SSE Vectorization
EM Gaussian Mixture Estimation

Tyler Karrels
ECE 734 – Spring 2010
Streaming SIMD Extensions (SSE)

- Single Instruction Multiple Data
- Single Precision floating point (MMX = integer)
- 8 x 128bit registers (XMM0-XMM7)

![Diagram of packed single-FP data]

Packed Single-FP
SIMD Execution Model

- 64 or 128bit memory access
- 1 vector operation = 4 FP operations
SIMD FP Intrinsics

• High level language access to SSE registers
• Leverages C/C++ languages and compiler technology
• Avoids assembly coding
• __m128 data type
SIMD Instructions

- ADDPS – add
- DIVPS – divide
- RSQRTPS – compute $1/\sqrt{x}$
- SUBPS – subtract
- MULSS – multiply
Mixture of Gaussian

• Exactly as it sounds

\[ p(x) = \sum_{n=1}^{K} w_n G_n(x) \]

• \( G_n(x) \) is the Gaussian kernel (density function)
EM Algorithm

\[
\text{init\_em}(n, X, k, W, M, V);
\]

while( change in likelihood > tolerance) 
    & (niter <= maxiter))
{
    expectation(E, n, X, k, W, M, V);
    maximization(E, n, X, k, W, M, V);
    Lo = Ln;
    Ln = likelihood(n, X, k, W, M, V);
    niter++;
}
E-Step

for(int i = 0; i < n; i++){
    row_sum = 0;
    for(int j = 0; j < k; j++){
        *(E+j+i*k) = *(W+j)*(1/sqrt(2*PI**(V+j)))*exp(-1/(2***(V+j)))*pow((*(X+i)-*(M+j)),2));
        row_sum += *(E+j+i*k);
    }
    for(int j = 0; j < k; j++){
        *(E+j+i*k) = *(E+j+i*k)/row_sum;
    }
}

• Change pow(x,2) to mulss
• Change 1/sqrt(x) to rsqrtps
• Change row_sum += *(E+j+i*k) to a series of addps, 4 at a time
• Unfold outerloop 4x
Preliminary Results

• 900 1-d data points drawn from 3 Gaussians
• 2.4GHz Intel Quad-core CPU, 64-bit OS
• Average runtime over 100 trials....
  – Matlab: 31.2125 [s] (EM_GM.m)
  3.5 [ms]  (optimized version using Matlab libraries)
  – C++ without SSE: 272.22 [ms] (EM_GM recoded)
What’s Next

• Recode the C++ EM_GM version using SSE SIMD instructions
• Probably will not beat the 3.5 [ms] mark
• Expect a speedup of 2-3x = 90-135 [ms]