Integrating Constraints and Metric Learning in Semi-Supervised Clustering

M. Bilenko, S. Basu, R.J.Mooney

Presentor: Lei Tang
Arizona State University

Machine Learning Seminar

Semi-supervised Clustering

1. Constrained-based method
   - *Seeded KMeans*, *Constrained KMeans* given partial label information.
   - *COP KMeans* given pairwise constraint(must-link, cannot-link)
2. Metric-based method
   - Learn a metric to satisfy the constraint, such that the data of the same cluster gets closer, whereas data of different clusters gets further away

Limitations

- Previous metric learning excludes unlabeled data during metric training.
- A single distance metric is used for all clusterings, forcing them to have the same shape.

Semi-supervised Clustering

1. Constrained-based method
    - *Seeded KMeans*, *Constrained KMeans* given partial label information.
    - *COP KMeans* given pairwise constraint(must-link, cannot-link)
2. Metric-based method
    - Learn a metric to satisfy the constraint, such that the data of the same cluster gets closer, whereas data of different clusters gets further away

Limitations

- Previous metric learning excludes unlabeled data during metric training.
- A single distance metric is used for all clusterings, forcing them to have the same shape.

Constrait-based method

- K-means clustering:

$$Minimize \quad \sum_{x_i \in \mathcal{X}} ||x_i - \mu_{l_i}||^2$$

- Semi-supervised clustering with constraints

$$Minimize \underbrace{\sum_{x_i \in \mathcal{X}} ||x_i - \mu_{l_i}||^2}_{\text{Typical k-means}} + \underbrace{\sum_{(x_i, x_j) \in \mathcal{M}} w_{ij} \mathbf{1}[l_i \neq l_j]}_{\text{must-link}} + \underbrace{\sum_{(x_i, x_j) \in \mathcal{C}} \bar{w}_{ij} \mathbf{1}[l_i = l_j]}_{\text{cannot-link}}$$

Constrait-based method

- K-means clustering:

$$Minimize \quad \sum_{x_i \in \mathcal{X}} ||x_i - \mu_{l_i}||^2$$

- Semi-supervised clustering with constraints

$$Minimize \underbrace{\sum_{x_i \in \mathcal{X}} ||x_i - \mu_{l_i}||^2}_{\text{Typical k-means}} + \underbrace{\sum_{(x_i, x_j) \in \mathcal{M}} w_{ij} \mathbf{1}[l_i \neq l_j]}_{\text{must-link}} + \underbrace{\sum_{(x_i, x_j) \in \mathcal{C}} \bar{w}_{ij} \mathbf{1}[l_i = l_j]}_{\text{cannot-link}}$$

Metric-based Method

- Euclidean distance:

$$||x_i - x_j|| = \sqrt{(x_i - x_j)^T (x_i - x_j)}$$

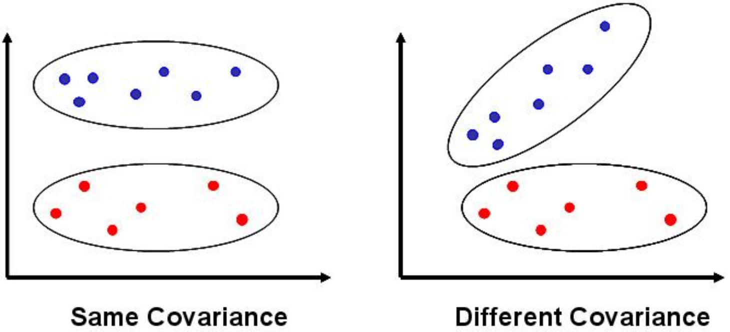
- Mahalanobis distance:

$$||x_i - x_j||_{\mathbf{A}} = \sqrt{(x_i - x_j)^T \mathbf{A} (x_i - x_j)}$$

  where **A** is a covariance matrix.
- $\mathbf{A} \succeq 0$
- If a **A** is used for calculate distance, then each cluster is modeled as a multivariate Gaussian distribution with covariance $\mathbf{A}^{-1}$.

