FAMER: Making Multi-Instance Learning Better and Faster

Wei Ping*, Ye Xu§, Jianyong Wang*, Xian-Sheng Hua◊

* Tsinghua University

§Dartmouth College

◊Microsoft Research Asia

Outline

- Background
- Thinking in MIL (Multi-Instance Learning) Kernels
- Description of proposed FAMER (FAst kernel for Multi-instancE leaRning)
- Experiment
- Conclusion

Background: Multi-Instance Learning

- Multi-Instance Learning:
  - Training example is bag containing instances.
  - Bag label is known, but instance label is unseen
  - Bag is positive iff it contains at least one positive instance; otherwise it is negative.

Background: Multi-Instance Learning

- Two perspectives for solving MIL
  - Instance level
    - Eliminate the uncertainty of instance label in positive bag
    - Learn a model in instance space
  - Bag level
    - View each bag as entity
    - Learn a model for bags.

Background: MIL Kernels

- MIL Kernel is bag level method
  - A powerful family of method in MIL
  - Kernel value between bags
  - Some typical MIL Kernels:
    - NSK and STK [Gartner, *ICML02*]
    - Marginalized Kernel [Kwok, *IJCAI07*]
    - PPMM Kernel [Wang, *ICML08*]
    - MI-Graph and mi-Graph [Zhou, *ICML09*]
    - ……

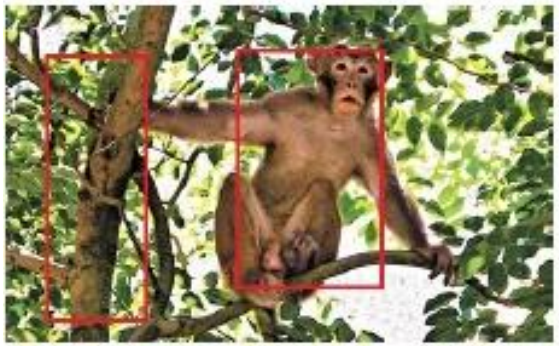
Outline

- Background
- **Thinking in MIL Kernels**
- Description of proposed FAMER
- Experiment
- Conclusion

Thinking in MIL Kernel

- Shortcomings of current MIL Kernels:
  - Correspondence information;

  - Co-occurrence information;



  - Assumption: all instances within a bag have equal weight (key instances)
  - Heavy computing load(pair-wise manner).

Thinking in MIL Kernel: Correspondence

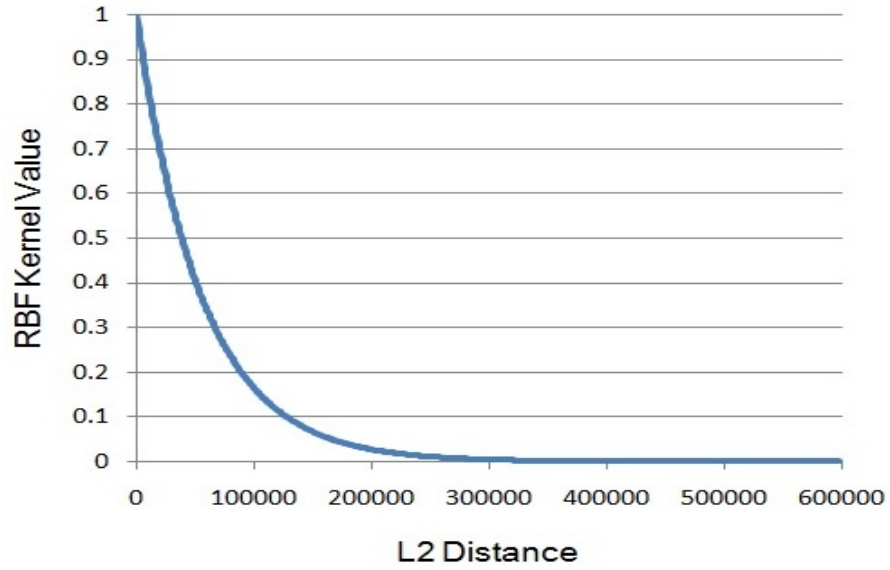
- The **correspondence** relationship between these two image bags (i.e., the respective coast, sea and sky segmentations/instances) indicates a high similarity.

