Adaptive Distances on Sets of Vectors

Adam Woźnica and Alexandros Kalousis

Department of Computer Science
University of Geneva
Switzerland

December 17, 2010

Motivation

Motivation

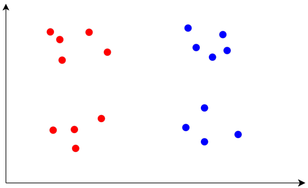
Distance learning for kNN:

- Learn distance to better separates classes
- Works well in practice
- Efficient implementations
- Mostly for vectorial data
  - Focus on Mahalanobis distance (=linear transformation)

Motivation

Distance learning for kNN:

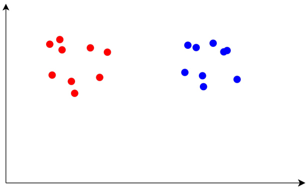
- Learn distance to better separates classes
- Works well in practice
- Efficient implementations
- Mostly for vectorial data
  - Focus on Mahalanobis distance (=linear transformation)



Motivation

Distance learning for kNN:

- Learn distance to better separates classes
- Works well in practice
- Efficient implementations
- Mostly for vectorial data
  - Focus on Mahalanobis distance (=linear transformation)



Motivation (cont.)

Motivation (cont.)

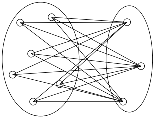
Other structures (sets, graphs)?

Motivation (cont.)

Other structures (sets, graphs)?

- Possible to kernelize existing methods

$$K(X_i, X_j) = \sum_{(x_i, x_j) \in X_i \times X_j} k(x_i, x_j)$$

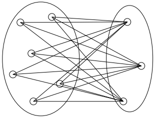


Motivation (cont.)

Other structures (sets, graphs)?

- Possible to kernelize existing methods

$$K(X_i, X_j) = \sum_{(x_i,x_j)\in X_i \times X_j} k(x_i, x_j)$$



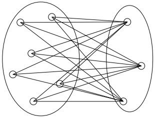
- Might not be appropriate in some cases

Motivation (cont.)

Other structures (sets, graphs)?

- Possible to kernelize existing methods

$$K(X_i, X_j) = \sum_{(x_i,x_j)\in X_i \times X_j} k(x_i, x_j)$$



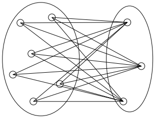
- Might not be appropriate in some cases
- Multiple-instance problems

Motivation (cont.)

Other structures (sets, graphs)?

- Possible to kernelize existing methods

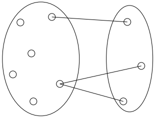
$$K(X_i, X_j) = \sum_{(x_i,x_j)\in X_i \times X_j} k(x_i, x_j)$$



- Might not be appropriate in some cases
- Multiple-instance problems

Our solution:

Learn matching based set distances



Outline

Set Distances

Learning Set Distances

Experiments
- Artificial Data
- Real world Data

Conclusions and Future Research

Matching Based Set Distances

Matching Based Set Distances

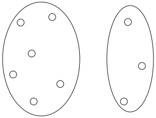
$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

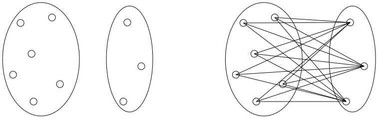
- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)



Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

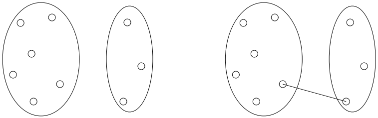


Average Linkage

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

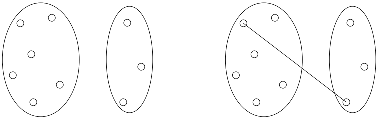


Single Linkage

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

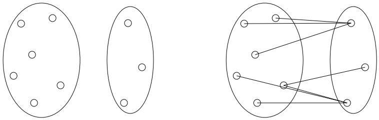


Complete Linkage

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

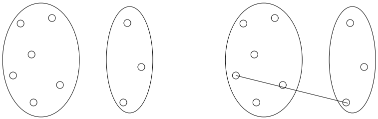


Sum of Minimum Distances

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

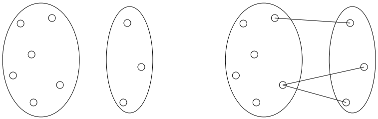


Hausdorff

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)

