Vector Quantization, Gaussian Mixtures, and EM

Geoffrey Zweig
April 23, 2009
What is VQ?

- Representation of data in terms of codewords
- A data point is represented as the index of the nearest codeword
- In 2-D you end up storing one number, not two

Image from Wikipedia
Why is it Useful?

- Data compression
- Data transmission
- Discrete representation of data is convenient to work with:
  - Enumerated probability distributions over single events
  - Language models and discrete HMMs for sequences
K-Means Algorithm

- Common form of vector quantization
- Creates K centers
- Initialization:
  - Choose K distinct points at random for the first centers
- Repeat:
  - Assign each data point to the nearest center
  - Reset each center to the mean of the points assigned to it
- Stopping criteria can be:
  - an absolute number of iterations
  - or threshold on sum of all distances between centers and assigned points (total distortion)
Example of K-Means Algorithm

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Example of K-Means Algorithm

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Example of K-Means Algorithm

- http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Example of K-Means Algorithm

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Example of K-Means Algorithm

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Convergence: Lyapunov Condition

- Applies to iterative processes
- If you can identify a quantity $G$ that decreases at each iteration
- And which can’t go below a bound $B$
- Then the process will converge
VQ Convergence

- Consider the total “distortion”
- Sum of distances between points and their assigned centers

$$\sum_{i} (x_i - c(x_i))^2$$

Step 1: Assign each point to the nearest center

Defines $C(x_i)$ explicitly to minimize $(x_i - c(x_i))^2$

Since the contribution of each point individually to the distortion goes down, the total distortion must decrease.
Step 2: Re-estimate each center as the mean of its assigned points

- Consider what happens to one center $c$

\[
\frac{d}{dc} \sum_{i=1}^{N} (x_i - c)^2 = 0
\]

\[
\sum_{i=1}^{N} \frac{d}{dc} (x_i - c)^2 = 0
\]

Distortion contributed by each $c$ individually is minimized

\[
\sum_{i=1}^{N} 2(x_i - c) = 0
\]

$\Rightarrow$ The total distortion is minimized

\[
c = \frac{1}{N} \sum_{i=1}^{N} x_i
\]
Iteratively increases the number of codewords – 2,4,8,16...

Can be used to induce a tree structured quantizer

Initialize:

- Make one codeword in the center of everything
- Assign all the data to it

Repeat:

- Split each current codeword into two slightly different variants
- Do k-means with the current codewords
LBG Example

VQ: Some Things to be Aware Of
Speeding Up VQ with a Tree

Recursively partition the data as the tree is built.

Only follow one branch when finding a codeword after the tree is built.
Sub-band Coding

- Typical speech recognition features are MFCCs, Deltas and double-Deltas
- Historically a total of 39 dimensions
- Some quantization options:
  - One codebook for everything (e.g. 2048 centers)
  - One codebook for each (e.g. 1024 centers x 3)
  - One codebook for each, but energy and its derivatives get a separate codebook (e.g. 1024x3 + 256)
- The answer is empirically determined
- Multiple codebooks allow for many more distinct outcomes (the product of the codebook sizes)
  - But correlations between dimensions are lost
Further Extensions

- Quantize equal-length blocks of frames
  - Distance is sum of framewise distances
  - Contrasts with sub-band coding
    - Captures more correlation effects than individual frame quantization
    - Sacrifices combinatorial increase in possible outcomes
- Template codebook (Matrix quantization in Rabiner & Juang)
  - Distance is dynamic-time warp distance
  - Reestimation by averaging frames that have been matched to a center frame (why is this guaranteed to work?)
- Conditional codebooks
  - The codeword of the last input determined the codebook of the next
Gaussian Mixtures & EM
Gaussian Mixtures

- Codebook centers are gaussians
- An example may be assigned partially to a center
- A generative model
- Implies a data likelihood
Gaussian Refresher

\[ N(\mathbf{x}; \mu, \Sigma) = \pi^{-d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^t \Sigma^{-1} (x-\mu)\right) \]

