1 Distributed Algorithm Design

We use a method called MapReduce to distribute a learning algorithm across multiple processors. It can be used for any learning algorithm that has a training objective summing over data points. In this case, the data set is split up into batches. "Map" refers to distributing a process across multiple processors, and "Reduce" refers to merging the results back together.

1.1 MapReduce for Batch GD

1. Split data set D into $D_1, D_2, .., D_m$
2. (Across multiple processors) Map:
   - $temp_m := \sum_{x,t \in D_m} (t - wx)x$
3. Reduce:
   - $w \leftarrow w + \eta(\sum^M temp_m)$

1.2 MapReduce for K-Means

Split data set D into $D_1, D_2, .., D_m$ and randomly initialise μ

Map:

1. set temporary variables $\hat{\mu}, \hat{n} = 0$
2. for $n \in 1 : |D_m|$:
   - $k* := argmin_k ||x_n - \mu_k||$
   - $\hat{\mu}_{k*,m} := \hat{\mu}_{k*,m} x$
   - $\hat{n}_{k*,m} := \hat{n}_{k*,m} + 1$

Reduce:
for $k \in K$:

- $\mu_k := \frac{\sum^M \hat{\mu}_{k,m}}{\sum^M \hat{n}_{k,m}}$

1.3 MapReduce for GMM

Split data set D into $D_1, D_2, .., D_m$ and initialise μ, Σ, ϕ

Map:

- set temporary variables $\hat{\mu}, \hat{\Sigma}, \hat{N} = 0$
- for $n \in 1 : |D_m|$:
  - For $k \in K$:
    * $\gamma(Z_{nk}) = \frac{\phi_k N(x_n;\theta_k)}{\sum^J \phi_j N(x_n;\theta_j)}$
    * $\hat{\mu}_{k,m} := \hat{\mu}_{k,m} + \gamma(Z_{nk})x_n$
    * $\hat{\Sigma}_{k,m} := \hat{\Sigma}_{k,m} + \gamma(Z_{nk})x_n \cdot x_n^T$
    * $\hat{N}_{k,m} := \hat{N}_{k,m} + \gamma(Z_{nk})$

Reduce:

- For $k \in K$:
  - $\mu_k := \frac{\sum^M \hat{\mu}_{k,m}}{\sum^M \hat{n}_{k,m}}$
  - $\Sigma_k := \frac{\sum^M \hat{\Sigma}_{k,m}}{\sum^M \hat{n}_{k,m}} - \mu_k \cdot \mu_k^T$
  - $\mu_k := \frac{\sum^M \hat{\mu}_{k,m}}{\sum^M \hat{n}_{k,m}}$