Clustering with different shape

What if the shape of clusters are different?

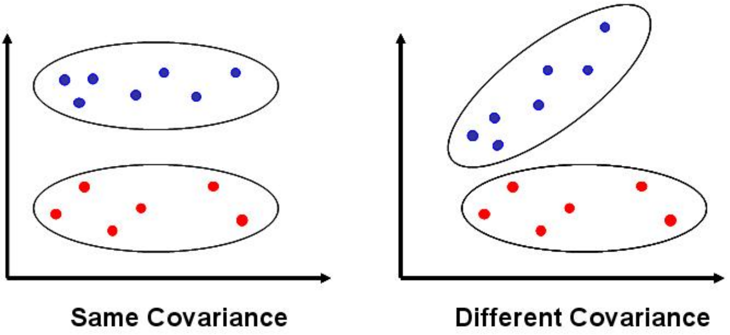


- Use different **A** for each cluster(Assign different covariance).
- To Maximize the likelihood boils down to :

$$Minimize \quad \sum_{x_i \in \mathcal{X}} \left(||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i}) \right)$$

Clustering with different shape

What if the shape of clusters are different?



- Use different **A** for each cluster(Assign different covariance).
- To Maximize the likelihood boils down to :

$$\textit{Minimize} \quad \sum_{x_i \in \mathcal{X}} \left(||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i}) \right)$$

Combine Constraints and Metric Learning

$$\begin{aligned} \textit{Minimize} \quad & \underbrace{\sum_{x_i \in \mathcal{X}} [||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i})]}_{\text{Metric Learning}} \\ + \quad & \underbrace{\sum_{(x_i,x_j) \in \mathcal{M}} w_{ij} \mathbf{1}[l_i \neq l_j] + \sum_{(x_i,x_j) \in \mathcal{C}} \bar{w}_{ij} \mathbf{1}[l_i = l_j]}_{\text{Constraints}} \end{aligned}$$

Intuitively, the penality w_{ij} and $\bar{w}_{ij}$ should be based on distance of two data points.

$$\begin{aligned} \textit{Minimize} \quad & \sum_{x_i \in \mathcal{X}} [||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i})] \\ + \quad & \sum_{(x_i,x_j) \in \mathcal{M}} f_M(x_i, x_j) \mathbf{1}[l_i \neq l_j] + \sum_{(x_i,x_j) \in \mathcal{C}} f_c(x_i, x_j) \mathbf{1}[l_i = l_j] \end{aligned}$$

Combine Constraints and Metric Learning

$$Minimize \quad \underbrace{\sum_{x_i \in \mathcal{X}} [||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i})]}_{\text{Metric Learning}}$$

$$+ \quad \underbrace{\sum_{(x_i,x_j)\in\mathcal{M}} w_{ij}\mathbf{1}[l_i \neq l_j] + \sum_{(x_i,x_j)\in\mathcal{C}} \bar{w}_{ij}\mathbf{1}[l_i = l_j]}_{\text{Constraints}}$$

Intuitively, the penality w_{ij} and $\bar{w}_{ij}$ should be based on distance of two data points.

$$Minimize \quad \sum_{x_i \in \mathcal{X}} [||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det\mathbf{A}_{l_i})]$$

$$+ \quad \sum_{(x_i,x_j)\in\mathcal{M}} f_M(x_i,x_j)\mathbf{1}[l_i \neq l_j] + \sum_{(x_i,x_j)\in\mathcal{C}} f_c(x_i,x_j)\mathbf{1}[l_i = l_j]$$

Penality based on distance

- Must-link: Violations means data belongs to different cluster.

$$f_M(x_i, x_j) = \underbrace{\frac{1}{2}(||x_i - x_j||^2_{A_{l_i}} + ||x_i - x_j||^2_{A_{l_j}})}_{Average}$$

The further away two data are, the more penality.