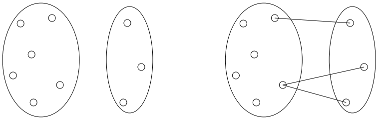


RIBL

Matching Based Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- $\mathcal{F}$: mapping family
- $F \subseteq X_i \times X_j$
- $d(F)$: set of pairwise distances d
- $\mathcal{A}$: aggregation function (usually max, min, average)



RIBL

- Other options possible

Smooth Set Distances

Smooth Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

Smooth Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

Distances not smooth, if $\mathcal{A}$ involves min or max

- More difficult to optimize

Smooth Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

Distances not smooth, if $\mathcal{A}$ involves min or max

- More difficult to optimize

Possible to replace

- min with softmin
- max with softmax

Smooth Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

Distances not smooth, if $\mathcal{A}$ involves min or max

- More difficult to optimize

Possible to replace

- min with softmin
- max with softmax

Details and empirical results in the paper

Outline

How to Learn Set Distances

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- Difficult to learn $\mathcal{F}$ and $\mathcal{A}$
  - Not clear how to represent them in a structured manner

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- Difficult to learn $\mathcal{F}$ and $\mathcal{A}$
  - Not clear how to represent them in a structured manner
- For fixed $\mathcal{F}$ and $\mathcal{A}$, distance determined by d

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- Difficult to learn $\mathcal{F}$ and $\mathcal{A}$
  - Not clear how to represent them in a structured manner
- For fixed $\mathcal{F}$ and $\mathcal{A}$, distance determined by d

$$D(X_i, X_j) = \mathcal{A}\{d_{\boldsymbol{A}}(F) \mid F \in \mathcal{F}\}$$

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- Difficult to learn $\mathcal{F}$ and $\mathcal{A}$
  - Not clear how to represent them in a structured manner
- For fixed $\mathcal{F}$ and $\mathcal{A}$, distance determined by d

$$D(X_i, X_j) = \mathcal{A}\{d_{\boldsymbol{A}}(F) \mid F \in \mathcal{F}\}$$

- $d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^T \boldsymbol{A}(\boldsymbol{x}_i - \boldsymbol{x}_j)$

How to Learn Set Distances

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

- Difficult to learn $\mathcal{F}$ and $\mathcal{A}$
  - Not clear how to represent them in a structured manner
- For fixed $\mathcal{F}$ and $\mathcal{A}$, distance determined by d

$$D(X_i, X_j) = \mathcal{A}\{d_{\boldsymbol{A}}(F) \mid F \in \mathcal{F}\}$$

- $d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^T \boldsymbol{A} (\boldsymbol{x}_i - \boldsymbol{x}_j)$
- We focus on learning $\boldsymbol{A}$
  - Equivalent to learning representation of sets' elements

”Optimal” set distance

"Optimal" set distance

$$\min_{\boldsymbol{A}} \quad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

"Optimal" set distance

$$\min_{\boldsymbol{A}} \quad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

$\mathcal{F}_{\boldsymbol{A}}$: Continuous version of LOO

"Optimal" set distance

$$\min_{\boldsymbol{A}} \qquad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

$\mathcal{F}_{\boldsymbol{A}}$: Continuous version of LOO

- $\mathcal{F}_{\boldsymbol{A}} = \frac{1}{n}\sum_i \sum_{\{j \,:\, y_i = y_j\}} \frac{\exp - D_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{k \neq i} \exp - D_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k)}$

"Optimal" set distance

$$\min_{\boldsymbol{A}} \qquad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

