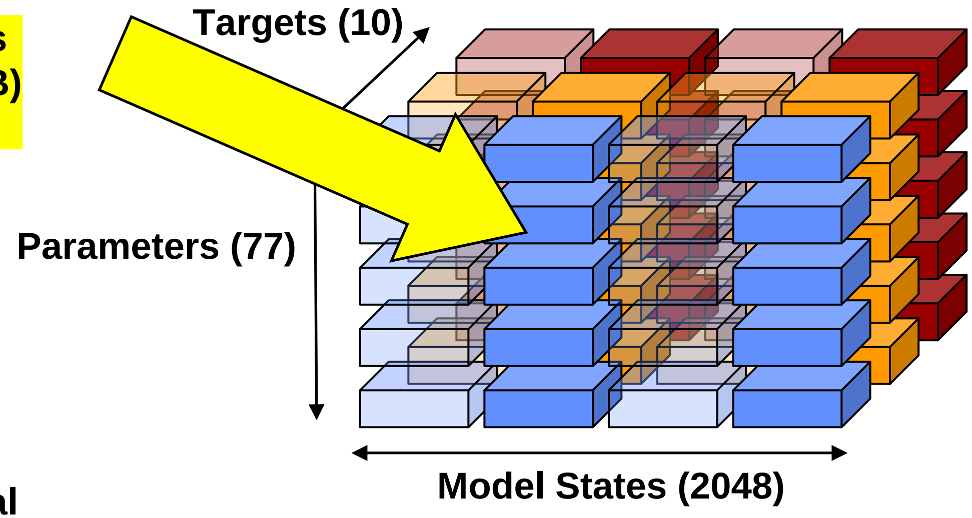


Performance Model

Data Structures

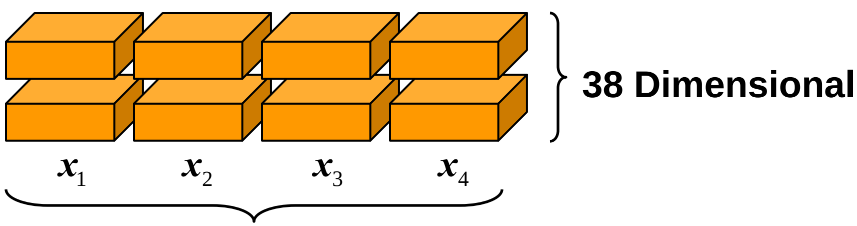
6 MB for all Targets
Only 5 States (15 KB)
Used Per Frame

