Clustering will not be satisfactory if:
-- in the input space the clusters are not linearly separable;
-- the distance measure is not adequate;
-- the assumptions limit the shape or the number of the clusters.

Our solution:
1. map to a high-dimensional space;
2. minimize a convex objective function;
3. do mean shift clustering in the learned space.
Semi-Supervised Kernel Mean Shift Clustering

- **Input**
  - unlabeled data
  - pairwise constraints

- **Parameters**
  - automatically determined using pairwise constraints

- **Output**
  - clusters and labels
a few must-link and cannot-link pairs cluster label not specified

nonlinear input

linearly separable kernel space

Bregman logdet divergence

kernel mean shift
Motivation: High Dimensional Spaces

• Input space
  – nonlinearly separable data
  – clusters not consistent with class labels

• Mapping to a high-dimensional space
  – data becomes linearly separable
  – meaningful clusters... maybe not

• Explicit mapping is not always possible
  – solution: use kernel methods

\[ \phi(x) = [x \ x^2] \]
Kernel Space Representation

- Inner product representation using a Gram Matrix $K$
- Feature space representation
  - nonlinear mapping
- Distance computation in the feature space
  - sample points
    $$d_H^2(x_i, x_j) = K_{ii} + K_{jj} - 2K_{ij}$$
  - linear combinations of sample points
    $$\|y - y'\|_H^2 = \alpha_y^T K \alpha_y + \alpha_{y'}^T K \alpha_{y'} - 2\alpha_y^T K \alpha_{y'}$$
- Mapping function $\phi(\cdot)$ need not be known

\[\text{Input Space}\]
\[x_i \in \mathcal{X}, i = 1, \ldots, n\]
\[X = [x_1, x_2, \ldots, x_n]\]

\[\text{Feature Space}\]
\[\phi : \mathcal{X} \mapsto \mathcal{H}\]
\[\Phi = [\phi(x_1), \ldots, \phi(x_n)]\]
\[K = \Phi^T \Phi\]

Linear combination of samples in feature space
\[y' = \Phi \alpha_{y'}\]
\[y = \Phi \alpha_y\]
Kernel Parameter Selection $\sigma$.

Gaussian kernel function for the initial kernel matrix.

$$K_{ij}(\sigma) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \in [0, 1]$$

distance of any two points

$$\|\Phi \alpha_{x_i} - \Phi \alpha_{x_j}\|^2 = \alpha_{x_i}^T K \alpha_{x_i} + \alpha_{x_j}^T K \alpha_{x_j} - 2\alpha_{x_i}^T K \alpha_{x_j}$$

is between 0 and 2

Kernel parameter estimation is already using Bergman log det divergence.

$$\hat{\sigma} = \arg \min_{\sigma \in S} D_{ld}(\text{diag}(\xi_\sigma), \text{diag}(\xi))$$

$$\xi_{(ij)} = \begin{cases} 
  d_m & \forall (i, j) \in M \\
  d_c & \forall (i, j) \in C. 
\end{cases}$$

$m + c = n_c$ constraints

$$d_m = \min(d_5, 0.05)$$

$$d_c = \max(d_{95}, 1.95)$$

The kernel matrix $K_{\hat{\sigma}}$ has rank $n$, the number of points.

The reduced rank $r < n$ kernel $K$ has only 0.99 percentage of the original kernels energy in the rest of computation.
All databases have a miminum in sigma. Two examples:

**Olympic circles.** 5 classes.  
\( n = 1500, \ r = 58. \)  x300 faster (low_rank/full_rank)

**USPS digits.** 10 classes.  
(here) \( n = 1000, \ r = 499. \)  x2 faster (low_rank/full_rank)
Motivation: Need for Supervision

- Kernel space
  - mapping to kernel space may make data linearly separable
  - data may still not form ‘desired’ clusters

- Use supervision information
  - pairwise constraints to guide clustering
  - similarity constraint pair \((j_1, j_2)\)
  \[
  a_j = \phi(x_{j_1}) - \phi(x_{j_2}) = \Phi(e_{j_1} - e_{j_2})
  \]

- Projected kernel space
  - project all data to null space of \(a_j\)
  - similar points coincide

Projecting in to the null space.

Data in kernel space may still not form desired linear clusters.

Need a few pairwise constraints of the data to guide clustering.

Each projection reduces with one the dimensionality of new kernel space.

Null space projection does not have cannot-link constraints.

Sensitive to labeling errors.
An example of null space projection. 5x150 points.

