University of Innsbruck

Doctoral Thesis

Inference, Learning and Optimization on Structure Domains: Methods and Applications

Author: Hanchen Xiong

Supervisor: Prof. Justus Piater

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Intelligent and Interactive System Group
Institute of Computer Science

April 2015

Declaration of Authorship

I, Hanchen XIONG, declare that this thesis titled, 'Inference, Learning and Optimization on Structure Domains: Methods and Applications' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Act without doing, work without effort. Think of the small as large and the few as many. Confront the difficult while it is still easy; accomplish the great task by a series of small acts. The Master never reaches the great; thus achieves greatness. "

Lao Tuz

UNIVERSITY OF INNSBRUCK

Abstract

Faculty of Mathmatics, Computer Science and Physics
Institute of Computer Science

Doctor of Philosophy

Inference, Learning and Optimization on Structure Domains: Methods and Applications

by Hanchen XIONG

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

To my wife Youlin FANG,
thanks for being in my life !

Chapter 1

Introduction

"Before the begining of great brilliance, there must be chaos."

I Ching

This is an introductory chapter which provides an overview of the thesis. As the title suggests, this thesis focuses on working with structural data, including inference, learning and optimization. In section 1.1, some motivation and background of studying structure domains are explained. In section 1.2, an outline of the thesis is presented. Basically, the thesis is composed of three parts and each part is briefly introduced. In section 1.3, as the author's contribution to relevant scientific fields, his research papers during Phd study are listed and summarized.

1.1 Work on Structure Domains

This section mainly explains the concept of "structure" and some structure domains which will be studied later.

Generally, structures exist almost everywhere in the universe. For instance, it is usually said that a galaxy, an architecture or a society is a structure. Looking closer at these three examples and abstracting their similarities, a rough definition of the "structure" can somehow be derived:

A structure is is a set of elements, which exist with certain status based on the interactions among them.

The "interactions" can be also replaced with other terms at different scenarios, *e.g.* dependencies, compatibilities, constraints. Tow critical points in understanding the definition are: (i). there should be more than one element; (ii). each element's status can be affected by the statuses of other ones. Based on this specification, more "structure" examples can also be easily observed. A sentence is a structure by considering words as its elements, an image is a structure as a set of dependent pixels, a human body is a structure composed of arms, legs and other parts, just name a few. Indeed, structure domains is very common in many areas.

In old times, people are already aware of integrating data collected from different sources to find internal patterns among them. However, this procedure was usually slow and unstable. With the development of modern computers and digitization, increasingly more structural data are available.

In these thesis, structures are dealt with in three forms: (i). graphs, (ii). kernels, (iii). manifolds.

1.2 Overview

The thesis can be divided into three parts: inference, learning and optimization. The first part contains Chapter 2, where (probabilistic) graphical models and corresponding inference are presented. Chapter 3 and Chapter 4 belong to the second part, where two principles for handling structures, *kernels* and *graphs*, are studied respectively. The third part contains Chapter 5, which studies the optimization on *graphs* and *matrix manifold*.

1.2.1 Part I: Inference with Graphical Models

1.2.2 Part II: Structural Output Learning

1.2.3 Part III: Optimization on Structures

1.3 Author's Contribution

During the author's PhD study, ten relevant research papers were finished, out of which nine are published (or accepted for publication) in journals or conference proceedings.

They are listed as follows:

I. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Implicit Learning of Simpler Output Kernels for Multi-Lable Prediction*, NIPS workshop on Representation and Learning Methods for Complex Outputs (NIPS-RLCO2014).

II. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Towards Maximum Likelihood: Learning Undirected Graphical Models using Persistent Sequential Monte Carlo*, The 6th Asian Conference on Machine Learning (ACML2014), **Best Paper Award**.

III. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels*, Neurocomputing Journal (Accepted).

IV. **Hanchen Xiong**, Sandor Szedmak, Antonio Rodríguez Sánchez, Justus Piater. *Towards Sparsity and Selectivity: Bayesian Learning of Restricted Boltzmann Machine for Early Visual Features*,In Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN14), 2013, Springer.

V. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Joint SVM for Accurate and Fast Image Tagging*,In Proceedings of the 22nd European Symposium on Artificial Neural Network (ESANN14).

VI. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *3D Object Class Geometry Modeling with Spatial Latent Dirichlet Markov Random Fields*, In Proceedings of the 35th German Conference on Pattern Recognition (GCPR13), pp 51-60, 2013, Springer.

VII. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Homogeneity Analysis for Object-Action Relations Reasoning in Kitchen Scenarios*, In Proceedings of 2nd Workshop on Machine Learning for Intelligent Systems (MLIS13), pp 37-44, 2013, ACM.

VIII. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *A Study of Point Cloud Registration with Probability Product Kernel Functions*, In Proceedings of 2013 International Conference on 3D Vision (3DV13), pp 207-214, 2013, IEEE.

IX. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Efficient, General Point Cloud Registration With Kernel Feature Maps*, In Proceedings of 10th International Conference on Computer and Robot Vision (CRV13), pp 83-90, 2013, IEEE.

To minimize content redundancy, only a subset of them are inserted at appropriate (sub)chapters of the thesis. In addition, to keep them self-contained and self-consistent,

their original content and formats are preserved. Which should be distinguished from regular paragraphs. The preprints of these publications can be also found at: http://iis.uibk.ac.at/publications, complying with their corresponding copyrights.

Besides, one unpublished work is also be presented in the thesis.

X. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Multi-Label Learning with Kernel Generalized Homogeneity Analysis*, Unpublished, 2015.

Throughout the thesis, all these papers will be referred to by their corresponding roman numerals specified above.

Chapter 2

Graphical Models: Structural Modeling and Inference

"Study the past if you would define the future."

Confucius

2.1 Probabilistic Graphical Models

This chapter

Graphical models have been studied in many disciplines, such as artificial intelligence (**?**), statistics (**?**). It is widely considered as one of the main progresses in machine learning research by bringing graphs in to assist probabilistic representation, analysis and computing.

Basically, graphical models are divided into two categories: directed and undirected.

2.1.1 Bayesian Networks

Definition 2.1. A Bayesian network is a Directed Acyclic Graph (DAG), which corresponds to a distribution of the form:

$$P(X) = \prod_i P(x_i|\text{pa}(x_i)) \tag{2.1}$$

where $X = \cup\{x_i\}$, $\text{pa}(x_i)$ denotes the parent nodes of x_i.

Theorem 2.2. ***D-Separation Rule**: Given three non-intersecting subsets of nodes A, B, S, in a graph G, we consider all possible paths from any node in A to any node in B. The path is said blocked if either:*

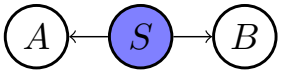
1. *the path goes through either* head-to-tail *or* tail-to-tail *at the node, and the node is in set S, or*
2. *the path goes through* head-to-head *at the node, and neither the node nor any of its descendants is in the set S.*

If all paths are blocked, then S D-separates A from B, and then:

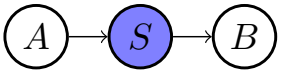
$$A \perp\!\!\!\perp B|S$$

The D-Separation rule can be simply proofed as follows:

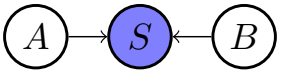
Proof. Three type of separating nodes:*tail-to-tail*,*head-to-tail* and *head-to-head* are considered in three simple graphs respectively



$$\begin{aligned} & P(A,B|S) \\ = \; & \tfrac{P(A,B,S)}{P(S)} \\ = \; & \tfrac{P(A|S)P(B|S)P(S)}{P(S)} \\ = \; & P(A|S)P(B|S) \end{aligned}$$



$$\begin{aligned} & P(A,B|S) \\ = \; & \tfrac{P(A,B,S)}{P(S)} \\ = \; & \tfrac{P(B|S)P(S|A)P(A)}{P(S)} \\ = \; & P(B|S)\tfrac{P(S|A)P(A)}{P(S)} \\ = \; & P(B|S)P(A|S) \end{aligned}$$



$$\begin{aligned} & P(A,B|S) \\ = \; & \tfrac{P(A,B,S)}{P(S)} \\ = \; & \tfrac{P(S|A,B)P(A)P(B)}{P(S)} \end{aligned}$$

