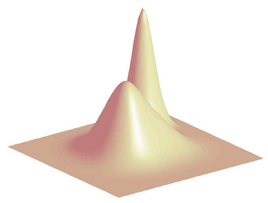
Université de Montréal

Gaussian Mixtures with Missing Data: an Efficient EM Training Algorithm

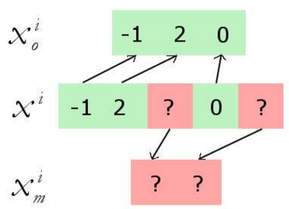
Olivier Delalleau
Aaron Courville
Yoshua Bengio

Outlook

- A mixture of Gaussians is a standard probabilistic model for high dimensional data
- It is straightforward to handle missing values in the classic Expectation-Maximization (EM) training algorithm...
- But to do it efficiently with many missing values, you must read this poster!



EM Algorithm with Missing Values



1. **E-Step**: Compute posteriors

$$p_{ij} = P(G = j|X_o = x_o^i) = \frac{P(X_o = x_o^i|G = j)}{\sum_{k=1}^{M} P(X_o = x_o^i|G = k)}$$

2. **M-Step**: For each Gaussian j:
   - Fill-in missing values by setting

$$x_m^i \leftarrow E[X_m|X_o = x_o^i, G = j]$$

   - Update mean and covariance

$$\mu_j = \frac{\sum_{i=1}^{n} p_{ij} x^i}{\sum_{i=1}^{n} p_{ij}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} p_{ij}(x^i - \mu_j)(x^i - \mu_j))^T}{\sum_{i=1}^{n} p_{ij}} + C_j$$

EM Computational Bottleneck

At each iteration of EM, for sample x^i with n_o observed and n_m missing variables, the main computational cost consists in:

1. the computation of p_{ij}: $O(n_o^3)$
2. the computation of its contribution to C_j: $O(n_m^3)$ (roughly)

$\Rightarrow$ cubic in the dimension!

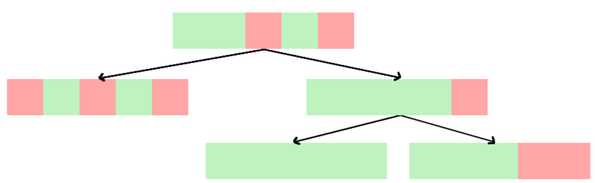
- **Basic observation:** two samples having the same *missing pattern* share the same expensive computations

x^i	-1	2	?	0	?
x^j	3	1	?	-1	?

- **This is not enough:** there may be (almost) as many different missing patterns as training samples!

Proposed Algorithm

Desired quantities for sample x^j can be obtained by *updating* quantities previously computed for another sample x^i. This update is cheap as long as both samples have similar missing patterns $\Rightarrow$ speed-up on the order of $O(n_m/n_d)$ with n_d the number of differences between missing patterns. The optimal path is a minimum spanning tree over missing patterns:



1. Compute minimum spanning tree on the graph of unique missing patterns (edge weight = cost of update)
2. Use K-means to initialize mixture means
3. Use empirical covariance in each cluster to initialize covariances (e.g. imputing missing values with cluster mean)
4. Perform EM, where at each iteration we go through the training set traversing the tree obtained in step 1, in order to perform computationally efficient updates

Experiment: Missing Values Imputation

- Randomly set as missing 5x5 patches in MNIST digits database (28x28 images)

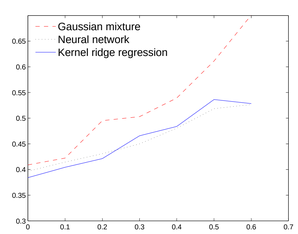
- For each class (digit), train a mixture of Gaussians
- Use the model with best NLL to impute missing values

- Compare speed with naive EM

$\Rightarrow$ speedup from 8 to 20 (architecture-dependent)

Combining Generative and Discriminative Models

- *Abalone* dataset (regression task) with a various proportion of missing values being added
- Train a mixture of Gaussians to model the data (including the target value)
- Test error when varying the proportion of missing values: the mixture directly used as a regressor does not work as well as a neural network or kernel ridge regressor trained **with missing values imputed by the mixture**



- Test error with neural network: imputing the Gaussian mixture's conditional mean works better than nearest neighbor or global mean imputation