6 x 5 = 30 must-link constraints
pairwise 4 points per class

Example – Five concentric circles

Initial data with constraint points
Clustering using clean training data

Clustering after adding one mislabeled constraint (black link)

Plus one mislabeled constraint.
Linear Transformations in Kernel Spaces

- Let $\mathcal{M}$ be the set of similarity constraint pairs

$$a_j = \phi(x_{j1}) - \phi(x_{j2}) = \Phi(e_{j1} - e_{j2})$$

$$p = \|a_j\|^2 = (e_{j1} - e_{j2})^\top K (e_{j1} - e_{j2})$$

- A mapping $\hat{\phi}(x) = P\phi(x)$, defined using the symmetric matrix

$$P = I - \frac{s}{p} (a_j a_j^\top)$$

- $P$ is a projection matrix for $s = 1$

$$P\Phi e_{j1} - P\Phi e_{j2} = 0$$

$\Phi = [\phi(x_1), \ldots, \phi(x_n)]$

$I_{n \times n} = [e_1, e_2, \ldots, e_n]$}

$$K = \Phi^\top \Phi$$

Feature Space
Kernel Learning using Linear Transformations

• Given a target distance $d_m$ between a constraint pair $(j_1, j_2) \in \mathcal{M}$

• The learned kernel function is
  \[
  \hat{K}(x, x') = \hat{\phi}(x)^\top \hat{\phi}(x') = \phi(x)^\top P^\top P \phi(x')
  \]

• The learned kernel matrix can be expressed directly in terms of $K$
  \[
  \hat{K} = K + \beta K (e_{j_1} - e_{j_2}) (e_{j_1} - e_{j_2})^\top K
  \]
  \[
  \beta = \left( \frac{d_m}{p^2} - \frac{1}{p} \right)
  \]
  \[
  p = (e_{j_1} - e_{j_2})^\top K (e_{j_1} - e_{j_2})
  \]
  \[
  \text{Crucial result: Minimizes the Log Det Bregman divergence between } K \text{ and } \hat{K}
  \]

\[
\begin{align*}
P &= I - \frac{s}{p} (a_j a_j^\top) \\
\end{align*}
\]
Bregman Divergences

For real, symmetric \( n \times n \) matrices \( \mathbf{X} \) and \( \mathbf{Y} \) the log det divergence is a Bregman divergence for the convex function \( \varphi(\mathbf{X}) = -\log \det \mathbf{X} \)

\[
D_{\varphi}(\mathbf{X}, \mathbf{Y}) = \varphi(\mathbf{X}) - \varphi(\mathbf{Y}) - \text{tr} \left( (\nabla \varphi(\mathbf{Y}))^\top (\mathbf{X} - \mathbf{Y}) \right)
\]

\[
D_{ld}(\mathbf{X}, \mathbf{Y}) = \text{tr} (\mathbf{X} \mathbf{Y}^{-1}) - \log \det (\mathbf{X} \mathbf{Y}^{-1}) - n
\]

Nonnegative scalar function \( D_{ld}(\mathbf{X}, \mathbf{Y}) = 0 \) iff \( \mathbf{X} = \mathbf{Y} \)

- convex in the first argument
- not a metric, does not obey triangle inequality

Defined only for positive semidefinite matrices like the kernel mapping functions
Kernel Learning by Minimizing the Log Det Divergence

A convex kernel learning problem formulated having both the similarity ($\mathcal{M}$) and dissimilarity ($\mathcal{C}$) with soft constraints

$$\min_{\hat{K}, \xi} \quad D_{ld}(\hat{K}, K) + \gamma D_{ld}(\text{diag} (\hat{\xi}), \text{diag} (\xi))$$

s.t.  

$$(e_{j_1} - e_{j_2})^\top \hat{K} (e_{j_1} - e_{j_2}) \leq \hat{\xi}_j \quad \forall (j_1, j_2) \in \mathcal{M}$$

$$(e_{j_1} - e_{j_2})^\top \hat{K} (e_{j_1} - e_{j_2}) \geq \hat{\xi}_j \quad \forall (j_1, j_2) \in \mathcal{C}.$$  

At each iteration the slack variable is updated.

- For each constraint, the optimization is solved by Bregman projection based updates

$$\hat{K}_{t+1} = \hat{K}_t + \beta_t \hat{K}_t (e_{j_1} - e_{j_2}) (e_{j_1} - e_{j_2})^\top \hat{K}_t$$

- Updates are repeated until convergence to global minimum


Kernel Learning by Minimizing the Log Det Divergence

• For each constraint, the optimization is solved by Bregman projection based updates.

\[ \hat{K}_{t+1} = \hat{K}_t + \beta_t \hat{K}_t (e_i - e_j) (e_i - e_j)^\top \hat{K}_t \]

• Updates are repeated until convergence.