Thinking in MIL Kernel: Correspondence

- NSK [Gartner, *ICML02*] gets decent results for most of real-world datasets

$$K(X_i, X_j) = \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} k(\boldsymbol{x}_{ia}, \boldsymbol{y}_{jb})$$

  - Therein, $k(x_{ia}, y_{jb}) = exp(-\gamma ||x_{ia} - y_{jb}||^2)$



- Exponential amplification: the relation between RBF Kernel Value and *L2 distance using the empirical rule of thumb* indicated in [8]

Thinking in MIL Kernel: Correspondence

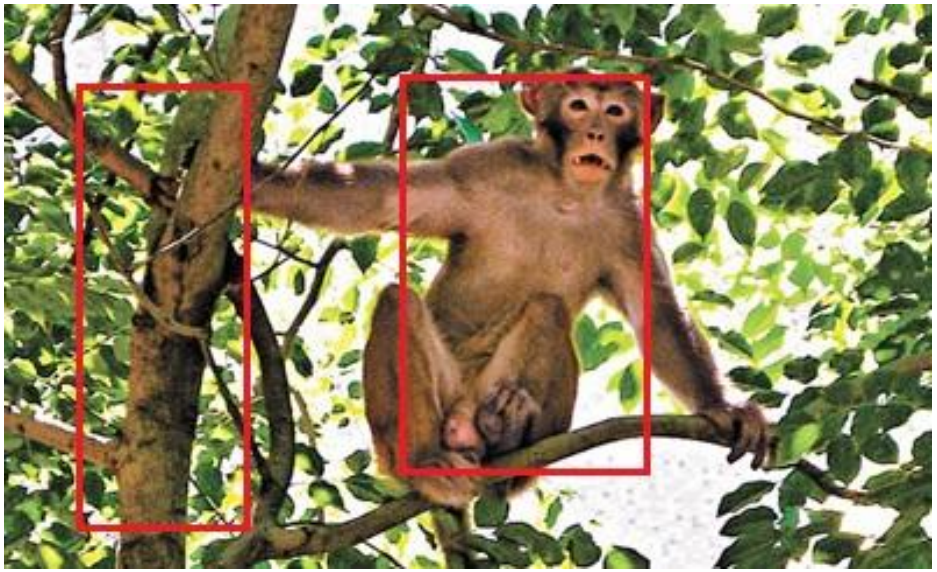
- Classification Accuracy of NSK using linear kernel vs. RBF kernel

Data set	Musk1	Musk2	Elept	Fox	Tiger
NSK_{Linear}	86.1%	82.4%	79.0%	55.5%	77.3%
NSK_{RBF}	**88.0%**	**89.3%**	**84.3%**	**60.3%**	**84.2%**

- The exponential amplification of RBF result in soft correspondence scheme: similar pair of instances make much more significant contribution for bag kernel

Thinking in MIL Kernel: Co-occurrence

- Co-occurrence Information (Non. I.I.D [Zhou, *ICML09*]) :
  - In image bag, instances(segmentations) containing monkeys are very likely to co-occur with those instances containing trees.



Outline

- Background
- Thinking in MIL Kernel
- **Description of proposed FAMER**
- Experiment
- Conclusion

FAMER: Preliminaries for LSH

- Locality Sensitive Hashing is a random distribution on a family ***H*** *of hash functions*, such that the probability of two object being mapped to the same value reflects the similarity between them:

$$Pr_{h\in\mathcal{H}}[h(\boldsymbol{x}) = h(\boldsymbol{y})] = s_{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.1}$$

- Given an arbitrary locally sensitive hash function h∈ ***H****, we define the corresponding induced similarity* measure as:

$$s_h(\boldsymbol{x}, \boldsymbol{y}) = \delta_{h(\boldsymbol{x}),h(\boldsymbol{y})} \tag{3.2}$$

  - Therein, δ is Kronecker delta. From (3.1) and (3.2),

$$E_{h\in\mathcal{H}}[s_h(\boldsymbol{x}, \boldsymbol{y})] = s_{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.3}$$

FAMER: Construct Similarity Metric with Correspondence

- In order to capture correspondence
  - A LSH based similarity metric: by analogy from NSK,

(3.4)
$$S_{MI,\mathcal{H}}(X_i, X_j) = \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} exp(-\gamma(1 - s_{\mathcal{H}}(x_{ia}, x_{jb})))$$

    - Therein, $s_{\mathcal{H}}(x_{ia}, y_{jb})$ is induced from LSH family ***H*** as (3.1)