- Parameterized by mean and covariance matrix
- Integral over all space is 1 (probability density function)
- Diagonal covariance matrix most common in speech
  - \( O(d) \) parameters rather than \( O(d^2) \)
Diagonal vs. Full Covariance Matrices

Diagonal covariance gaussians are axis-aligned
Full covariance gaussians need not be

Images from Wikipedia
Approximating Full Covariance with Diagonal Covariance gaussians

A sum of enough diagonal covariance gaussians can approximate a full covariance gaussian to arbitrary accuracy.
Maximum Likelihood Parameter Estimation (MLE) – Data Likelihood

\[
\log p_n(x \mid \Phi) = \sum_{k=1}^{n} \log p(x_k \mid \Phi)
\]

\[
= \sum_{k=1}^{n} \log \left( \frac{1}{\sqrt{2\pi\sigma}} \exp \left[ -\frac{(x_k - \mu)^2}{2\sigma^2} \right] \right)
\]

\[
= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{k=1}^{n} (x_k - \mu)^2
\]

(n 1-dimensional points)

From Acero et al. Chapter 3
MLE – Take the Derivative

\[
\frac{\partial}{\partial \mu} \log p_n(x | \Phi) = \sum_{k=1}^{n} \frac{1}{\sigma^2} (x_k - \mu)
\]

\[
\frac{\partial}{\partial \sigma^2} \log p_n(x | \Phi) = -\frac{n}{2\sigma^2} + \sum_{k=1}^{n} \frac{(x_k - \mu)^2}{2\sigma^4}
\]

Set it equal to 0 and solve

\[
\mu_{MLE} = \frac{1}{n} \sum_{k=1}^{n} x_k = E(x)
\]

\[
\sigma^2_{MLE} = \frac{1}{n} \sum_{k=1}^{n} (x_k - \mu_{MLE})^2 = E[(x - \mu_{MLE})^2]
\]

From Acero et al. Chapter 3
Why Do We Care About Gaussians

- A single gaussian is highly restricted
- But with enough gaussians you can model any probability distribution
  => A parametric modeling approach that becomes non-parametric
- And they are well understood in terms of
  - Parameter estimation
  - Computational complexity (and speedups)
  - Discriminative training
  - Adaptation to new data sets
K-Means for GMMs

- Same process as for VQ, but “soft” assignment
- Repeat:
  - Assign each data point to each gaussian with some weight
  - Re-estimate the gaussian centers using the weighted data assigned to each
MLE with GMMs

Where we are going:

- Parameter estimation will be as before
- But the $x_k$s below will be weighted by “degree of membership”
- And $n$ will be the sum of the weights

\[
\mu_{MLE} = \frac{1}{n} \sum_{k=1}^{n} x_k = E(x)
\]
\[
\sigma_{MLE}^2 = \frac{1}{n} \sum_{k=1}^{n} (x_k - \mu_{MLE})^2 = E\left[ (x - \mu_{MLE})^2 \right]
\]

From Acero et al. Chapter 3
Convergence of EM Process

Analysis will follow Sean Borman, “The Expectation Maximization Algorithm A Short Tutorial”

See also:
* Jeff Bilmes “A Gentle Tutorial on the EM Algorithm” and
* Acero et al. Chapter 4.

The data likelihood will go up at each iteration, analogous to distortion going down

\[
L(\theta) = \ln \mathcal{P}(X|\theta).
\]

\[
\mathcal{P}(X|\theta) = \sum_z \mathcal{P}(X|z, \theta)\mathcal{P}(z|\theta)
\]

Jensen’s Inequality

\[
\ln \sum_{i=1}^{n} \lambda_i x_i \geq \sum_{i=1}^{n} \lambda_i \ln(x_i).
\]