□

Example: One morning Tracey found that the grass in her garden is wet ($T \in \{0,1\}$). Is it due to overnight rain ($R \in \{0,1\}$)or did she forget to turn off the sprinkler last

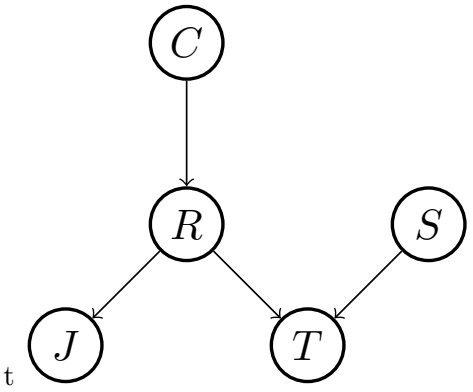


FIGURE 2.1: The Bayesian network corresponding to the wet-glass example.

night ($S \in \{0,1\}$)? Next she notices that the grass of her neighbor, Jack, is also wet ($J \in \{0,1\}$), and she also remembered it was cloudy on the previous daytime ($C \in \{0,1\}$).

$$P(J,T,R,C,S) = P(J|R)P(T|R,S)P(R|C)P(C) \tag{2.2}$$

2.1.2 Markov Networks

A more general graph for modeling dependencies among variables is Markov network, or sometimes referred to as Markov random fields (MRFs).

we can rewrite another factorization form of the joint distribution :

$$\begin{aligned} & P(J,T,R,C,S) \\ = \; & P(J|R)P(T|R,S)P(R|C)P(C)P(S) \\ = \; & \underbrace{P(J|R)}_{\phi_1(J,R)}\underbrace{P(R|C)P(C)}_{\phi_2(R,C)}\underbrace{P(T|R,S)P(S)}_{\phi_3(T,R,S)} \\ = \; & \phi_1(J,R)\phi_2(R,C)\phi_3(T,R,S) \end{aligned}$$

where $\phi_1(J,R), \phi_2(R,C), \phi_3(T,R,S)$ are no longer conditional probabilities

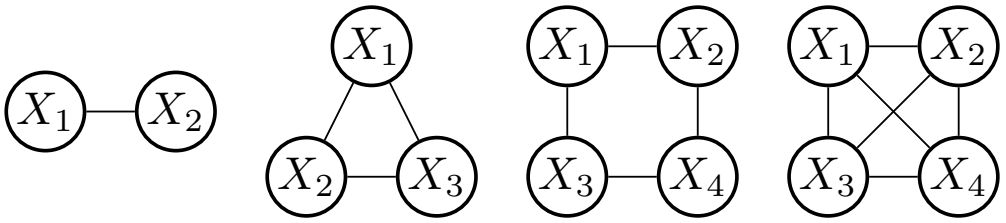
Definition 2.3. A Markov Network is an undirected graph which corresponds to the distribution of the form:

$$P(x_1, x_2, ...x_n) = \frac{1}{Z}\prod_c \phi_c(X_c) \tag{2.3}$$

where X_c is a clique of the graph nodes, and $\phi_c(X_c)$ is called potential function over clique X_c, and Z is to ensure the distribution normalized.

Remarks:

- the potential function is non-negative, and can be a function of any form over the clique X_c, and in most cases it reflects the dependences /compatibilities /constrains among $x_i \in X_c$;
- the normalization term can be computed as $Z = \sum_{x_1,x_2,\ldots x_n} \prod_c \phi_c(X_c)$ or
- the clique defined here means maximal *fully connected subgraph*



$\psi(X_1, X_2)$ $\psi(X_1, X_2, X_3)$ $\psi(X_1, X_2)$ $\psi(X_2, X_4)$ $\psi(X_1, X_3)$ $\psi(X_3, X_4)$ $\psi(X_1, X_2, X_3, X_4)$

