A Novel Uncertainty Sampling Algorithm for Cost-sensitive Multiclass Active Learning

Kuan-Hao Huang[1] and Hsuan-Tien Lin[1,2]

[1]Department of Computer Science & Information Engineering
National Taiwan University

[2]Appier Inc.

Appier, November 24, 2016

Outline

- Problem Introduction
- Proposed Algorithm
- Experiments
- Conclusion

Outline

- Problem Introduction
- Proposed Algorithm
- Experiments
- Conclusion

Multiclass Classification (MCC)

Multiclass classification

- learning from lots of labeled data
- example: animal recognition

image						
label	dog	cat	dog	rabbit	cat	rabbit

Multiclass Classification (MCC)

Multiclass classification

- learning from lots of labeled data
- example: animal recognition

image						
label	dog	cat	dog	rabbit	cat	rabbit

labeling is expensive!

Active Learning for Multiclass Classification

Labeled pool

image						
label	dog	cat	dog	rabbit	cat	rabbit

Unlabeled pool

image						
label						

Active Learning for Multiclass Classification

Labeled pool

image						
label	dog	cat	dog	rabbit	cat	rabbit

Unlabeled pool

image						
label		cat				

Active Learning for Multiclass Classification

Labeled pool

image						
label	dog	cat	dog	rabbit	cat	rabbit

Unlabeled pool

image						
label		cat	rabbit			

Active Learning for Multiclass Classification

Labeled pool

image						
label	dog	cat	dog	rabbit	cat	rabbit

Unlabeled pool

image						
label		cat	rabbit		dog	

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Pool-based active learning

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$ and unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Pool-based active learning

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$ and unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$
- for round $t = 1, 2, ..., T$
  - use **querying strategy** to query $\mathbf{x}_s$ in unlabeled pool $\mathcal{D}_u$ and get the label y_s

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Pool-based active learning

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$ and unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$
- for round $t = 1, 2, ..., T$
  - use **querying strategy** to query $\mathbf{x}_s$ in unlabeled pool $\mathcal{D}_u$ and get the label y_s
  - move $(\mathbf{x}_s, y_s)$ from unlabeled pool $\mathcal{D}_u$ to labeled pool $\mathcal{D}_l$

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Pool-based active learning

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$ and unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$
- for round $t = 1, 2, ..., T$
  - use **querying strategy** to query $\mathbf{x}_s$ in unlabeled pool $\mathcal{D}_u$ and get the label y_s
  - move $(\mathbf{x}_s, y_s)$ from unlabeled pool $\mathcal{D}_u$ to labeled pool $\mathcal{D}_l$
  - learn a **classifier** $f^{(t)}$ from the current label pool $\mathcal{D}_l$

Active Learning

Notation

- feature (image): $\mathbf{x} \in \mathbb{R}^d$
- label: $y \in \{c_1, c_2, ..., c_K\}$

Pool-based active learning

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$ and unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$
- for round $t = 1, 2, ..., T$
  - use **querying strategy** to query $\mathbf{x}_s$ in unlabeled pool $\mathcal{D}_u$ and get the label y_s
  - move $(\mathbf{x}_s, y_s)$ from unlabeled pool $\mathcal{D}_u$ to labeled pool $\mathcal{D}_l$
  - learn a **classifier** $f^{(t)}$ from the current label pool $\mathcal{D}_l$
- improve the **performance** of $f^{(t)}$ with respect to **#queries**

Querying Strategies

Uncertainty sampling

- query most **uncertain** $\mathbf{x}$
- distance, entropy, least confidence [Tong et al. 2001; Jing et al., 2004; Culotta et al., 2005]

Querying Strategies

Uncertainty sampling

- query most **uncertain** $\mathbf{x}$
- distance, entropy, least confidence [Tong et al. 2001; Jing et al., 2004; Culotta et al., 2005]

Representative sampling

- query most **representative** $\mathbf{x}$
- information density, QUIRE, cluster [Settles et al., 2008; Huang et al., 2014; Xu et al., 2003]

Querying Strategies

Uncertainty sampling

- query most **uncertain** $\mathbf{x}$
- distance, entropy, least confidence [Tong et al. 2001; Jing et al., 2004; Culotta et al., 2005]

Representative sampling

- query most **representative** $\mathbf{x}$
- information density, QUIRE, cluster [Settles et al., 2008; Huang et al., 2014; Xu et al., 2003]

Expected error reduction

- query most **helpful** $\mathbf{x}$
- error reduction [Roy et al., 2001]