All \(\lambda\)s must be non-negative and sum to 1.
Change in Likelihood

\[
L(\theta) - L(\theta_n) = \ln \left( \sum_z \mathcal{P}(X|z, \theta) \mathcal{P}(z|\theta) \right) - \ln \mathcal{P}(X|\theta_n)
\]

\[
= \ln \left( \sum_z \mathcal{P}(X|z, \theta) \mathcal{P}(z|\theta) \cdot \frac{\mathcal{P}(z|X, \theta_n)}{\mathcal{P}(z|X, \theta_n)} \right) - \ln \mathcal{P}(X|\theta_n)
\]

\[
= \ln \left( \sum_z \mathcal{P}(z|X, \theta_n) \frac{\mathcal{P}(X|z, \theta) \mathcal{P}(z|\theta)}{\mathcal{P}(z|X, \theta_n)} \right) - \ln \mathcal{P}(X|\theta_n)
\]

\[
\geq \sum_z \mathcal{P}(z|X, \theta_n) \ln \left( \frac{\mathcal{P}(X|z, \theta) \mathcal{P}(z|\theta)}{\mathcal{P}(z|X, \theta_n)} \right) - \ln \mathcal{P}(X|\theta_n)
\]

\[
= \sum_z \mathcal{P}(z|X, \theta_n) \ln \left( \frac{\mathcal{P}(X|z, \theta) \mathcal{P}(z|\theta)}{\mathcal{P}(z|X, \theta_n) \mathcal{P}(X|\theta_n)} \right)
\]

\[
\Delta \triangleq \Delta(\theta|\theta_n).
\]

In going from Equation (12) to Equation (13) we made use of the fact that
\[
\sum_z \mathcal{P}(z|X, \theta_n) = 1
\]
so that \(\ln \mathcal{P}(X|\theta_n) = \sum_z \mathcal{P}(z|X, \theta_n) \ln \mathcal{P}(X|\theta_n)\) which allows the term \(\ln \mathcal{P}(X|\theta_n)\) to be brought into the summation.

Lower Bound on New Likelihood

We’ll work by increasing this lower bound. But will increasing a lower bound increase what we want?

\[ L(\theta) \geq L(\theta_n) + \Delta(\theta|\theta_n) \triangleq l(\theta|\theta_n) \]
Lower Bound Evaluated at Current Parameters is the Likelihood Itself!

\[
l(\theta_n|\theta_n) = L(\theta_n) + \Delta(\theta_n|\theta_n) \\
= L(\theta_n) + \sum_z \mathcal{P}(z|X, \theta_n) \ln \frac{\mathcal{P}(X|z, \theta_n) \mathcal{P}(z|\theta_n)}{\mathcal{P}(z|X, \theta_n) \mathcal{P}(X|\theta_n)} \\
= L(\theta_n) + \sum_z \mathcal{P}(z|X, \theta_n) \ln \frac{\mathcal{P}(X, z|\theta_n)}{\mathcal{P}(X, z|\theta_n)} \\
= L(\theta_n) + \sum_z \mathcal{P}(z|X, \theta_n) \ln 1 \\
= L(\theta_n),
\]
The Real Picture

Maximizing the Lower Bound on Likelihood

\[ \theta_{n+1} = \arg \max_{\theta} \{ l(\theta | \theta_n) \} \]

\[ = \arg \max_{\theta} \left\{ L(\theta_n) + \sum_z \mathcal{P}(z|X, \theta_n) \ln \frac{\mathcal{P}(X,z, \theta) \mathcal{P}(z|\theta)}{\mathcal{P}(X|\theta_n) \mathcal{P}(z|X, \theta_n)} \right\} \]

Now drop terms which are constant w.r.t. \( \theta \)

\[ = \arg \max_{\theta} \left\{ \sum_z \mathcal{P}(z|X, \theta_n) \ln \frac{\mathcal{P}(X,z, \theta) \mathcal{P}(z, \theta)}{\mathcal{P}(z|\theta) \mathcal{P}(\theta)} \right\} \]

\[ = \arg \max_{\theta} \left\{ \sum_z \mathcal{P}(z|X, \theta_n) \ln \frac{\mathcal{P}(X,z, \theta)}{\mathcal{P}(z, \theta) \mathcal{P}(\theta)} \right\} \]

\[ = \arg \max_{\theta} \left\{ \mathbb{E}_{Z|X, \theta_n} \{ \ln \mathcal{P}(X,z, \theta) \} \right\} \]