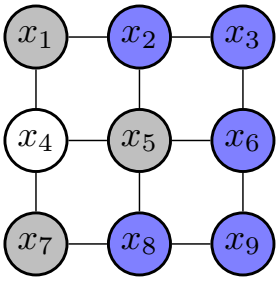
Theorem 2.4. ***Markov Separation Rule*** *: a subset $\mathcal{A}$ is said to be separated from another subset $\mathcal{B}$ by subset $\mathcal{S}$ if all possible paths from any member of $\mathcal{A}$ to any member $\mathcal{B}$ pass through $\mathcal{S}$. Conditional Independence: If $\mathcal{S}$ separates $\mathcal{A}$ from $\mathcal{B}$, then*

$$\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\mathcal{S}$$

Corollary 2.5. ***Local Conditional Independence****: when conditioned on its neighbor, x_i is independent of all other variables of the graph:*

$$P(x_i|X \backslash x_i) = P(x_i|ne(x_i))$$

For instance,

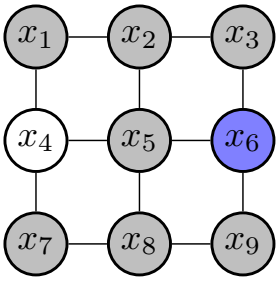


$x_4 \perp\!\!\!\perp \{x_2, x_3, x_6, x_8, x_9\}|\{x_1, x_5, x_7\}$

Corollary 2.6. ***Pairwise Conditional Independence****: given two non-adjacent variables x_i and x_j variables, they are independent conditioned all other variables of the graph:*

$$x_i \perp\!\!\!\perp x_j | X \backslash \{x_i, x_j\}$$

For instance,



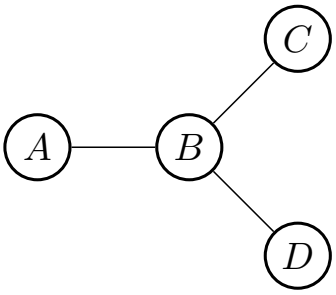
$$x_4 \perp\!\!\!\perp x_6 | \{x_1, x_2, x_3, x_5, x_7, x_8, x_9\}$$

2.1.3 Connecting Bayesian Networks and Markov Networks

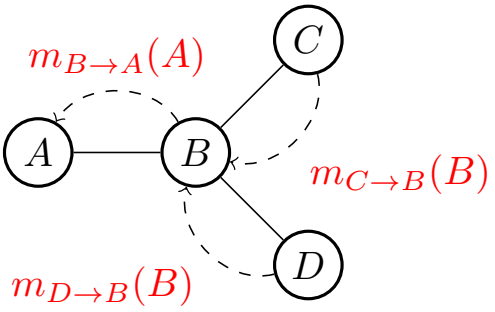
2.2 Exact and Approximate Inference

2.2.1 Belief Propagation

How to compute the probability $P(A)$

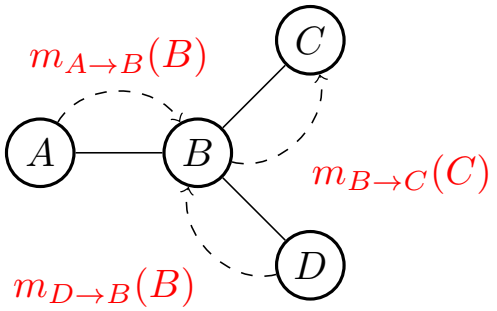


$$\begin{aligned}
& P(A) \\
= & \overbrace{\sum_{B,C,D} P(A,B,C,D)}^{complexity:\mathcal{O}(n^3)} \\
\propto & \sum_{B,C,D} [\phi(A,B)\phi(B,C)\phi(B,D)] \\
\propto & \sum_B \phi(A,B) \underbrace{\sum_C \phi(B,C)}_{m_{C\to B}(B)} \underbrace{\sum_D \phi(B,D)}_{m_{D\to B}(B)} \\
\propto & \sum_B \{\phi(A,B) m_{C\to B}(B) m_{D\to B}(B)\} \\
\propto & m_{B\to A}(A)
\end{aligned}$$



$$\begin{aligned}
& P(A) \\
\propto & \sum_B \phi(A,B) \sum_C \phi(B,C) \sum_D \phi(B,D) \\
\propto & \sum_B \{\phi(A,B) m_{C\to B}(B) m_{D\to B}(B)\} \\
\propto & m_{B\to A}(A)
\end{aligned}$$