Querying Strategies

Uncertainty sampling

- query most **uncertain** $\mathbf{x}$
- distance, entropy, least confidence [Tong et al. 2001; Jing et al., 2004; Culotta et al., 2005]

Representative sampling

- query most **representative** $\mathbf{x}$
- information density, QUIRE, cluster [Settles et al., 2008; Huang et al., 2014; Xu et al., 2003]

Expected error reduction

- query most **helpful** $\mathbf{x}$
- error reduction [Roy et al., 2001]

this work focuses on **uncertainty sampling**

Evaluation Criteria

Regular (Error rate)

actual \ predicted	healthy	cold	Zika
healthy	0	1	1
cold	1	0	1
Zika	1	1	0

- **same** misclassified penalties
- most common criterion

Evaluation Criteria

Regular (Error rate)

actual \ predicted	healthy	cold	Zika
healthy	0	1	1
cold	1	0	1
Zika	1	1	0

- **same** misclassified penalties
- most common criterion

Cost matrix

actual \ predicted	healthy	cold	Zika
healthy	0	10	50
cold	200	0	100
Zika	**1000**	800	0

- **different** misclassified penalties
- cost matrix $\mathbf{C}$
- $\mathbf{C}_{i,j}$: predict c_i as c_j

Cost-sensitive Active Learning Algorithms

Cost-sensitive algorithms

- cost-sensitive algorithms take cost matrix $\mathbf{C}$ into account

Cost-sensitive Active Learning Algorithms

Cost-sensitive algorithms

- cost-sensitive algorithms take cost matrix $\mathbf{C}$ into account

	query strategy	classifier f
regular algorithms	by f, $\mathcal{D}_l$, and $\mathcal{D}_u$	learned from $\mathcal{D}_l$
cost-sensitive algorithms	by f, $\mathcal{D}_l$, $\mathcal{D}_u$, and $\mathbf{C}$	learned from $\mathcal{D}_l$ and $\mathbf{C}$

Cost-sensitive Active Learning Algorithms

Cost-sensitive algorithms

- cost-sensitive algorithms take cost matrix $\mathbf{C}$ into account

	query strategy	classifier f
regular algorithms	by f, $\mathcal{D}_l$, and $\mathcal{D}_u$	learned from $\mathcal{D}_l$
cost-sensitive algorithms	by f, $\mathcal{D}_l$, $\mathcal{D}_u$, and $\mathbf{C}$	learned from $\mathcal{D}_l$ and $\mathbf{C}$

Goal of our work

- **cost-sensitive uncertainty sampling algorithms**

Cost-sensitive Active Learning Algorithms

Cost-sensitive algorithms

- cost-sensitive algorithms take cost matrix $\mathbf{C}$ into account

	query strategy	classifier f
regular algorithms	by f, $\mathcal{D}_l$, and $\mathcal{D}_u$	learned from $\mathcal{D}_l$
cost-sensitive algorithms	by f, $\mathcal{D}_l$, $\mathcal{D}_u$, and $\mathbf{C}$	learned from $\mathcal{D}_l$ and $\mathbf{C}$

Goal of our work

- **cost-sensitive uncertainty sampling algorithms**

	regular	cost-sensitive
probabilistic uncertainty	well-studied	known [Chen et al., 2013]
non-probabilistic uncertainty	well-studied	ongoing (our work)

Outline

- Problem Introduction
- Proposed Algorithm
- Experiments
- Conclusion

Cost-sensitive Active Learning

Main tasks

- **query strategy**: non-probabilistic cost-sensitive uncertainty
- **classifier** f: non-probabilistic cost-sensitive multiclass classifier

Cost-sensitive Active Learning

Main tasks

- **query strategy**: non-probabilistic cost-sensitive uncertainty
- **classifier** f: non-probabilistic cost-sensitive multiclass classifier

Difficulty in non-probabilistic uncertainty

- **one-versus-all view** and **one-versus-one view**
- multiple boundaries

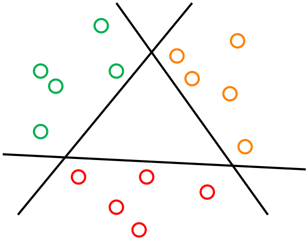
Cost-sensitive Active Learning

Main tasks

- **query strategy**: non-probabilistic cost-sensitive uncertainty
- **classifier** f: non-probabilistic cost-sensitive multiclass classifier