Prob. of \( z \) w.r.t current parameters

The Q Function

Full data likelihood w.r.t new params

What This Tells Us

- Compute the expected values of the hidden variables
- Assume the hidden variables are seen with these probabilities
- Compute a new set of parameters $\theta$ to optimize the complete data likelihood
  - The Q function is a function of $\theta$
  - It is maximized wrt $\theta$
- This is guaranteed to increase the likelihood
Application to GMMs

One data point, $K$ gaussians

$$p(y | \Phi) = \sum_{k=1}^{K} c_k p_k(y | \Phi_k) = \sum_{k=1}^{K} c_k N_k(y | \mu_k, \Sigma_k)$$

Posterior probability (count) of gaussian $k$ wrt data point $i$

$$\gamma_k = \frac{c_k p_k(y_i | \Phi_k)}{P(y_i | \Phi)}$$

Total number of points assigned to Gaussian $k$

$$\gamma_k = \sum_{i=1}^{N} \frac{c_k p_k(y_i | \Phi_k)}{P(y_i | \Phi)}$$

From Acero et al. Chapter 4
Application to GMMs (2)

New prior for Gaussian $k$

\[
\hat{c}_k = \frac{\gamma_k}{\sum_{k=1}^{K} \gamma_k} = \frac{\gamma_k}{N}
\]

Mean is posterior-weighted average of the points

\[
\hat{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_k^i y_i}{\sum_{i=1}^{N} \gamma_k^i} = \frac{\sum_{i=1}^{N} c_k p_k(y_i | \Phi_k) y_i}{\sum_{i=1}^{N} c_k p_k(y_i | \Phi_k) P(y_i | \Phi)}
\]

Variance also a weighted sum

\[
\hat{\Sigma}_k = \frac{\sum_{i=1}^{N} \gamma_k^i (y_i - \mu_k)(y_i - \mu_k)^t}{\sum_{i=1}^{N} \gamma_k^i} = \frac{\sum_{i=1}^{N} c_k p_k(y_i | \Phi_k)(y_i - \mu_k)(y_i - \mu_k)^t}{\sum_{i=1}^{N} c_k p_k(y_i | \Phi_k) P(y_i | \Phi)}
\]

From Acero et al. Chapter 4
Break

- Then:
- Advanced topics in GMMs
Fast Gaussian Computation

- Competition-grade systems may have close to 1M gaussians
- Typically features are extracted 100 times a second
- Evaluating and accumulating each dimension takes something like 2 additions and 2 multiplies
- 39 dimensions
- 100 million gaussian evaluations per second amounts to something like 15 billion ops/sec
- This is a problem for real-time or near real-time systems!
Some Options for Speeding Things Up

- **On-Demand Computation**
  - Only evaluate gaussians required by the search strategy
  - But: introduces linkage between search and gaussian computation, requires caching, and is complex

- **Dimension-wise pruning**
  - Likelihood computations involve sum of \((x-u)/\sigma)^2\) across dimensions – big number means low likelihood
  - Stop when you know the likelihood will be bad
  - But: limited benefit in practice

- **Hierarchical evaluation**
- **Cache optimization**
Hierarchical Evaluation

1. Evaluate the top level gaussians against a frame
2. Select the top N (e.g. 100)
3. Evaluate the “real” gaussians assigned to these top N
4. Assume everything else is zero
5. 20x speedup!

Cluster them into a few high-level gaussians (e.g. 2000)

The gaussians we need to evaluate
How to Cluster the Gaussians?

- K-Means of course!
- Some distance metrics:
  1. Euclidian distance between means
  2. KL-Divergence between a gaussian and the centroid
Cache Optimization

For each frame
  For each gaussian
    Do an evaluation

For each gaussian
  For each frame
    Do an evaluation

Gaussians have means *and* variances
A frame takes ½ the memory!
½ as many cache misses
Maybe twice the speed
Applicable to hierarchical evaluation too
Cache Optimization (2)

Re-order for locality
Low Memory Gaussian Computation

- Think circa 1990
- Dragon Dictate and IBM ViaVoice just introduced
- Think Intel 486
- 20MHz, 16MB RAM
- Memory was an issue!