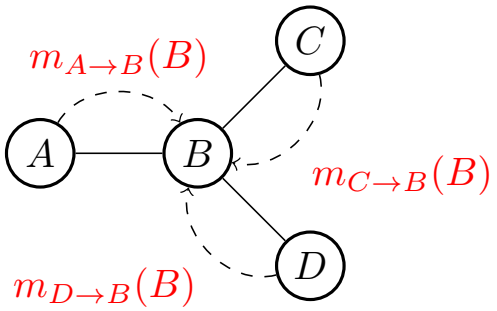
- $m_{C\to B}(B)$ is called belief function of B over C by summing out C from the potential function $\sum_C \phi(B,C)$, and it will propagate afterwards;
- intuitively, belief propagation can be understood as a 'message passing scheme', $m_{C\to B}(B)$ is a message sent from C to B by telling B all possible states of C; and then all messages are processed and passed forward$A$;



$$
\begin{aligned}
& P(C) \\
\propto\ & \textstyle\sum_A \phi(A,B) \sum_B \phi(B,C) \sum_D \phi(B,D) \\
\propto\ & \textstyle\sum_B \phi(B,C) \sum_A \phi(A,B) \sum_D \phi(B,D) \\
\propto\ & \textstyle\sum_B \{\phi(B,C) m_{A\to B}(B) m_{D\to B}(B)\} \\
\propto\ & m_{B\to C}(C)
\end{aligned}
$$

one variable can send a message to one of its neighbor only if it has received messages from all other variables

$$m_{B\to C}(C) = \sum_B \{\phi(B,C) m_{A\to B}(B) m_{D\to B}(B)\}$$



$$
\begin{aligned}
& P(B) \\
\propto\ & \textstyle\sum_A \phi(A,B) \sum_C \phi(B,C) \sum_D \phi(B,D) \\
\propto\ & m_{A\to B}(B) m_{C\to B}(B) m_{D\to B}(B)
\end{aligned}
$$

the marginal probabilities are computed as the product of incoming messages:

$$
\begin{aligned}
P(B) &= \tfrac{1}{Z} m_{A\to B}(B) m_{C\to B}(B) m_{D\to B}(B) \\
P(A) &= \tfrac{1}{Z} m_{B\to A}(A) \\
P(C) &= \tfrac{1}{Z} m_{B\to C}(C) \\
P(D) &= \tfrac{1}{Z} m_{B\to D}(D)
\end{aligned}
$$

Algorithm 1 Belief Propagation for tree-structured Markov networks

1: select one variable as the root;
2: starting from all leaves, propagate *beliefs* toward the root as:

$$m_{i\to j}(x_j) = \sum_{x_i} \left\{ \phi(x_i, x_j) \prod_{k\in \mathrm{Ne}(i)\setminus j} m_{k\to i}(x_i) \right\}$$

3: when the root receives all messages from its neighbors, then propagate backwards the *"inverse beliefs"* as the step 2;
4: the marginal probability of each variable is computed as:

$$P(x_i) \propto \prod_{k\in \mathrm{Ne}(i)} m_{k\to i}(x_i)$$

where $\mathrm{Ne}(i)$ denotes the neighboring variables of x_i in the graph.

Algorithm 2 Generalized Belief Propagation for Loopy Markov networks

1: initialize all messages $m_{i\to j}$ in both directions of all connected variables randomly or with a constant value (*e.g.* 1);
2: **while** all messages converge **do**
3: update messages as:

$$m_{i\to j}^{(t+1)}(x_j) = \sum_{x_i} \left\{ \phi(x_i, x_j) \prod_{k\in \mathrm{Ne}(i)\setminus j} m_{k\to i}^{(t)}(x_i) \right\}$$