Difficulty in non-probabilistic uncertainty

- **one-versus-all view** and **one-versus-one view**
- multiple boundaries



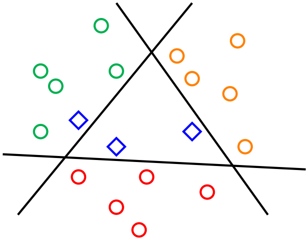
Cost-sensitive Active Learning

Main tasks

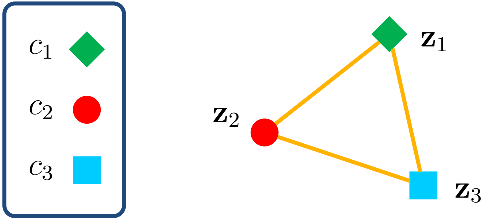
- **query strategy**: non-probabilistic cost-sensitive uncertainty
- **classifier** f: non-probabilistic cost-sensitive multiclass classifier

Difficulty in non-probabilistic uncertainty

- **one-versus-all view** and **one-versus-one view**
- multiple boundaries



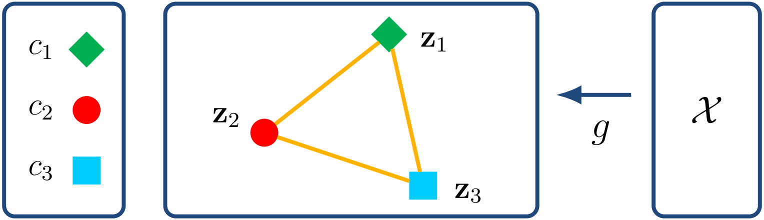
Embedding View for MCC (Training)



Training stage

- for classes $c_1, c_2, ..., c_K$, find K **hidden points** $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$ with **equal distances**

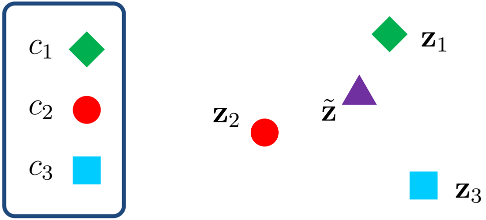
Embedding View for MCC (Training)



Training stage

- for classes $c_1, c_2, ..., c_K$, find K **hidden points** $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$ with **equal distances**
- learn a **regressor** g from $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^{N_l}$

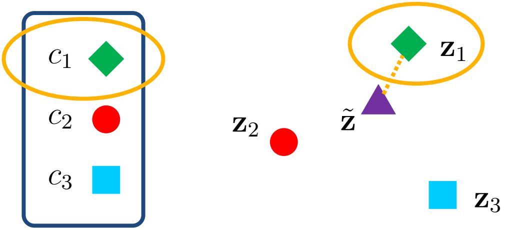
Embedding View for MCC (Predicting)



Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$

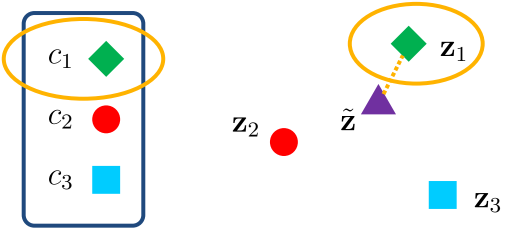
Embedding View for MCC (Predicting)



Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$
- find the **nearest hidden point** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- take the corresponding class as the prediction

Embedding View for MCC (Predicting)

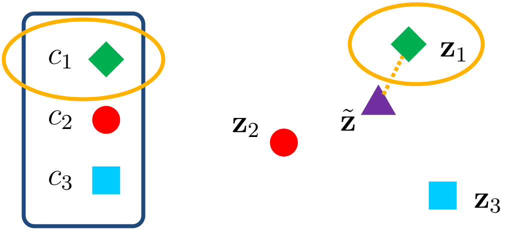


Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$
- find the **nearest hidden point** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- take the corresponding class as the prediction

equivalent to one-versus-all scenario when g is linear

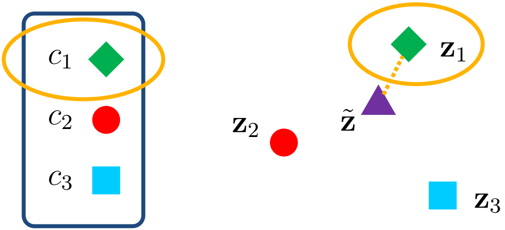
Cost Information



Embedding cost information

- get prediction by nearest neighbor (smallest distance)