- What to do?
Low Memory Gaussians (2)

- Break gaussians into bands
  - Each e.g. 2 dimensions
- Cluster all the samples in each band
  - Analogous to clustering the gaussians in the first place
- Diagonal covariance gaussians decompose into sum of bands
- Represent a gaussian as the sum of its bands

![Diagram](image)
Consider 1-Dimensional Quantization

The quantized mean/variance of the d-th dimension of the j-th gaussian is:

\[
\mu_d^{q(j)}, \sigma_d^{q(j)}
\]

\[
\log N^j(x; \mu, \Sigma) \propto D \log 2\pi + \sum_d \log \sigma_d^j + \sum_d (x_d - \mu_d^j)^2 (\sigma_d^j)^{-2}
\]

\[
\approx D \log 2\pi + \sum_d \log \sigma_d^{q(j)} + \sum_d (x_d - \mu_d^{q(j)})^2 (\sigma_d^{q(j)})^{-2}
\]

Compute: \[
\log \sigma_d^{q(j)} + (x_d - \mu_d^{q(j)})^2 (\sigma_d^{q(j)})^{-2}
\]

Once for each codeword and re-use across gaussians
Memory Requirements

- Say 40 dimensions and bands are 2-dimensions
- Quantize to 256 codewords per band
- Each gaussian is now represented 20 bytes
- Used to be 40*2*4 = 320 (assuming floats)
- Factor of 16 reduction
Compute requirements

- Evaluate $256 \times 20 = 5120$ 2-dimensional gaussians
- Add 20 numbers to get the score for a “real” gaussian
- Repeatedly access the 5120 atomic numbers
  - Good for cache!
Further Speedups – Two References

• Aiyer, Gales & Picheny, “Rapid Likelihood Computation of Subspace Clustered Gaussian Components” (2000)
  • Many gaussians use common sets of codewords
  • Redundant computation
  • Can be optimized with compiler technology for evaluating common subexpressions once only
  • Numerous tricks for efficient organization of complete recognizer
Full Covariance Matrices

\[ N(\xi; \mu, \Sigma) = \ell \pi^{-n/2} |\Sigma|^{-1/2} \exp \left(-\frac{1}{2}(x-\mu)' \Sigma^{-1} (x-\mu) \right) \]

- When $\Sigma^{-1}$ is not diagonal
  - Number of parameters is $O(D^2)$ not $O(D)$
  - Need more data to estimate the parameters
  - Evaluation is much slower
  - Band quantization doesn’t work
  - Adaptation methods are more complex
- Nevertheless, people sometimes see improvements
  - EMLLT is an interesting compromise
Inverse covariance matrix is sum of outer products of basis vectors
• Can also think of as sum of basis matrices
• Basis vectors shared across all gaussians
  • Potentially many fewer covariance parameters – just $D$ per gaussian
  • Plus the pool of basis vectors

See Olsen & Gopinath, “Modeling Inverse Covariance Matrices by Basis Expansion”
Some EMLLT Results

d is the vector dimensionality
From Olsen & Gopinath,
“Modeling Inverse Covariance Matrices by Basis Expansion”

<table>
<thead>
<tr>
<th></th>
<th>WER</th>
<th></th>
<th>WER</th>
<th></th>
<th>D</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal</td>
<td></td>
<td>MLLT</td>
<td></td>
<td>EMLLT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nGauss</td>
<td></td>
<td>nGauss</td>
<td></td>
<td>nGauss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10253</td>
<td>3.14%</td>
<td>10253</td>
<td>2.84%</td>
<td>10253</td>
<td>2.54%</td>
<td></td>
</tr>
<tr>
<td>17028</td>
<td>3.08%</td>
<td>17028</td>
<td>2.74%</td>
<td>10253</td>
<td>2.34%</td>
<td></td>
</tr>
<tr>
<td>26460</td>
<td>3.01%</td>
<td>26460</td>
<td>2.58%</td>
<td>10253</td>
<td>2.15%</td>
<td></td>
</tr>
<tr>
<td>46500</td>
<td>2.84%</td>
<td>46500</td>
<td>2.50%</td>
<td>10253</td>
<td>2.04%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2d</td>
<td></td>
<td>14d</td>
<td></td>
<td>2.11%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4d</td>
<td></td>
<td>20d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adaptation