• The initial kernel matrix \( K \) has rank \( r \leq n \), then, and the update can be rewritten as

\[ K = GG^\top \]

\[ \hat{K}_{t+1} = \hat{G}_t \left( I_r + \beta_t \hat{G}_t^\top (e_i - e_j)(e_i - e_j)^\top \hat{G}_t \right) \hat{G}_t^\top \]

• The scalar variable \( \beta_t \) is computed in each iteration using the kernel matrix and the constraint pair.

• The \( n \times r \) matrix \( \hat{G}_t \) is updated using the Cholesky decomposition.

Initialize $t = 0$, $B_0 = I_{r \times r}$, $\lambda_j = 0$

Initialize slack variables
$\hat{\xi}_j = \xi_j = d_m$ for $(j_1, j_2) \in \mathcal{M}$
$\hat{\xi}_j = \xi_j = d_c$ for $(j_1, j_2) \in \mathcal{C}$

Repeat until convergence
- Pick a constraint $(j_1, j_2) \in \mathcal{M}$ or $\mathcal{C}$.
- $\delta = 1$ if $(j_1, j_2) \in \mathcal{M}$ or $-1$ if $(j_1, j_2) \in \mathcal{C}$.
- $w = B_t \left[ G(j_1, \cdot) - G(j_2, \cdot) \right]^T$
- $\alpha = \min \left( \lambda_j, \frac{\delta \gamma}{\gamma + 1} \left( \frac{1}{\|w\|_2^2} - \frac{1}{\hat{\xi}_j} \right) \right)$.
- $\hat{\xi}_j \leftarrow \gamma \hat{\xi}_j / \left( \gamma + \delta \alpha \hat{\xi}_j \right)$.
- $\lambda_j \leftarrow \lambda_j - \alpha$.
- $\beta = \delta \alpha / (1 - \delta \alpha \|w\|_2^2)$.
- Factorize $(I + \beta ww^T)$ into $L_t L_t^T$ using Cholesky decomposition. Update $B_{t+1} = B_t L_t$.
- $t \leftarrow t + 1$.

return $\hat{G} = GB_{t_{\text{conv}}}$

Low rank kernel learning algorithm

Input:
$G$ - $n \times r$ initial square root matrix $K = GG^T$
$\mathcal{M}, \mathcal{C}$ - similarity and dissimilarity constraint sets $d_m, d_c$ - the distance thresholds.
$\gamma$ - the trade-off parameter
Output:
$\hat{G}$ - the $n \times r$ learned square root matrix.
Kernel Learning Using the Log Det Divergence

• For very large datasets, it is infeasible
  – to learn the entire kernel matrix, and
  – to store it in the memory.

• Generalization to *out of sample* points

\[
\hat{K}(x, y) = K(x, y) + k_x^\top \left(K^{-1} \left(\hat{K} - K\right) K^{-1}\right) k_y
\]
\[
k_x = [K(x, x_1), \ldots, K(x, x_n)]^\top \quad k_y = [K(y, x_1), \ldots, K(y, x_n)]^\top
\]

where \(x\) or \(y\) or both are out of sample.

• Distances can be computed using the learned kernel function \(\hat{K}(x, y)\)
Kernel Mean Shift Clustering

A nonparametric kernel mean shift (here the $x$ is generic !) finds the closest mode (maximum) of the probability estimate.

$$k(x) \text{ the kernel profile}$$

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} k \left( \left\| \frac{x-x_i}{h_i} \right\|^2 \right)$$

$$\bar{x} = \frac{\sum_{i=1}^{n} \frac{x_i}{h_i^{d+2}} g \left( \left\| \frac{x-x_i}{h_i} \right\|^2 \right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g \left( \left\| \frac{x-x_i}{h_i} \right\|^2 \right)}$$

From the obtained optimal kernel matrix $\hat{K}$ only at most the 25 largest singular values give the dimension of the mean shift, for all the databases.

The bandwidth parameter $h_i$ for each constraint pair of must-link points is the median of the k-th nearest neighbor.
Bandwidth Parameter Selection

- For a constraint pair $(j_1, j_2) \in \mathcal{M}$,
  \[ d_i = (e_{j_1} - e_i)^\top \hat{K} (e_{j_1} - e_i) \]
  \(i = 1, \ldots, n, \ i \neq j_1\)

- Sort in increasing order of $d_i$.

- Find $k$ as the index of $j_2$ in the sorted list.

- Median of $k$ over all must-link constraints is used as bandwidth parameter in mean shift for all $n$ points.

Equal number of cannot-link also used for the doing the entire algorithm.

Olympic Circles 5x10 must-l.