- Cannot-link: Violations means data belongs to the same cluster.

$$f_C(x_i, x_j) = \underbrace{||x'_{l_i} - x''_{l_i}||^2_{\mathbf{A}_{l_i}}}_{\text{Maximum distant points}} - ||x_i - x_j||^2_{\mathbf{A}_{l_i}}$$

The closer two data are, the more penality.

Penality based on distance

- Must-link: Violations means data belongs to different cluster.

$$f_M(x_i, x_j) = \underbrace{\frac{1}{2}(||x_i - x_j||^2_{A_{l_i}} + ||x_i - x_j||^2_{A_{l_j}})}_{Average}$$

  The further away two data are, the more penality.

- Cannot-link: Violations means data belongs to the same cluster.

$$f_C(x_i, x_j) = \underbrace{||x'_{l_i} - x''_{l_i}||^2_{\mathbf{A}_{l_i}}}_{\text{Maximum distant points}} - ||x_i - x_j||^2_{\mathbf{A}_{l_i}}$$

  The closer two data are, the more penality.

Metric pairwise constrained K-means(MPCK)

General Framework of MPCK algorithm based on EM

- Initialize clusters
- Repeat until convergence:
  - Assign Cluster to minimize the objective goal.
  - Estimate the mean
  - Update the metric

Difference with k-means

- Cluster assignment takes constraint into consideration.
- The metric is updated in each round.

Metric pairwise constrained K-means(MPCK)

General Framework of MPCK algorithm based on EM

- Initialize clusters
- Repeat until convergence:
  - Assign Cluster to minimize the objective goal.
  - Estimate the mean
  - Update the metric

Difference with k-means

- Cluster assignment takes constraint into consideration.
- The metric is updated in each round.

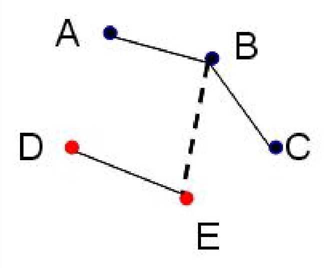
Initialization

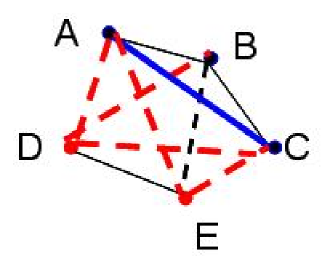
Basic idea

- Construct traversive closure of the must-link
- Choose the mean of each component as the seed.
- Extend the sets of must-link and cannot-link.

Construct traversive closure of the must-link

Must-link: {AB, BC, DE}; Cannot link: {BE};





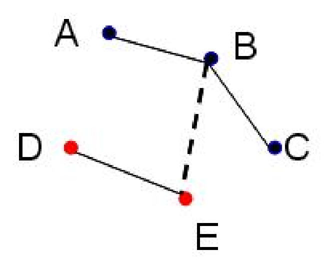
Initialization

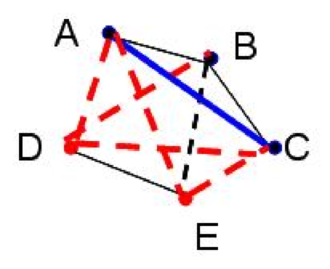
Basic idea

- Construct traversive closure of the must-link
- Choose the mean of each component as the seed.
- Extend the sets of must-link and cannot-link.