$\mathcal{F}_{\boldsymbol{A}}$: Continuous version of LOO

- $\mathcal{F}_{\boldsymbol{A}} = \frac{1}{n}\sum_i \sum_{\{j\ :\ y_i=y_j\}} \frac{\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j)}{\sum_{k\neq i}\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k)}$
- Keep equally labeled instances close together and far apart from other instances

"Optimal" set distance

$$\min_{\boldsymbol{A}} \quad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

$\mathcal{F}_{\boldsymbol{A}}$: Continuous version of LOO

- $\mathcal{F}_{\boldsymbol{A}} = \frac{1}{n}\sum_i \sum_{\{j\,:\,y_i=y_j\}} \frac{\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j)}{\sum_{k\neq i}\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k)}$
- Keep equally labeled instances close together and far apart from other instances
- $O(n^2m^2p^2)$

"Optimal" set distance

$$\min_{\boldsymbol{A}} \qquad \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) + \lambda\Omega(\boldsymbol{A})$$

- $\mathcal{F}_{\boldsymbol{A}}$: cost function
- $\Omega(\boldsymbol{A})$: regularization factor
- λ: regularization parameter

$\mathcal{F}_{\boldsymbol{A}}$: Continuous version of LOO

- $\mathcal{F}_{\boldsymbol{A}} = \frac{1}{n}\sum_i \sum_{\{j\ :\ y_i=y_j\}} \frac{\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j)}{\sum_{k\neq i}\exp -D_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k)}$
- Keep equally labeled instances close together and far apart from other instances
- $O(n^2 m^2 p^2)$
- NCA : **N**eighborhood **C**omponent **A**nalysis

Outline

Outline

Experimental Setup

Experimental Setup

Datasets (x10):

- 100 sets
- 1 to 10 vectors per set; in $\mathbb{R}^{100}$
- Binary labels
- *l* dimensions generated from a Gaussian distr., with mean
  - μ for positive sets
  - $-\mu$ for negative sets

Experimental Setup

Datasets (x10):

- 100 sets
- 1 to 10 vectors per set; in $\mathbb{R}^{100}$
- Binary labels
- *l* dimensions generated from a Gaussian distr., with mean
  - μ for positive sets
  - $-\mu$ for negative sets

Intuition:

- The larger $|\mu|$: the more discriminatory is an attribute
- The larger *l*: the easier is the classification problem

Experimental Setup

Datasets (x10):

- 100 sets
- 1 to 10 vectors per set; in $\mathbb{R}^{100}$
- Binary labels
- *l* dimensions generated from a Gaussian distr., with mean
  - μ for positive sets
  - $-\mu$ for negative sets

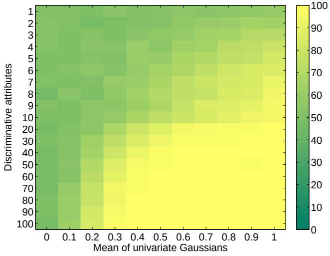
Intuition:

- The larger $|\mu|$: the more discriminatory is an attribute
- The larger *l*: the easier is the classification problem

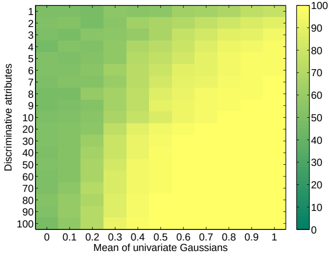
Algorithm:

- kNN

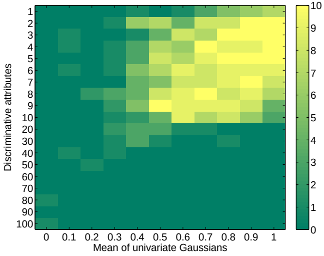
Results



(a) standard set distances



(b) adaptive set distances



(c) significance tests

Outline

Experimental Setup

Experimental Setup

Goals:

- Adaptive distances vs. standard distances
- Non-convexity of NCA

Experimental Setup

Goals:

- Adaptive distances vs. standard distances
- Non-convexity of NCA

Algorithm:

- kNN
- Full and diagonal $\boldsymbol{A}$

Experimental Setup

Goals:

- Adaptive distances vs. standard distances
- Non-convexity of NCA

Algorithm:

- kNN
- Full and diagonal $\boldsymbol{A}$

Datasets:

Datasets	n	m	p	c
musk1	92	5.17	167	2
musk2	102	64.69	167	2
muta	188	27.89	101	2
fm	349	25.62	51	2
mm	336	25.40	51	2
tiger	200	6.1	230	2
fox	200	6.95	230	2
elephant	200	6.6	230	2

Adaptive distances vs. standard distances

		musk1	musk2	muta	fm	mm	tiger	fox	eleph.
D^{AL}	D	76.4	71.0	57.6	58.6	63.0	72.0	52.0	76.0
	$D_{\mathbf{A}}^{f}$	75.9	71.5	58.2	50.1-	53.5-	74.0	56.5	79.5
	$D_{\mathbf{A}}^{d}$	78.0	76.5	**68.4+**	40.4-	48.1-	**78.5+**	**59.5+**	77.0
D^{SL}	D	81.1	77.2	42.2	41.6	39.3	76.0	56.5	71.5
	$D_{\mathbf{A}}^{f}$	78.4	69.5	41.6	41.6	39.3	73.5	57.5	72.5
	$D_{\mathbf{A}}^{d}$	82.4	68.5-	42.8	41.6	39.3	79.0	61.5+	72.5
D^{CL}	D	67.5	71.5	39.4	43.2	40.9	67.5	54.5	64.5
	$D_{\mathbf{A}}^{f}$	71.7	65.5	56.3+	47.3	**46.6+**	68.5	47.0	55.5
	$D_{\mathbf{A}}^{d}$	**77.3+**	**79.5+**	**58.2+**	**52.1+**	**58.1+**	72.5	51.5	74.0+
D^{H}	D	81.3	78.0	75.5	46.8	48.3	74.0	51.5	74.5
	$D_{\mathbf{A}}^{f}$	83.5	73.3	77.5	**57.5+**	**57.2+**	74.0	55.5	65.5
	$D_{\mathbf{A}}^{d}$	78.8	77.2	**80.6+**	**57.2+**	**53.8+**	76.5	**65.5+**	75.5
D^{SMD}	D	83.3	75.2	79.4	56.6	58.2	77.5	60.0	79.5
	$D_{\mathbf{A}}^{f}$	77.1	73.3	**81.4+**	58.5	58.3	76.0	59.0	77.5
	$D_{\mathbf{A}}^{d}$	81.3	73.3	80.5	56.6	58.7	**85.0+**	62.0	**85.5+**

Adaptive distances vs. standard distances

		musk1	musk2	muta	fm	mm	tiger	fox	eleph.
D^{AL}	D	76.4	71.0	57.6	58.6	63.0	72.0	52.0	76.0
	D^f_A	75.9	71.5	58.2	50.1-	53.5-	74.0	56.5	79.5
	D^d_A	78.0	76.5	**68.4+**	40.4-	48.1-	**78.5+**	**59.5+**	77.0
D^{SL}	D	81.1	77.2	42.2	41.6	39.3	76.0	56.5	71.5
	D^f_A	78.4	69.5	41.6	41.6	39.3	73.5	57.5	72.5
	D^d_A	82.4	68.5-	42.8	41.6	39.3	79.0	61.5+	72.5
D^{CL}	D	67.5	71.5	39.4	43.2	40.9	67.5	54.5	64.5
	D^f_A	71.7	65.5	56.3+	47.3	**46.6+**	68.5	47.0	55.5
	D^d_A	**77.3+**	**79.5+**	**58.2+**	**52.1+**	**58.1+**	72.5	51.5	74.0+
D^{H}	D	81.3	78.0	75.5	46.8	48.3	74.0	51.5	74.5
	D^f_A	83.5	73.3	77.5	**57.5+**	**57.2+**	74.0	55.5	65.5
	D^d_A	78.8	77.2	**80.6+**	**57.2+**	**53.8+**	76.5	**65.5+**	75.5
D^{SMD}	D	83.3	75.2	79.4	56.6	58.2	77.5	60.0	79.5
	D^f_A	77.1	73.3	**81.4+**	58.5	58.3	76.0	59.0	77.5
	D^d_A	81.3	73.3	80.5	56.6	58.7	**85.0+**	62.0	**85.5+**