USPS Digits 10x10 must-l.
CONTINGENCY TABLE

<table>
<thead>
<tr>
<th></th>
<th>Ground-truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>True T</td>
<td>Not true</td>
</tr>
<tr>
<td>False F</td>
<td>True</td>
</tr>
<tr>
<td>Negative N</td>
<td></td>
</tr>
<tr>
<td>Positive P</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result</th>
<th>Not true</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not true</td>
<td>TN</td>
<td>FN</td>
</tr>
<tr>
<td>True</td>
<td>FP</td>
<td>TP</td>
</tr>
</tbody>
</table>

DEFINITIONS

\[
\text{quality} = \frac{TP}{TP + FP + FN}
\]

\[
\text{precision} = \frac{TP}{TP + FP} \quad [\text{are correct}]
\]

\[
\text{recall} = \frac{TP}{TP + FN} \quad [\text{true return}]
\]

\[
F_{\text{measure}} = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

ROC curve

\[
\text{ordinate: true positive rate (recall)} \quad \frac{TP}{TP+FN}
\]

\[
\text{abscissa: false positive alarm rate} \quad \frac{FP}{FP+TN}
\]

A threshold measures if the results are false or true. Changing the threshold rearranges all four true/not_true boxes function of the ground truth. The thresholding start at one extreme, just one sample is true, \(TP = 1\). Precision is one, recall close the zero. The other extreme is when the threshold has all the returned samples true. The \(TP\) is now the number of true samples. The precision is close to zero, the recall is one.
A simple example:
The ground-truth have 15 items of A, associated as not_true, and 5 items of B, associated as true.

Threshold returning a single true value gives the contingency table

<table>
<thead>
<tr>
<th></th>
<th>Ground-truth</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not true</td>
<td>True</td>
</tr>
<tr>
<td>Result</td>
<td>Not true</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>0</td>
</tr>
</tbody>
</table>

and \( \text{precision} = 1 \) while \( \text{recall} = 0.2 \).

Threshold returning an immediate result gives the contingency table

<table>
<thead>
<tr>
<th></th>
<th>Ground-truth</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not true</td>
<td>True</td>
</tr>
<tr>
<td>Result</td>
<td>Not true</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>6</td>
</tr>
</tbody>
</table>

and \( \text{precision} = 0.33 \) while \( \text{recall} = 0.6 \).

Threshold returning all the results as true gives the contingency table

<table>
<thead>
<tr>
<th></th>
<th>Ground-truth</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not true</td>
<td>True</td>
</tr>
<tr>
<td>Result</td>
<td>Not true</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td>15</td>
</tr>
</tbody>
</table>

and \( \text{precision} = 0.25 \) while \( \text{recall} = 1 \).
Adjusted Rand Index

• *Scalar* measure to evaluate clustering performance from the clustering output.

\[
AR = \frac{RI - \text{expected}_RI}{\max_{RI} - \text{expected}_RI}
\]

\[
RI = \frac{TP + TN}{TP + FP + FN + TN}
\]

negative if expected larger than obtained

TP – true positive; TN – true negative;
FP – false positive; FN – false negative.

• Compensates for chance, randomly assigned cluster labels get a low score.
Comparison metric. The clustering performance of different algorithms is compared using the Adjusted Rand (AR) index [22]. It is an adaptation of the rand index that penalizes random cluster assignments, while measuring the agreement between the clustering output and the ground truth labels. Using the contingency table $t$, the AR index can be computed as [10]

$$AR = \frac{RI - \mathbb{E}(RI)}{\text{max}_RI - \mathbb{E}(RI)}$$

$$= \frac{N (a + d) - [(a + b)(a + c) + (d + b)(d + c)]}{N^2 - [(a + b)(a + c) + (d + b)(d + c)]}$$

$$a = \sum_{r=1}^{R} \sum_{c=1}^{C} \binom{t_{rc}}{2}, \quad b = \sum_{r=1}^{R} \binom{t_r.}{2} - a$$

$$c = \sum_{c=1}^{C} \binom{t_{.c}}{2} - a, \quad d = \binom{n}{2} - a - b - c$$

where $t_{rc} = t(r, c)$, $t_r. = \sum_c t(r, c)$ and $t_{.c} = \sum_r t(r, c)$ and $N$ is the sum of all entries in the contingency table, and $\mathbb{E}(\cdot)$ is the expectation operator. The AR index is a scalar and takes values between zero and one, with perfect clustering yielding a value of one.
Trade-off Parameter Gamma

Measure with the scalar Adjusted Rand (AR) with half data points for learning and half for testing. (two-fold cross-validation)

Pairwise constraints: 15 points per class. Number of must-link equal number of cannot-link. *The entire algorithm is evaluated.*

Must-link constraints.