Construct traversive closure of the must-link

Must-link: {AB, BC, DE}; Cannot link: {BE};





Cluster Assignment

1. Randomly re-order the data points
2. Assign each data point to a cluster that minimize the objective function:

$$\mathit{Minimize} \qquad \mathcal{J} = \sum_{x_i \in \mathcal{X}} [||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det \mathbf{A}_{l_i})]$$
$$+ \sum_{(x_i, x_j) \in \mathcal{M}} f_M(x_i, x_j) \mathbf{1}[l_i \neq l_j] + \sum_{(x_i, x_j) \in \mathcal{C}} f_c(x_i, x_j) \mathbf{1}[l_i = l_j]$$

Update the metric

1. Update the centroid of each cluster
2. Update the distance metric of each cluster; Take the derivative of the goal function and set it to 0 to get the new metric:

$$
\begin{aligned}
\mathbf{A}_h \quad = \quad |\mathcal{X}_h| \Bigg\{ & \sum_{x_i \in \mathcal{X}_h} (x_i - \mu_i)(x_i - \mu_i)^T \\
& + \sum_{(x_i, x_j) \in \mathcal{M}_h} \frac{1}{2} w_{ij} (x_i - x_j)(x_i - x_j)^T \mathbf{1}[l_i \neq l_j])
\end{aligned}
$$

$$
+ \sum_{(x_i, x_j) \in \mathcal{C}_h} \bar{w}_{ij} \left((x'_h - x''_h)(x'_h - x''_h)^T - (x_i - x_j)(x_i - x_j)^T \right) \mathbf{1}[l_i = l_j] \Bigg\}^{-1}
$$

Some issues

1. Singularity: If the sum is singular, Set $\mathbf{A}_h^{-1} = \mathbf{A}_h^{-1} + \epsilon tr(\mathbf{A}_h^{-1})\mathbf{I}$ to ensure nonsiguarity.
2. Semi-positive definiteness: If $\mathbf{A}_h$ is negative definite, project it into set $C = \{\mathbf{A} : \mathbf{A} \succeq 0\}$ by setting negative eigenvalues to 0.
3. Computational cost: Use diagonal matrix. Or the same distance metric for all clusters.
4. Convergence: Theoretically, each step reduce the objective goal. But if singularity and semi-positive definiteness are involved, the algorithm might not converge in theory. Anyhow, it works fine in reality.

Some issues

1. Singularity: If the sum is singular, Set $\mathbf{A}_h^{-1} = \mathbf{A}_h^{-1} + \epsilon tr(\mathbf{A}_h^{-1})\mathbf{I}$ to ensure nonsiguarity.
2. Semi-positive definiteness: If $\mathbf{A}_h$ is negative definite, project it into set $C = \{\mathbf{A} : \mathbf{A} \succeq 0\}$ by setting negative eigenvalues to 0.
3. Computational cost: Use diagonal matrix. Or the same distance metric for all clusters.
4. Convergence: Theoretically, each step reduce the objective goal. But if singularity and semi-positive definiteness are involved, the algorithm might not converge in theory. Anyhow, it works fine in reality.

Some issues

1. Singularity: If the sum is singular, Set $\mathbf{A}_h^{-1} = \mathbf{A}_h^{-1} + \epsilon tr(\mathbf{A}_h^{-1})\mathbf{I}$ to ensure nonsiguarity.
2. Semi-positive definiteness: If $\mathbf{A}_h$ is negative definite, project it into set $C = \{\mathbf{A} : \mathbf{A} \succeq 0\}$ by setting negative eigenvalues to 0.
3. Computational cost: Use diagonal matrix. Or the same distance metric for all clusters.
4. Convergence: Theoretically, each step reduce the objective goal. But if singularity and semi-positive definiteness are involved, the algorithm might not converge in theory. Anyhow, it works fine in reality.

Some issues

1. Singularity: If the sum is singular, Set $\mathbf{A}_h^{-1} = \mathbf{A}_h^{-1} + \epsilon tr(\mathbf{A}_h^{-1})\mathbf{I}$ to ensure nonsiguarity.
2. Semi-positive definiteness: If $\mathbf{A}_h$ is negative definite, project it into set $C = \{\mathbf{A} : \mathbf{A} \succeq 0\}$ by setting negative eigenvalues to 0.
3. Computational cost: Use diagonal matrix. Or the same distance metric for all clusters.
4. Convergence: Theoretically, each step reduce the objective goal. But if singularity and semi-positive definiteness are involved, the algorithm might not converge in theory. Anyhow, it works fine in reality.