FAMER : How to compute (3.4) $S_{MI,\mathcal{H}}(X_i, X_j)$

- For each h$\in$ ***H***, we define

(3.5) $S_{MI,h}(X_i, X_j)$

$$= \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} exp(-\gamma(1 - s_h(x_{ia}, y_{jb})))$$

- We could use $E_{h\in\mathcal{H}}[S_{MI,h}(X_i, X_j)]$ to approximate $S_{MI,\mathcal{H}}(X_i, X_j)$
  - **Lemma**: (3.4) $S_{MI,\mathcal{H}}(X_i, X_j)$ is the lower bound of $E_{h\in\mathcal{H}}[S_{MI,h}(X_i, X_j)]$

FAMER: Efficient Computing

- Pair-wise manner is still used in (3.5): time consuming.
- Use embedding histogram to reduce the time complexity: $\mathbf{T}_h(X_i) = \sum_{x_{ij} \in X_i} \mathbf{e}[h(x_{ij})]$
  - No longer compute in pair-wise manner

(3.5)
$$
\begin{aligned}
& S_{MI,h}(X_i, X_j) \\
&= \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{a=1}^{n_j} exp(-\gamma(1 - s_h(x_{ia}, y_{jb}))) \\
&= \frac{1}{n_i n_j} \{ \mathbf{T}_h(X_i) \cdot \mathbf{T}_h(X_j) + [n_i n_j - \mathbf{T}_h(X_i) \cdot \mathbf{T}_h(X_j)] \\
&\quad * exp(-\gamma) \}
\end{aligned}
$$

FAMER: Weighting Scheme

- Imposing high weights on “key” instances
- Motivation:
  - Instances mapped in the same bin; similar instances under some specific probabilities.
  - Each bin in the histogram corresponds to a probabilistic region in instance space
  - As DD pointed, regions with many negative and few “positive” instance, have much discriminative information for negative bags, and vice versa

FAMER: Weighting Scheme

- ΔEntropy is employed to evaluate the "purity" of bin(region), i.e. the weight of each bin[i].

$$P_+[i] = \frac{N_+[i] + \epsilon}{N_+[i] + N_-[i] + 2\epsilon} \quad P_-[i] = \frac{N_-[i] + \epsilon}{N_+[i] + N_-[i] + 2\epsilon}$$

$$W[i] = \Delta H[i] + C = H_0[i] - H_e[i] + C$$

$$= ln2 - (P_+[i]\, ln\frac{1}{P_+[i]} + P_-[i]\, ln\frac{1}{P_-[i]}) + C$$

- In this way, instances within each bin are implicitly weighted.

FAMER: Weighting Scheme

- False Positive problem:
  - In practice, we regard all instances in positive bags as “positive” when estimating the probability
  - Influence of those false positive(negative) instances in positive bags, can be offset by the negative instances from corresponding negative regions indicated by negative bags.

FAMER: Weighting Scheme captures Co-occurrence

- Weighting Scheme also detect the co-occurrence information
  - The histogram naturally records the co-occurrence information of "positive"/negative instances.
  - The joint statistics respects the co-occurrence relations.

FAMER: Final Form

- Final form of FAMER:

(3.6)
$$K_{MI,H_M}(X_i, X_j) = \frac{1}{M}\frac{1}{n_i n_j}\{ (\mathbf{W} \circ \mathbf{T}_{H_M}(X_i)) \cdot \mathbf{T}_{H_M}(X_j) + [M n_i n_j - (\mathbf{W} \circ \mathbf{T}_{H_M}(X_i)) \cdot \mathbf{T}_{H_M}(X_j)] * exp(-\gamma) \}$$

  - Therein, M is the number of sampling
  - **Lemma**: $K_{MI,H_M}(X_i, X_j)$ is a Mercer Kernel

FAMER: Time Complexity

- Embedding + Computing of Kernel Matrix
- $O(C_1 \cdot dnN + C_2 \cdot N^2)$
  - N: number of bags;
  - n: average instances number within each bag;
  - d: dimensionality of feature vector.
  - C_1 and C_2 are constant depend on parameters
- Linear with n: efficient, compared with other typical MIL Kernels (Quadratic with n).