*Olympic circles:* $5 \times 10^5 = 525$

*USPS digits:* $10 \times 10^5 = 1050$

Selecting the trade-off parameter $\gamma$. The AR index vs $\log \gamma$ for 50 cross-validation runs. The asterisks mark the selected value of $\gamma = 100$. 
Examples.

The algorithm generalizes to *out of sample points* not taken into consideration in the kernel matrix optimization.
Pairwise Constraint Generation

• Assuming $b$ labeled points are selected at random from each class.
• $\binom{b}{2}$ similarity pairs are generated from each class.
• An equal number of dissimilarity pairs are also generated.
• The value of $b$ is varied.
Comparison with our SKMS Clustering

1. Efficient and exhaustive constraint propagation for spectral clustering (E2CP)
2. Semi-supervised kernel k-means (SSKK)
3. Kernel k-means (Kkm) using LogDet divergence
4. Kernel Spectral Clustering (KSC) using LogDet divergence

All these methods have to be given the number of clusters as input.

1. Zhiwu Lu and Horace H.S. Ip, Constrained Spectral Clustering via Exhaustive and Efficient Constraint Propagation, ECCV, 1—14, 2010
Experimental Evaluation

• Two synthetic data sets
  – Olympic circles (5 classes)
  – Concentric circles (10 classes)
    Nonlinearly separable
    Can have intersecting boundaries

• Four real data sets
  – Small number of classes
    • USPS (10 Classes)
    • MIT Scene (8 Classes)
  – Large number of classes
    • PIE faces (68 Classes)
    • Caltech Objects (50 Classes)

50 independent runs. Average parameter values!
(\(\sigma = 0.75\), \(k = 25\), \(r_{ms} = 10\))

Synthetic Example 1: Olympic Circles

- 300 points along each of the five circles.

- 25 points per class selected at random.

- **Experiment 1:**
  - Varied number of labeled points \([5, 7, 10, 12, 15, 17, 20, 25]\) from each class to generate pairwise constraints.
Synthetic Example 1: Olympic Circles

- **Experiment 2**
  - 20 labeled points per class.
  - Introduce labeling errors by swapping similarity pairs with dissimilarity pairs.
  - Varied fraction of mislabeled constraints.

\[(\sigma = 0.75, \ k = 25, \ r_{ms} = 10)\]
Synthetic Example 2: Concentric Circles

- 100 points along each of the ten concentric circles

- Experiment 1
  - Varied number of labeled points [5, 7, 10, 12, 15, 17, 20, 25] from each class to generate pairwise constraints

- Experiment 2
  - 25 labeled points per class
  - Introduce labeling errors by swapping similarity pairs with dissimilarity pairs
Real Example 1: USPS Digits

- Ten classes with 1100 points per class. A total of 11000 points
- 100 points per class → \( K \) 1000x1000 initial kernel matrix
- Varied number of labeled points [5, 7, 10, 12, 15, 17, 20, 25] from each class to generate pairwise constraints
- Cluster all 11000 data points by generalizing to the remaining 10000 points. ARI = 0.7529 ± 0.051

\[ (\sigma = 7, \ k = 10, \ r_{ms} = 25) \]
Real Example 2: MIT Scene

- Eight classes with 2688 points. The number of samples range between 260 and 410.
- 100 points per class → $K$ 800x800 initial kernel matrix.
- Varied number of labeled points [5, 7, 10, 12, 15, 17, 20] from each class to generate pairwise constraints.
- Cluster all 2688 data points by generalizing to the remaining 1888 points.
Is it really worth to do it *per image*?

Sample images from each of the eight categories of the MIT scene data set.

See three images misclassified from a class.
Real Example 3: PIE Faces

• 68 subjects with 21 samples per subject

• $K \rightarrow 1428 \times 1428$ full initial kernel matrix

• Varied number of labeled points [3, 4, 5, 6, 7] from each class to generate pairwise constraints

• Obtained perfect clustering for more than 5 labeled points per class
Real Example 4: Caltech-101 (subset)

- 50 categories with number of samples ranging between 31 and 40 points per class

- $K \rightarrow 1959 \times 1959$ full initial kernel matrix

- Varied number of labeled points $[5, 7, 10, 12, 15]$ from each class to generate pairwise constraints

$(\sigma = 0.5, \ k = 5, \ r_{ms} = 20)$
Conclusion

– recovers arbitrarily shaped nonparametric clusters

– performs well with databases having large number of clusters... within the convention of clustering

-- generalizes to out of sample points