Experiment Results(1)

A single diagonal matrix is used.

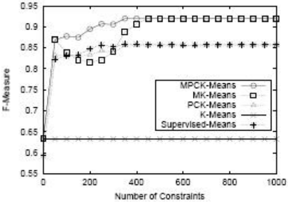


Figure 2. *Iris*: ablations

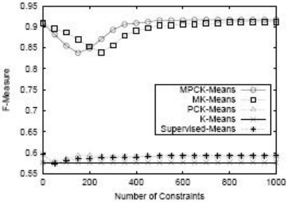


Figure 3. *Wine*: ablations

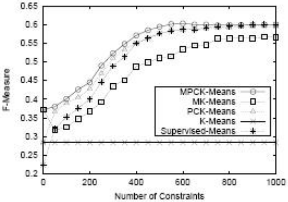


Figure 4. *Protein*: ablations

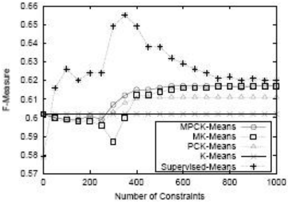


Figure 5. *Ionosphere*: ablations

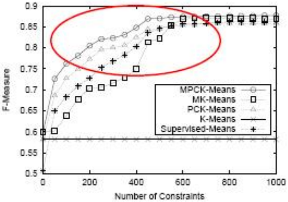


Figure 6. *Digits-389*: ablations

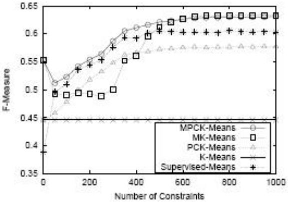


Figure 7. *Letters-IJL*: ablations

Experiment Results(2)

A single diagonal matrix compared with multiple full matrix.

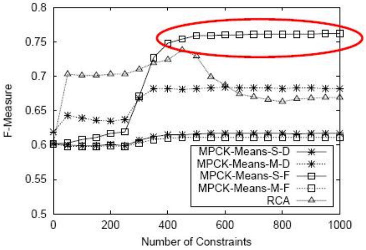


Figure 11. *Ionosphere*: metric learning

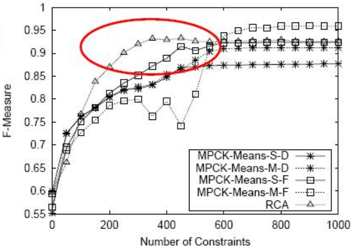


Figure 12. *Digits-389*: metric learning

Some phenomenons

- Use different matrix and cluster and use full matrix definitely increase the performance.
- When the constraints are few, RCA seems working better than MPCK-means. Why?

Experiment Results(2)

A single diagonal matrix compared with multiple full matrix.

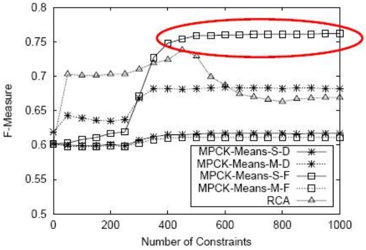


Figure 11. *Ionosphere*: metric learning

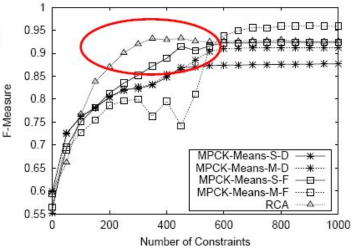


Figure 12. *Digits-389*: metric learning

Some phenomenons

- Use different matrix and cluster and use full matrix definitely increase the performance.
- When the constraints are few, RCA seems working better than MPCK-means. Why?

Conclusions

By integrating metric learning and constraints during clustering, it outperforms each single approach.

Questions?

Thank you!!

Conclusions

By integrating metric learning and constraints during clustering, it outperforms each single approach.

Questions?

Thank you!!