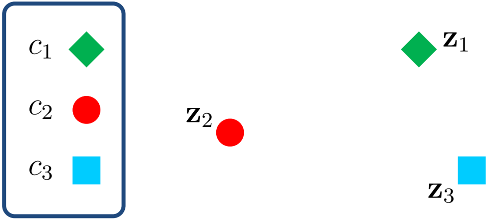
Cost Information



Embedding cost information

- get prediction by nearest neighbor (smallest distance)
- embed cost information in distance

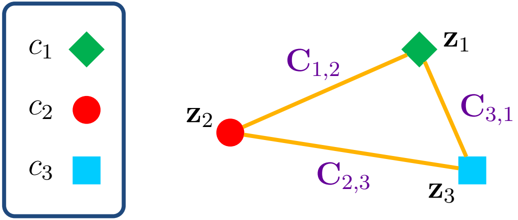
Cost Embedding (Training)



Training stage

- for classes $c_1, c_2, ..., c_K$, find K **hidden points** $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$

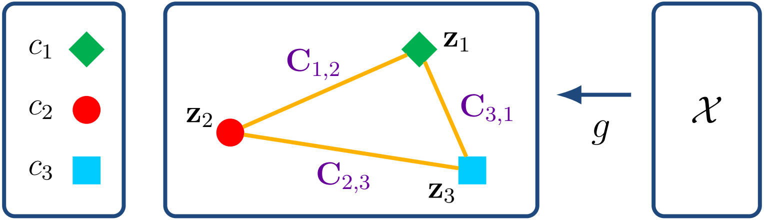
Cost Embedding (Training)



Training stage

- for classes $c_1, c_2, ..., c_K$, find K **hidden points** $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- **higher (lower) cost $\mathrm{C}_{i,j} \Leftrightarrow$ larger (smaller) distance $d(\mathbf{z}_i, \mathbf{z}_j)$**
- preserve the **order** of the costs

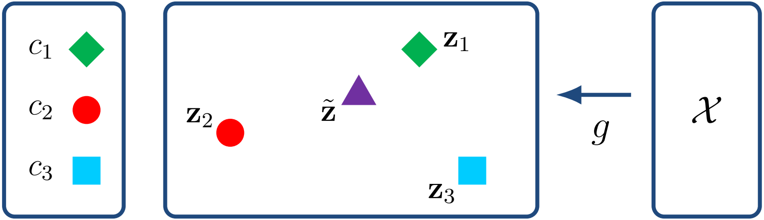
Cost Embedding (Training)



Training stage

- for classes $c_1, c_2, ..., c_K$, find K **hidden points** $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- **higher (lower) cost $\mathrm{C}_{i,j}$ $\Leftrightarrow$ larger (smaller) distance $d(\mathbf{z}_i, \mathbf{z}_j)$**
- preserve the **order** of the costs
- learn a **regressor** g from $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^{N_l}$

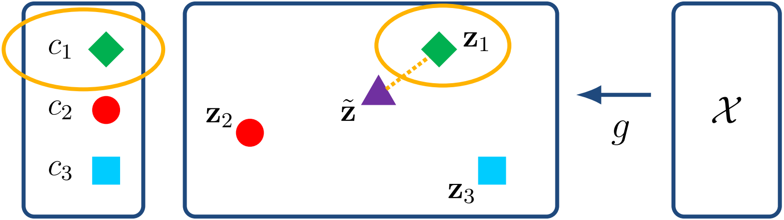
Cost Embedding (Predicting)



Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$

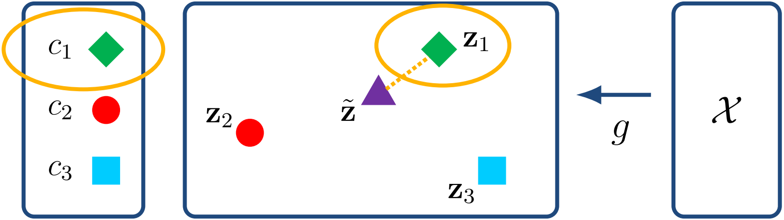
Cost Embedding (Predicting)



Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$
- find the **nearest hidden point** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- take the corresponding class as the **cost-sensitive prediction**

Cost Embedding (Predicting)



Predicting stage

- for a testing instance $\mathbf{x}$, get the **predicted hidden point** $\tilde{\mathbf{z}} = g(\mathbf{x})$
- find the **nearest hidden point** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K$
- take the corresponding class as the **cost-sensitive prediction**

how to preserve the **order** of the costs?