Outline

- Background
- Thinking in MIL Kernel
- Description of FAMER
- **Experiment**
- Conclusion

Experiments

- Effectiveness vs. Efficiency
- Three task:
  - Drug Activity Prediction
    - Musk1, Musk2
  - Automatic Image Annotation
    - Elephant, Fox, Tiger
  - Indentifying Trx-fold Proteins
    - Protein

Experiments

- Comparison
  - Typical MIL Kernels: NSK, STK, MG-ACC Kernel, PPMM Kernel, mi-Graph Kernel
  - Other famous MIL algorithms: mi-SVM, MI-SVM, EM-DD
- Use10 times10CV to obtain Classification Accuracy

Experiments: Effectiveness (1)

- Classification Accuracy on Musk data set

Algorithm	Musk1	Musk2	Average
NSK_{rbf} [8]	88.0%	89.3%	88.7%
STK [8]	91.6%	86.3%	89.0%
MIGraph Kernel [13]	90.0%	90.0%	90.0%
miGraph Kernel [13]	88.9%	90.3%	89.6%
MG-ACC Kernel [12]	90.1%	90.4%	90.3%
PPMM Kernel [21]	**95.6%**	81.2%	88.4%
FAMER	91.3%	**93.3%**	**92.3%**
MI-SVM [9]	77.9%	84.3%	81.1%
mi-SVM [9]	87.4%	83.6%	85.5%
MissSVM [33]	87.6%	80.0%	83.8%
DD [6]	88.0%	84.0%	86.0%
EM-DD [34]	84.8%	84.9%	84.9%
APR [1]	92.4%	89.2%	90.8%
MI-Box[30]	91.2%	90.3%	90.8%

Experiments: Effectiveness (2)

- Classification Accuracy on Automatic Image Annotation

Algorithm	Elephant	Fox	Tiger
NSK_{rbf}	84.3%	60.3%	84.2%
STK	83.5%	63.0%	79.0%
MIGraph Kernel	85.1%	61.2%	81.9%
miGraph Kernel	86.8%	61.6%	86.0%
PPMM Kernel	82.4%	60.3%	80.2%
FAMER	**87.5%**	**67.0%**	**87.0%**
MI-SVM	81.4%	59.4%	84.0%
mi-SVM	82.0%	58.2%	78.9%
EM-DD	78.3%	56.1%	72.1%

Experiments: Effectiveness (3)

- True Positive (TP) and True Negative(TN) typical Kernel under TrX Protein.

Algorithm	TP	TN	Average
NSK_{rbf}	69.0%	**72.6%**	**70.8%**
STK	58.1%	66.1%	62.1%
MIGraph Kernel	68.1%	70.3%	69.2%
miGraph Kernel	64.1%	69.7%	66.9%
FAMER	**70.1%**	70.9%	**70.5%**

Experiments: Efficiency

- Running Time and Detailed Description of Dataset

Algorithm	Musk1	Musk2	Elephant	Fox	Tiger	Protein
NSK_{rbf}	**375***ms*	67,562*ms*	4,438ms	4,015ms	3,453ms	161,844ms
MIGraph Kernel	1218*ms*	784,422*ms*	9,875ms	9,203ms	7,816ms	7,123,830ms
miGraph Kernel	385*ms*	77,485*ms*	5,250ms	4,703ms	4,016ms	199,461ms
FAMER	$609 + 169$ $= 778ms$	$9,431 + 461$ $= \mathbf{9,892}ms$	$1704 + 500$ $= \mathbf{2,204}ms$	$1625 + 501$ $= \mathbf{2,126}ms$	$1502 + 498$ $= \mathbf{2,000}ms$	$4,359 + 3,875$ $= \mathbf{8,234}ms$

Dataset	Musk1	Musk2	Elephant	Fox	Tiger	Protein
Number of bags	92	102	200	200	200	180
Number of positive bags	47	39	100	100	100	20
Number of negative bags	45	63	100	100	100	160
Average number of instances per bag	5.2	**64.7**	6.96	6.6	6.1	**137.8**
Dimensionality of instance space	166	166	230	230	230	8

Experiments: Efficiency

- Three observation：
  - FAMER： best efficiency on 5/6 dataset
  - Under small data set, FAMER is averagely better than typical Multi-Instance Kernels
  - As the number of instances increases in every bag, the superiority of FAMER becomes more and more significant

Outline

- Background
- Thinking in MIL Kernel
- Description of FAMER
- Experiment
- **Conclusion**

Conclusion

- FAMER: Fast kernel for MIL
  - Captures valuable correspondence and co-occurrence information.
  - Avoid pair-wise manner to speed up the computation.
  - Captures key instances in MIL by Weighting scheme

Thanks!