- Usually there exists a mapping better than cross product

Adaptive distances vs. standard distances

		musk1	musk2	muta	fm	mm	tiger	fox	eleph.
D^{AL}	D	76.4	71.0	57.6	58.6	63.0	72.0	52.0	76.0
	$D^f_{\boldsymbol{A}}$	75.9	71.5	58.2	50.1-	53.5-	74.0	56.5	79.5
	$D^d_{\boldsymbol{A}}$	78.0	76.5	**68.4+**	40.4-	48.1-	**78.5+**	**59.5+**	77.0
D^{SL}	D	81.1	77.2	42.2	41.6	39.3	76.0	56.5	71.5
	$D^f_{\boldsymbol{A}}$	78.4	69.5	41.6	41.6	39.3	73.5	57.5	72.5
	$D^d_{\boldsymbol{A}}$	82.4	68.5-	42.8	41.6	39.3	79.0	61.5+	72.5
D^{CL}	D	67.5	71.5	39.4	43.2	40.9	67.5	54.5	64.5
	$D^f_{\boldsymbol{A}}$	71.7	65.5	56.3+	47.3	**46.6+**	68.5	47.0	55.5
	$D^d_{\boldsymbol{A}}$	**77.3+**	**79.5+**	**58.2+**	**52.1+**	**58.1+**	72.5	51.5	74.0+
D^{H}	D	81.3	78.0	75.5	46.8	48.3	74.0	51.5	74.5
	$D^f_{\boldsymbol{A}}$	83.5	73.3	77.5	**57.5+**	**57.2+**	74.0	55.5	65.5
	$D^d_{\boldsymbol{A}}$	78.8	77.2	**80.6+**	**57.2+**	**53.8+**	76.5	**65.5+**	75.5
D^{SMD}	D	83.3	75.2	79.4	56.6	58.2	77.5	60.0	79.5
	$D^f_{\boldsymbol{A}}$	77.1	73.3	**81.4+**	58.5	58.3	76.0	59.0	77.5
	$D^d_{\boldsymbol{A}}$	81.3	73.3	80.5	56.6	58.7	**85.0+**	62.0	**85.5+**

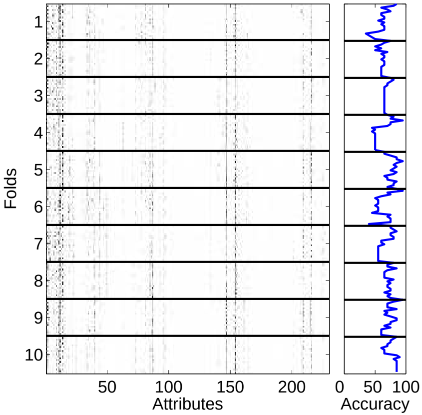
- Usually there exists a mapping better than cross product
- $D^d_{\boldsymbol{A}}$ usually better than D