Target Models



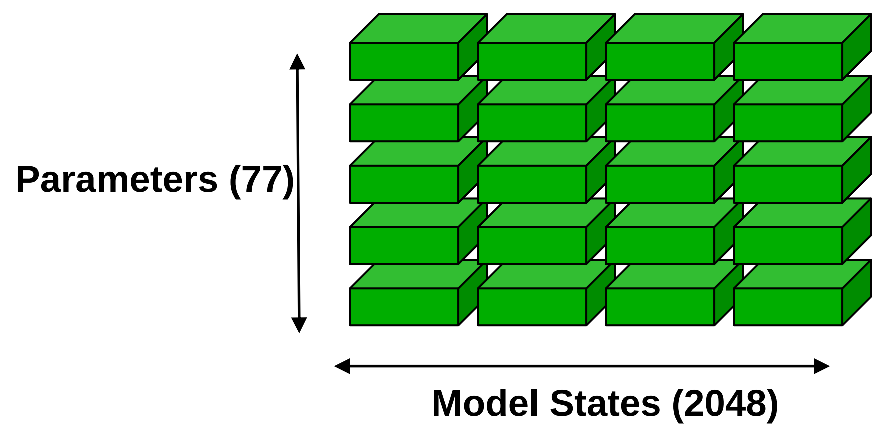
Feature Vectors

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_K\}$$



100 Features per Second
For Each Stream

Universal Background Model

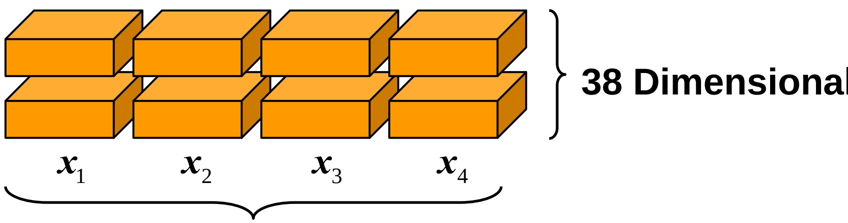


Performance Model

Data Structures