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Goal of non-metric multidimensional scaling

- L objects $O_1, O_2, ..., O_L$
- symmetric dissimilarity matrix $\mathbf{\Delta}$: $\mathbf{\Delta}_{i,j}$ for dissimilarity of O_i and O_j

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Goal of non-metric multidimensional scaling

- L objects $O_1, O_2, ..., O_L$
- symmetric dissimilarity matrix $\mathbf{\Delta}$: $\mathbf{\Delta}_{i,j}$ for dissimilarity of O_i and O_j
- find target points $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L$ with $\mathbf{\Delta}_{i,j} < \mathbf{\Delta}_{i',j'} \Leftrightarrow d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_{i'}, \mathbf{u}_{j'})$

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Goal of non-metric multidimensional scaling

- L objects $O_1, O_2, ..., O_L$
- symmetric dissimilarity matrix $\mathbf{\Delta}$: $\mathbf{\Delta}_{i,j}$ for dissimilarity of O_i and O_j
- find target points $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L$ with $\mathbf{\Delta}_{i,j} < \mathbf{\Delta}_{i',j'} \Leftrightarrow d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_{i'}, \mathbf{u}_{j'})$

Goal of cost embedding

- K classes $c_1, c_2, ..., c_K$
- cost matrix $\mathbf{C}$: $\mathbf{C}_{i,j}$ for cost of predicting c_i as c_j

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Goal of non-metric multidimensional scaling

- L objects $O_1, O_2, ..., O_L$
- symmetric dissimilarity matrix $\mathbf{\Delta}$: $\mathbf{\Delta}_{i,j}$ for dissimilarity of O_i and O_j
- find target points $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L$ with $\mathbf{\Delta}_{i,j} < \mathbf{\Delta}_{i',j'} \Leftrightarrow d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_{i'}, \mathbf{u}_{j'})$

Goal of cost embedding

- K classes $c_1, c_2, ..., c_K$
- cost matrix $\mathbf{C}$: $\mathbf{C}_{i,j}$ for cost of predicting c_i as c_j
- find hidden points $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L$ with $\mathbf{C}_{i,j} < \mathbf{C}_{i',j'} \Leftrightarrow d(\mathbf{z}_i, \mathbf{z}_j) < d(\mathbf{z}_{i'}, \mathbf{z}_{j'})$

Non-metric Multidimensional Scaling (NMDS)

Non-metric multidimensional scaling (NMDS)

- classic technique in manifold learning

Goal of non-metric multidimensional scaling

- L objects $O_1, O_2, ..., O_L$
- symmetric dissimilarity matrix $\boldsymbol{\Delta}$: $\boldsymbol{\Delta}_{i,j}$ for dissimilarity of O_i and O_j
- find target points $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L$ with $\boldsymbol{\Delta}_{i,j} < \boldsymbol{\Delta}_{i',j'} \Leftrightarrow d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_{i'}, \mathbf{u}_{j'})$

Goal of cost embedding

- K classes $c_1, c_2, ..., c_K$
- cost matrix $\mathbf{C}$: $\mathbf{C}_{i,j}$ for cost of predicting c_i as c_j
- find hidden points $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L$ with $\mathbf{C}_{i,j} < \mathbf{C}_{i',j'} \Leftrightarrow d(\mathbf{z}_i, \mathbf{z}_j) < d(\mathbf{z}_{i'}, \mathbf{z}_{j'})$

asymmetric cost $\mathbf{C}$ ($\mathbf{C}_{i,j} \neq \mathbf{C}_{j,i}$) vs. **symmetric dissimilarity** $\boldsymbol{\Delta}$

Asymmetry of Cost Matrix

Asymmetric cost

- $\mathbf{C}_{i,j} \neq \mathbf{C}_{j,i}$

Asymmetry of Cost Matrix

Asymmetric cost

- $\mathbf{C}_{i,j} \neq \mathbf{C}_{j,i}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction

Asymmetry of Cost Matrix

Asymmetric cost

- $\mathbf{C}_{i,j} \neq \mathbf{C}_{j,i}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction
- $\mathbf{C}_{j,i} \Rightarrow$ cost when c_i is prediction and c_j is ground truth