Old data modeled by some gaussians

New data

How should we update our estimate of what the gaussians are?
Option 1: Replace the old data

Old data modeled by some gaussians

New data, New gaussians
Option 2: Add the Data (MAP Adaptation)

Old data modeled by some gaussians

New data

Re-estimate gaussians, combining old and new data, possibly with a weighting factor

See, e.g. Gauvain & Lee, Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains
Option 3: Transform the Means

\[ \mu' = A(1 \mu^T)^T \]

- New mean is linear transformation of old
- An offset is added to the old mean as well
- Transformation matrix chosen to maximize the likelihood of the adaptation data under the transformed model
- One transformation (e.g. 39x39) shared by many gaussians (e.g. 1000s)
- See, e.g., Leggetter & Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models”
- Similar transforms possible for covariance matrix
Old data modeled by some gaussians

New data

New means are a linear transform of the old ones
Tying it All Together: Phone Probabilities

- **Want:** \( \arg\max_q P(q \mid y) = \arg\max_q P(q)P(y \mid q) \)
- **Need to model** \( P(y \mid \text{phone } q) \)
- **Discrete (VQ) probabilities**
- **Continuous (GMM) probabilities**
- **Semi-Continuous probabilities**

Is it an /ah/?
Is it an /eh/?
Is it a /p/?
The Discrete Approach

- Vector-quantize the feature vectors
  - Every 10ms or so
- \( P(y_t | /ah/) = P(27 | /ah/) \)
  - Learned by counting examples
  - Covered in HMM lecture

Note:
- Spectral slices should change
- MFCCs would normally be used
Continuous Probabilities

- Each phone has its own gaussian mixture

\[ p(y \mid \Phi) = \sum_{k=1}^{K} c_k p_k(y \mid \Phi_k) = \sum_{k=1}^{K} c_k N_k(y \mid \mu_k, \Sigma_k) \]

Image from http://oregonstate.edu/~hohenlop/Gaussianmix.jpg
Semi-Continuous Probabilities

- All models share the same gaussians
- Models differ only in the weight assigned to each
- Continuous gaussian models are a special case
  - With lots of zeros as coefficients
- Not much used anymore in ASR
Homework

• Write a VQ program for 2-dimensional data
  • First use Euclidean distortion $D(x,y)$ between points $x,y$ (eq’n. 4.77 of Acero, et al.), squared Euclidean distance

• Use the provided “points” file as input

• Make a plot of the input

• Fit 1, 2, 4, 8 and 16 centroids to the data
  • Plot the centers on top of the data

• Now use log distance
  • $\log(1 + D(x,y))$

• Can you find an analytical update, guaranteed to reduce distortion?

• Find an update that is guaranteed to reduce distortion at each iteration (analytical or not)

• Fit 1, 2, 4, 8 and 16 centroids to the data
  • Plot the centers on top of the data
  • Plot the Euclidean and Log-distance centroids together

• Finally, train a mixture of 1, 2, 4, 8 and 16 gaussians with these points (using either K-means or LBG). Be sure to adjust the variances.
  • Plot the positions of the centers.

• Turn in all 4 plots for 4 centers

• Turn in a printout of your program

• Turn in a printout of the total distortion after each iteration as the program runs
Project Reminder

- Please think about your projects!
- For Speech Recognition / Language ID / Speaker ID, please contact me
  - After class
  - By email gzweig@microsoft.com
- On 5/7 I’d like to meet with everyone doing a relevant project.