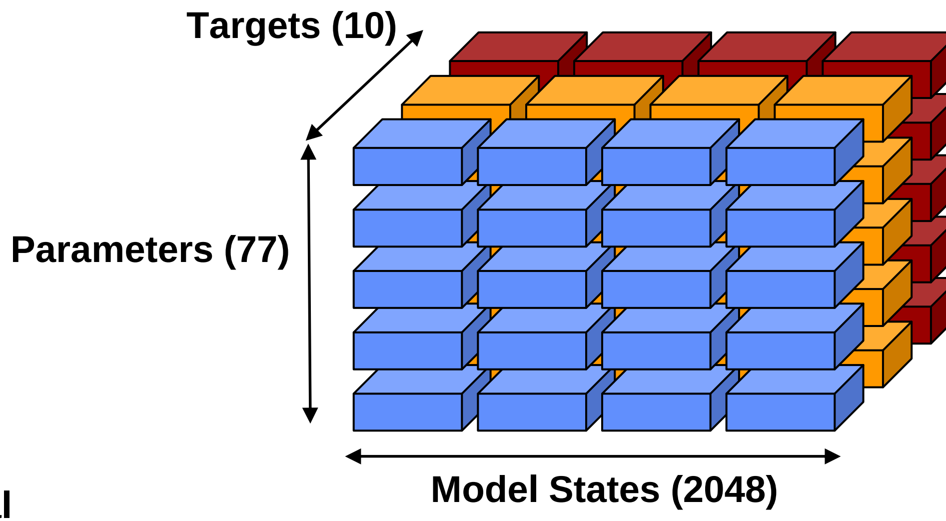
Feature Vectors

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_K\}$$

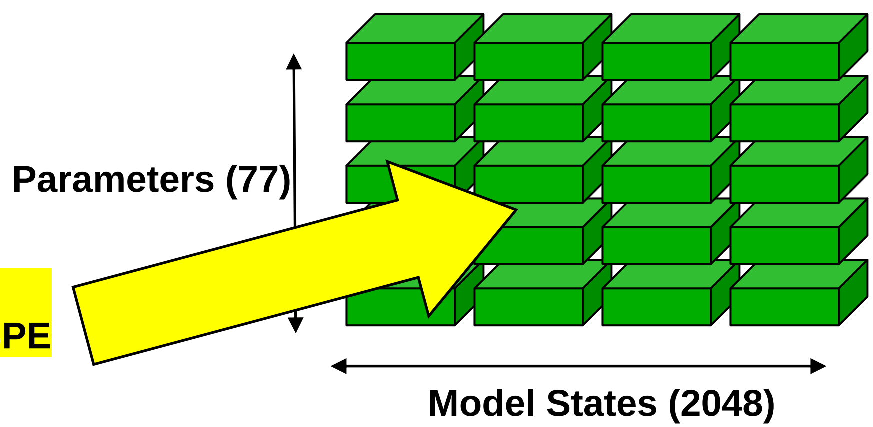


100 Features per Second
For Each Stream

Target Models



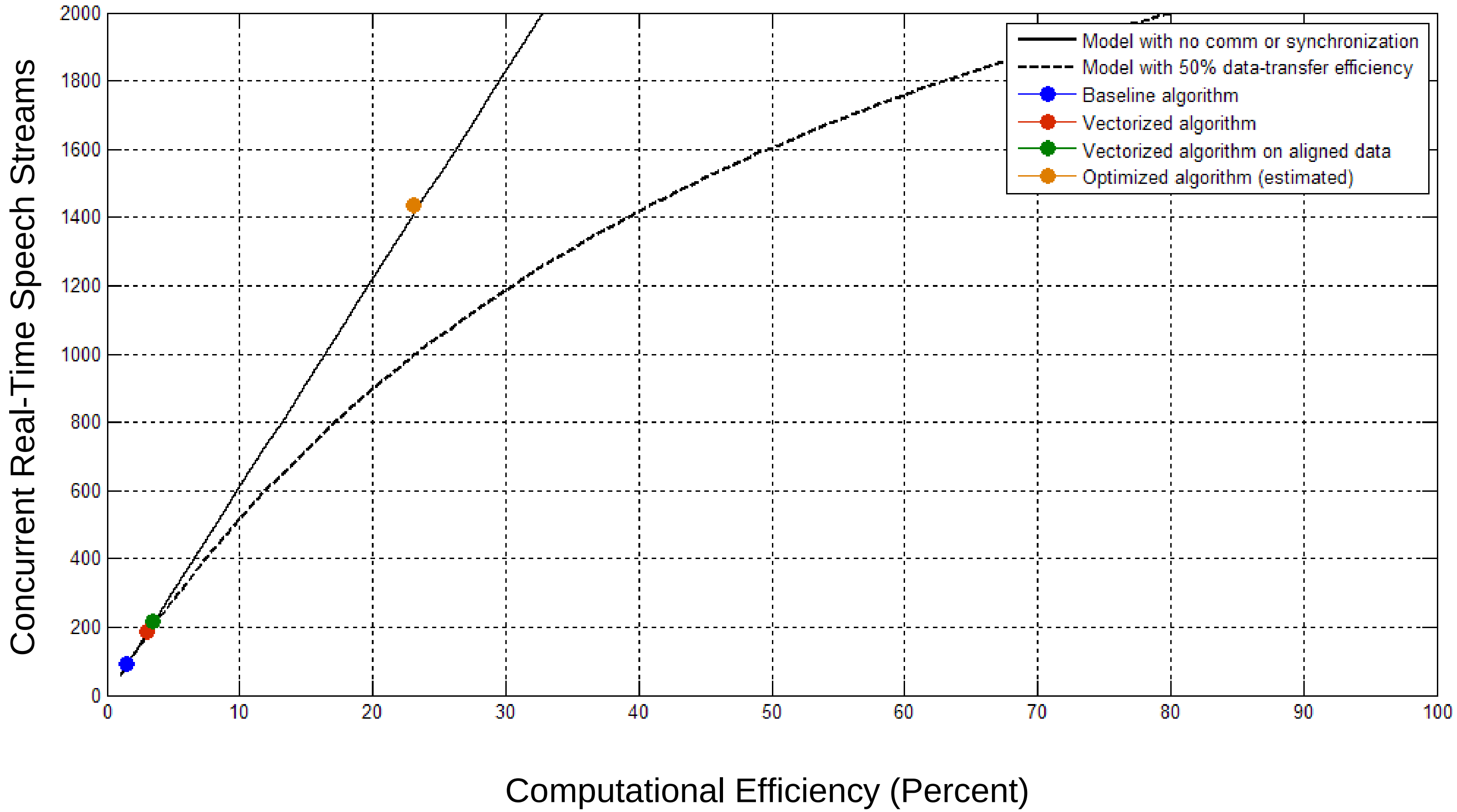
Universal Background Model