Asymmetry of Cost Matrix

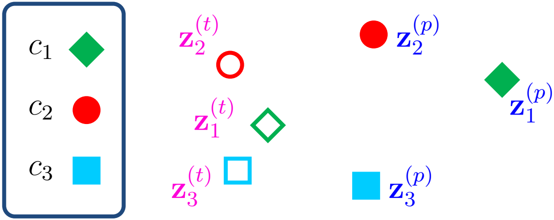
Asymmetric cost

- $\mathbf{C}_{i,j} \neq \mathbf{C}_{j,i}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction
- $\mathbf{C}_{j,i} \Rightarrow$ cost when c_i is prediction and c_j is ground truth

Two roles of classes

- two roles of class c_i: ground truth role and prediction role
- embed cost information in these two roles

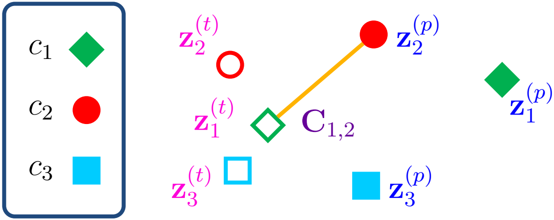
Mirroring Trick



Two roles of class

- two roles of class c_i: ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$

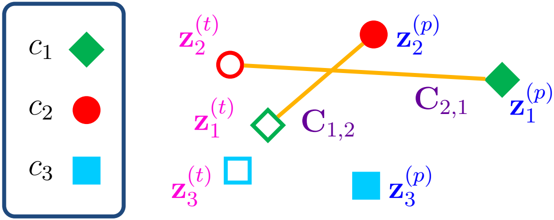
Mirroring Trick



Two roles of class

- two roles of class c_i: ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction $\Rightarrow$ for $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(p)}$

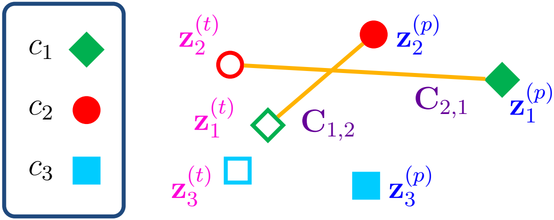
Mirroring Trick



Two roles of class

- two roles of class c_i: ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction $\Rightarrow$ for $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(p)}$
- $\mathbf{C}_{j,i} \Rightarrow$ cost when c_i is prediction and c_j is ground truth $\Rightarrow$ for $\mathbf{z}_i^{(p)}$ and $\mathbf{z}_j^{(t)}$

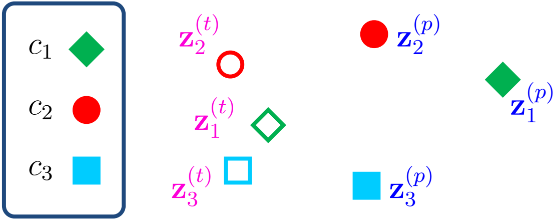
Mirroring Trick



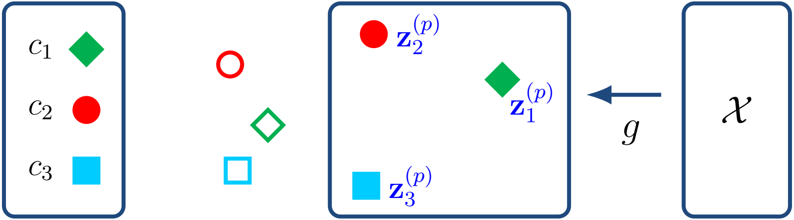
Two roles of class

- two roles of class c_i: ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$
- $\mathbf{C}_{i,j} \Rightarrow$ cost when c_i is ground truth and c_j is prediction $\Rightarrow$ for $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(p)}$
- $\mathbf{C}_{j,i} \Rightarrow$ cost when c_i is prediction and c_j is ground truth $\Rightarrow$ for $\mathbf{z}_i^{(p)}$ and $\mathbf{z}_j^{(t)}$
- cost information is embedded in **distance** between ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$

Mirroring Trick



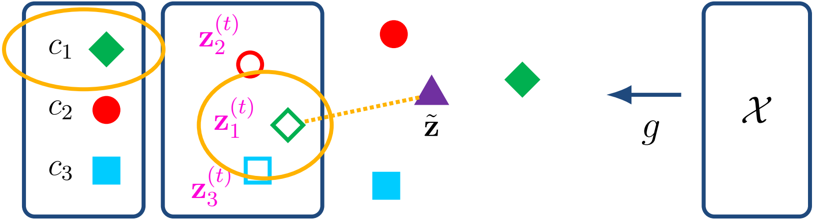
Mirroring Trick



Training stage

- $\tilde{\mathbf{z}} = g(\mathbf{x})$ learned from $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^{N} \Rightarrow$ prediction role
- learn **regressor** g from $\mathbf{z}_1^{(p)}, \mathbf{z}_2^{(p)}, \ldots, \mathbf{z}_K^{(p)}$