Adaptive distances vs. standard distances

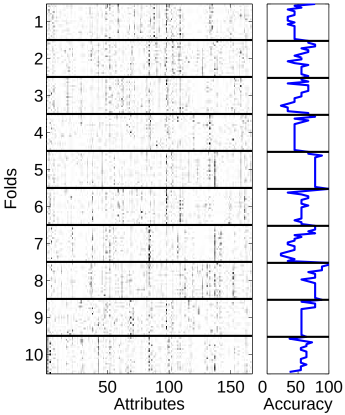
		musk1	musk2	muta	fm	mm	tiger	fox	eleph.
D^{AL}	D	76.4	71.0	57.6	58.6	63.0	72.0	52.0	76.0
	$D^f_{\boldsymbol{A}}$	75.9	71.5	58.2	50.1-	53.5-	74.0	56.5	79.5
	$D^d_{\boldsymbol{A}}$	78.0	76.5	**68.4+**	40.4-	48.1-	**78.5+**	**59.5+**	77.0
D^{SL}	D	81.1	77.2	42.2	41.6	39.3	76.0	56.5	71.5
	$D^f_{\boldsymbol{A}}$	78.4	69.5	41.6	41.6	39.3	73.5	57.5	72.5
	$D^d_{\boldsymbol{A}}$	82.4	68.5-	42.8	41.6	39.3	79.0	61.5+	72.5
D^{CL}	D	67.5	71.5	39.4	43.2	40.9	67.5	54.5	64.5
	$D^f_{\boldsymbol{A}}$	71.7	65.5	56.3+	47.3	**46.6+**	68.5	47.0	55.5
	$D^d_{\boldsymbol{A}}$	**77.3+**	**79.5+**	**58.2+**	**52.1+**	**58.1+**	72.5	51.5	74.0+
D^{H}	D	81.3	78.0	75.5	46.8	48.3	74.0	51.5	74.5
	$D^f_{\boldsymbol{A}}$	83.5	73.3	77.5	**57.5+**	**57.2+**	74.0	55.5	65.5
	$D^d_{\boldsymbol{A}}$	78.8	77.2	**80.6+**	**57.2+**	**53.8+**	76.5	**65.5+**	75.5
D^{SMD}	D	83.3	75.2	79.4	56.6	58.2	77.5	60.0	79.5
	$D^f_{\boldsymbol{A}}$	77.1	73.3	**81.4+**	58.5	58.3	76.0	59.0	77.5
	$D^d_{\boldsymbol{A}}$	81.3	73.3	80.5	56.6	58.7	**85.0+**	62.0	**85.5+**

- Usually there exists a mapping better than cross product
- $D^d_{\boldsymbol{A}}$ usually better than D
- $D^f_{\boldsymbol{A}}$ are not as good

Non-convexity of NCA

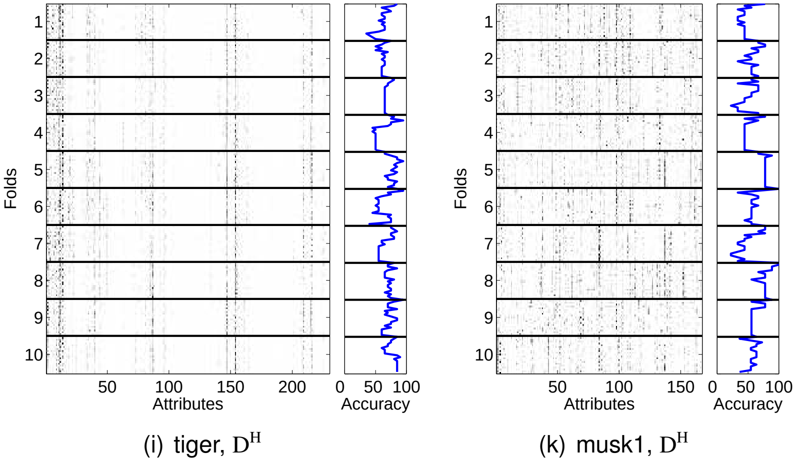


(e) tiger, D^H



(g) musk1, D^H

Non-convexity of NCA



- Local optimas are present

Outline

Conclusions and Open Problems

Conclusions and Open Problems

Pros:

- Practical problem with many applications
- General framework

Conclusions and Open Problems

Pros:

- Practical problem with many applications
- General framework

Cons:

- Empirical results somehow inconclusive
- Could be slow for large sets / datasets

Conclusions and Open Problems

Pros:

- Practical problem with many applications
- General framework

Cons:

- Empirical results somehow inconclusive
- Could be slow for large sets / datasets

Future research:

- Directly learn $\mathcal{F}$ and $\mathcal{A}$ in

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\}$$

Thank you!