616KB for Entire UBM
77 KB Resides on Each SPE

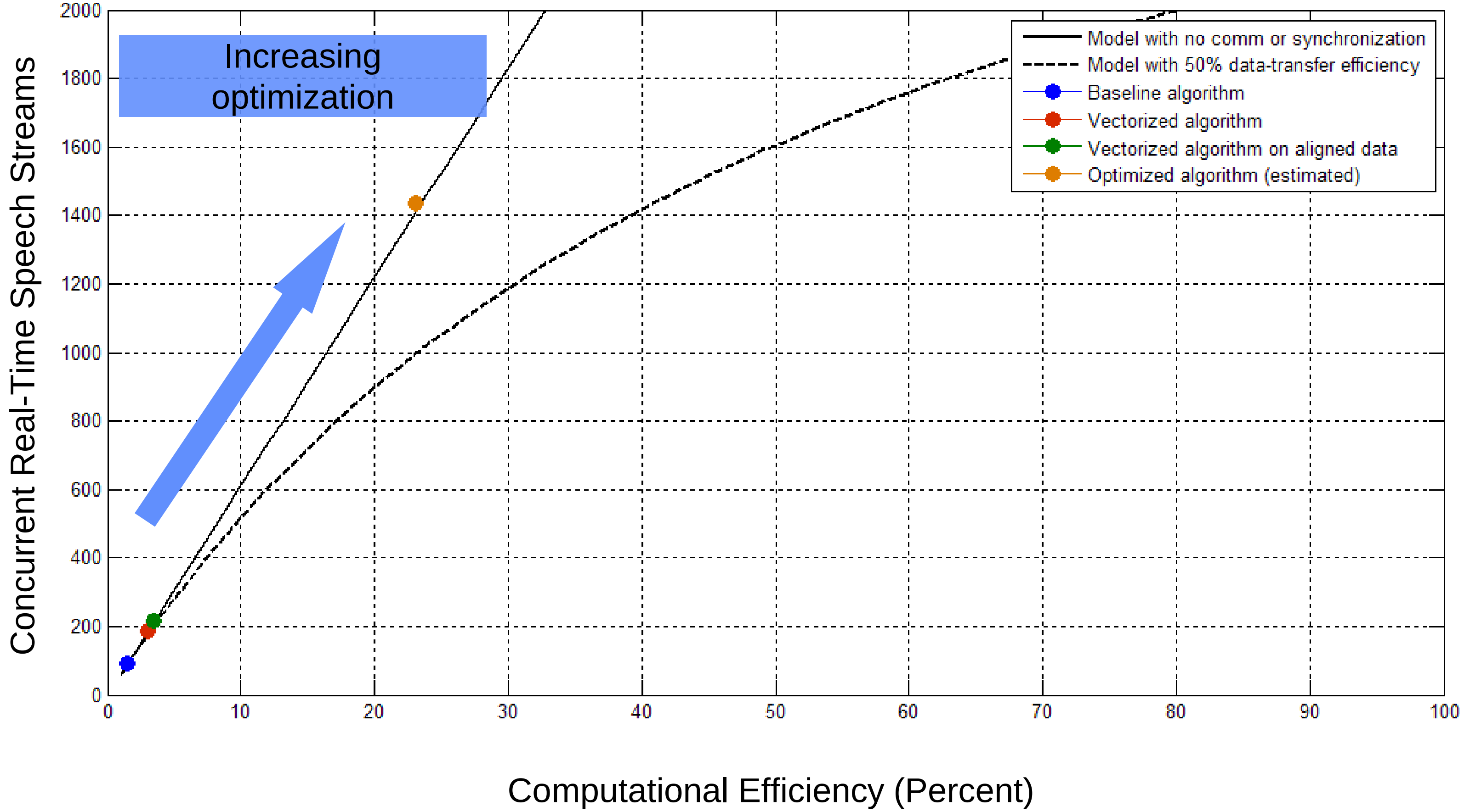
Performance Model

Simulation and Measurements



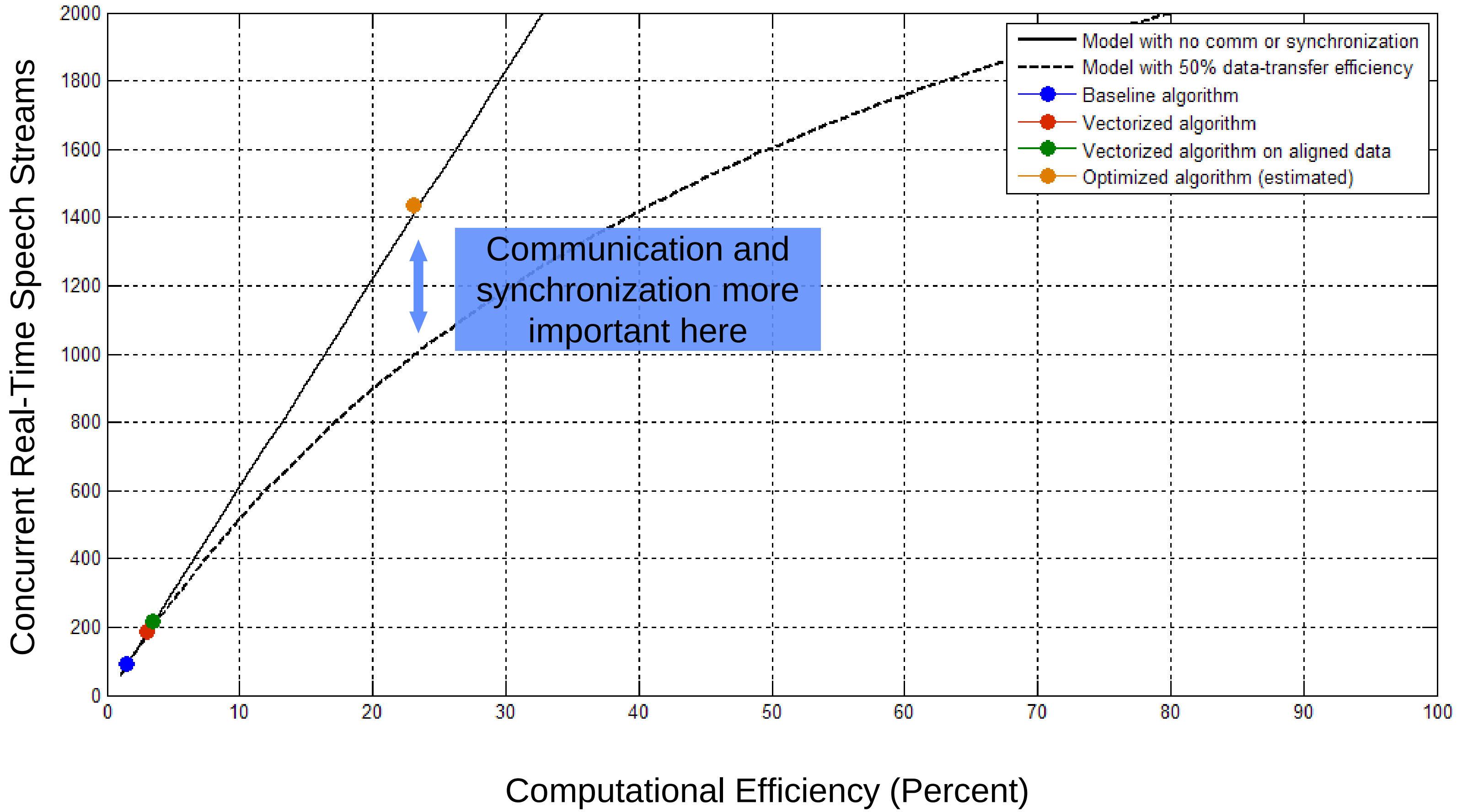
Performance Model

Simulation and Measurements



Performance Model

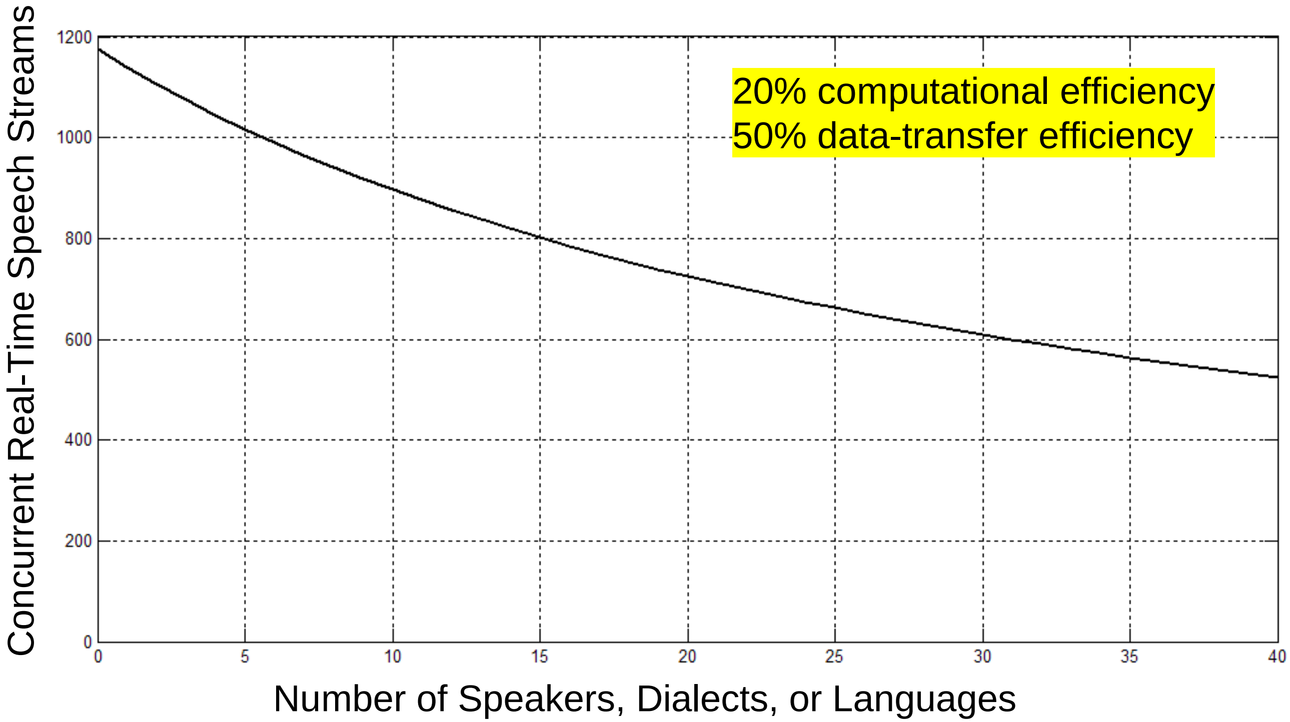
Simulation and Measurements



Performance Model

Simulation and Measurements

- **The effect of increasing the number of speakers, dialects, or languages (targets) was simulated**
  - **Changing the number of targets varies the amount of data sent to SPEs and the amount of calculation per SPE**