Mirroring Trick



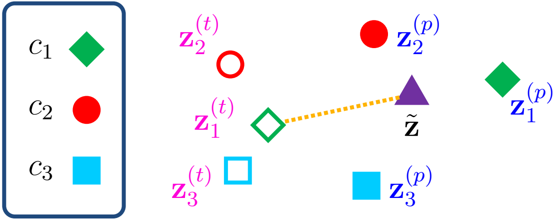
Training stage

- $\tilde{\mathbf{z}} = g(\mathbf{x})$ learned from $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^{N} \Rightarrow$ prediction role
- learn **regressor** g from $\mathbf{z}_1^{(p)}, \mathbf{z}_2^{(p)}, \ldots, \mathbf{z}_K^{(p)}$

Predicting stage

- nearest hidden point of $\tilde{\mathbf{z}} \Rightarrow$ ground truth role
- find the **nearest hidden point** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, \ldots, \mathbf{z}_K^{(t)}$

Cost-sensitive Uncertainty



Cost-sensitive Uncertainty

- nearest hidden point with large distance $\Rightarrow$ uncertain prediction
- **cost-sensitive uncertainty**: distance between nearest hidden point and predicted hidden point $\tilde{\mathbf{z}}$

Active Learning with Cost Embedding (ALCE)

Active Learning with Cost Embedding (ALCE)

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$, unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$, and cost matrix $\mathbf{C}$

Active Learning with Cost Embedding (ALCE)

Active Learning with Cost Embedding (ALCE)

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$, unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$, and cost matrix $\mathbf{C}$
- obtain **two roles of hidden points** from cost matrix $\mathbf{C}$ by NMDS

Active Learning with Cost Embedding (ALCE)

Active Learning with Cost Embedding (ALCE)

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$, unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$, and cost matrix $\mathbf{C}$
- obtain **two roles of hidden points** from cost matrix $\mathbf{C}$ by NMDS
- for round $t = 1, 2, ..., T$
  - select $\mathbf{x}_s$ in $\mathcal{D}_u$ with highest **cost-sensitive uncertainty** to query the label y_s

Active Learning with Cost Embedding (ALCE)

Active Learning with Cost Embedding (ALCE)

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$, unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$, and cost matrix $\mathbf{C}$
- obtain **two roles of hidden points** from cost matrix $\mathbf{C}$ by NMDS
- for round $t = 1, 2, ..., T$
  - select $\mathbf{x}_s$ in $\mathcal{D}_u$ with highest **cost-sensitive uncertainty** to query the label y_s
  - move $(\mathbf{x}_s, y_s)$ from unlabeled pool $\mathcal{D}_u$ to labeled pool $\mathcal{D}_l$

Active Learning with Cost Embedding (ALCE)

Active Learning with Cost Embedding (ALCE)

- labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N_l}$, unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$, and cost matrix $\mathbf{C}$
- obtain **two roles of hidden points** from cost matrix $\mathbf{C}$ by NMDS
- for round $t = 1, 2, ..., T$
  - select $\mathbf{x}_s$ in $\mathcal{D}_u$ with highest **cost-sensitive uncertainty** to query the label y_s
  - move $(\mathbf{x}_s, y_s)$ from unlabeled pool $\mathcal{D}_u$ to labeled pool $\mathcal{D}_l$
  - learn a **classifier** $f^{(t)}$ from the current label pool $\mathcal{D}_l$ by **cost embedding**

Outline

- Problem Introduction
- Proposed Algorithm
- Experiments
- Conclusion

Experiments

Settings

- 20 runs of experiments
- 60% data as training set and 40% data as testing set
- randomly select one instance of each class in the training set as the initial labeled pool $\mathcal{D}_l$
- $\mathbf{C}_{i,j}$ is uniformly sample from $\left[0, 2000\frac{|\{n:y^{(n)}=i\}|}{|\{n:y^{(n)}=j\}|}\right]$ [Beygelzimer et al., 2005]

Experiments

Settings

- 20 runs of experiments
- 60% data as training set and 40% data as testing set
- randomly select one instance of each class in the training set as the initial labeled pool $\mathcal{D}_l$
- $\mathbf{C}_{i,j}$ is uniformly sample from $\left[0, 2000\frac{|\{n:y^{(n)}=i\}|}{|\{n:y^{(n)}=j\}|}\right]$ [Beygelzimer et al., 2005]

List of Experiments

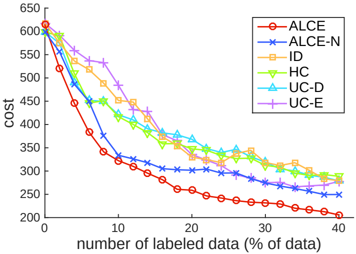
- comparison with cost-insensitive algorithms
- comparison with cost-sensitive algorithms
- analysis of the dimension of embedded space

Comparison with Cost-insensitive Algorithms

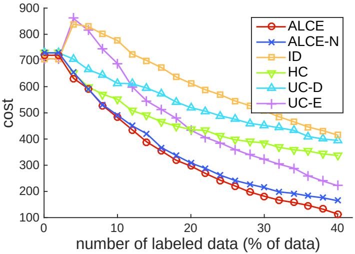
- ID, HC, UC-D, UC-E: their querying strategies + kernel SVM
- ALCE-N: proposed querying strategy + kernel SVM
- ALCE: proposed querying strategy + cost embedding (kernel)

Comparison with Cost-insensitive Algorithms

- ID, HC, UC-D, UC-E: their querying strategies + kernel SVM
- ALCE-N: proposed querying strategy + kernel SVM
- ALCE: proposed querying strategy + cost embedding (kernel)



(a) vehicle



(b) vowel

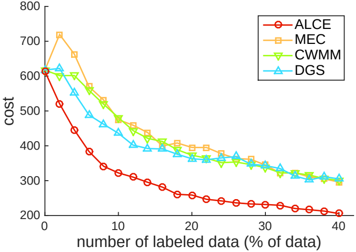
- ALCE-N outperforms ID, HC, UC-D, UC-E ⇒ querying strategy is useful
- ALCE outperforms ALCE-N ⇒ cost embedding is useful

Comparison with Cost-sensitive Algorithms

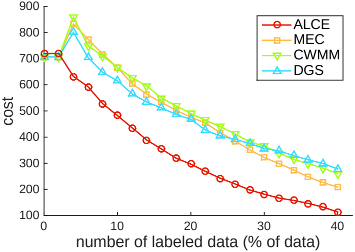
- MEC, CWMM, DGS: probabilistic uncertainty + kernel classifier
- ALCE: non-probabilistic uncertainty + cost embedding (kernel)

Comparison with Cost-sensitive Algorithms

- MEC, CWMM, DGS: probabilistic uncertainty + kernel classifier
- ALCE: non-probabilistic uncertainty + cost embedding (kernel)



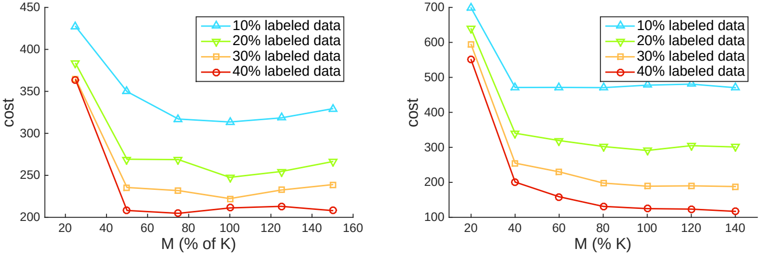
(a) vehicle



(b) vowel

- ALCE outperforms MEC, CWMM, DGS

Dimension of Embedded Space



(a) vehicle (b) vowel

- setting dimension of embedded space M as $60\%K$ is sufficient

Outline

Conclusion

- propose **active learning with cost embedding (ALCE)**
  - embedding view for cost-sensitive multiclass classification
  - embed cost information in distance by non-metric multidimensional scaling
  - mirroring trick for asymmetric cost matrix
  - define cost-sensitive uncertainty by distance
- **promising performance** of ALCE compared with state-of-the-art cost-sensitive active learning algorithms

Conclusion

- propose **active learning with cost embedding (ALCE)**
  - embedding view for cost-sensitive multiclass classification
  - embed cost information in distance by non-metric multidimensional scaling
  - mirroring trick for asymmetric cost matrix
  - define cost-sensitive uncertainty by distance
- **promising performance** of ALCE compared with state-of-the-art cost-sensitive active learning algorithms

Thank you! Any question?