Outline

- Introduction
- Recognition for speech applications using GMMs
- Parallel implementation of the GMM
- Performance model
- Conclusions and future work

Conclusions

Summary

- Language, dialect, and speaker recognition systems are large in scale and will benefit from parallelization due to their need for high throughput

- GMM scoring is expensive both in terms of computing resources and memory

- We have designed and modeled an algorithm to perform GMM scoring in an efficient way
  - Preserving often-used data on the SPEs
  - Performing most calculations on the SPEs

Conclusions
Future Work

- Optimization and measurement of the full algorithm to validate the model
- Compare our system against other state-of-the-art serial and parallel approaches
  - Intel single processor
  - Intel multicore
  - Intel networked
  - Cell PPE
- Our results will become part of the PVTOL library

Acknowledgement

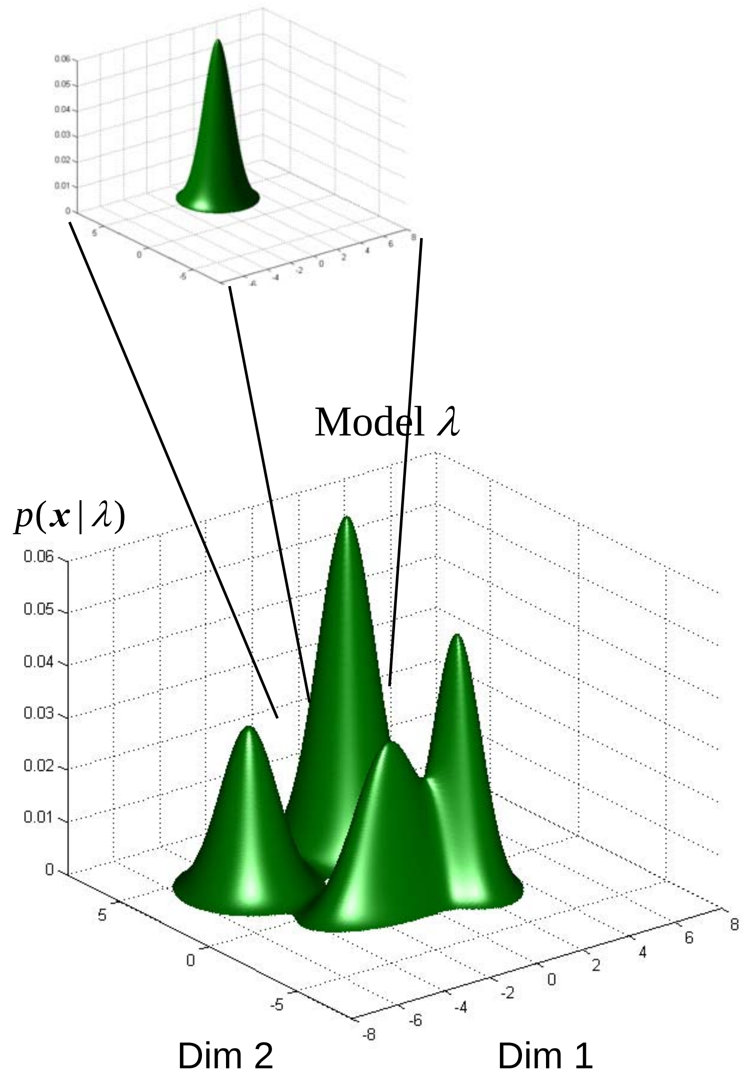
- Cliff Weinstein
- Joe Campbell
- Alan McCree
- Tom Quatieri
- Sharon Sacco

Backup

Gaussian Mixture Model
Equation

- **A Gaussian mixture model (GMM) represents features as the weighted sum of multiple Gaussian distributions**

- **Each Gaussian *state i* has a**
  - **Mean** $\boldsymbol{\mu}_i$
  - **Covariance** Σ_i
  - **Weight** w_i



$$p(\boldsymbol{x} \mid \lambda) = \sum_{i=1}^{M} \frac{w_i}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \exp\left(-\tfrac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right)$$