4: **end while**

2.2.2 Variational Method

2.2.3 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a family of sampling algorithms, which are designed to generate samples from a target distribution $p(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$. Usually, directly sampling from $p(\mathbf{x})$ is not possible since its density can be arbitrarily complex, however, $p(\mathbf{x})$ can be evaluated up to a normalizing constant:

$$p(\mathbf{x}) = \frac{\tilde{p}(\mathbf{x})}{\mathbf{Z}}$$

2.2.3.1 Metropolis Algorithm

- the proposal distribution is symmetric: $q(\mathbf{x}^*|\mathbf{x}^t) = q(\mathbf{x}^t|\mathbf{x}^*)$;
- acceptance probability is: $\mathcal{A}(\mathbf{x}^*; \mathbf{x}^t) = \min\{1, \frac{p(\mathbf{x}^*)}{p(\mathbf{x}^t)}\}$;

Algorithm 3 Metropolis Algorithm

initialize $\mathbf{x}^0$.
for $t = 0$ to $T-1$ **do**
  generate a sample uniformly from $[0,1]$: $u \sim \mathcal{U}_{[0,1]}$.
  generate a sample from a symmetric proposal distribution: $\mathbf{x}^* \sim q(\mathbf{x}^*|\mathbf{x}^t)$.
  if $u < \min\{1, \frac{p(\mathbf{x}^*)}{p(\mathbf{x}^t)}\}$ **then**
    $\mathbf{x}^{t+1} = \mathbf{x}^*$.
  else
    $\mathbf{x}^{t+1} = \mathbf{x}^t$.
  end if
end for

It can be seen in the Metropolis algorithm that $\mathbf{x}^0, \mathbf{x}^1, \cdot, \mathbf{x}^t$ are not independent because successive samples are correlated. Therefore, in practice, real independent samples can be obtained by only retraining samples at every M iterations.

The detailed balance of the Metropolis algorithm can be proof as follows:

Proof.

$$\begin{aligned} p(\mathbf{x}^t)p(\mathbf{x}^*|\mathbf{x}^t) &= p(\mathbf{x}^t)\left(q(\mathbf{x}^*|\mathbf{x}^t)\min\{1, \tfrac{p(\mathbf{x}^*)}{p(\mathbf{x}^t)}\}\right) \\ &= \min\{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t), q(\mathbf{x}^*|\mathbf{x}^t)p(\mathbf{x}^*)\} \\ &= p(\mathbf{x}^*)q(\mathbf{x}^*|\mathbf{x}^t)\min\{1, \tfrac{p(\mathbf{x}^t)}{p(\mathbf{x}^*)}\} \\ &\overset{\text{since } q(\mathbf{x}^*|\mathbf{x}^t)=q(\mathbf{x}^t|\mathbf{x}^*)}{=} p(\mathbf{x}^*)\left(q(\mathbf{x}^t|\mathbf{x}^*)\min\{1, \tfrac{p(\mathbf{x}^t)}{p(\mathbf{x}^*)}\}\right) \\ &= p(\mathbf{x}^*)p(\mathbf{x}^t|\mathbf{x}^*) \end{aligned} \tag{2.4}$$

□

2.2.3.2 Metropolis-Hasting Algorithm

- the proposal distribution can be arbitrary (no symmetric restriction): $q(\mathbf{x}^*|\mathbf{x}^t)$;
- acceptance probability is: $\mathcal{A}(\mathbf{x}^*; \mathbf{x}^t) = \min\{1, \frac{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}\}$;

The detailed balance of the Metropolis algorithm can be proof as follows:

Proof.

$$\begin{aligned} p(\mathbf{x}^t)p(\mathbf{x}^*|\mathbf{x}^t) &= p(\mathbf{x}^t)\left(q(\mathbf{x}^*|\mathbf{x}^t)\min\{1, \tfrac{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}\}\right) \\ &= \min\{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t), q(\mathbf{x}^*|\mathbf{x}^t)p(\mathbf{x}^*)\} \\ &= p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)\min\{1, \tfrac{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}\} \\ &= p(\mathbf{x}^*)\left(q(\mathbf{x}^t|\mathbf{x}^*)\min\{1, \tfrac{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}\}\right) \\ &= p(\mathbf{x}^*)p(\mathbf{x}^t|\mathbf{x}^*) \end{aligned} \tag{2.5}$$

Algorithm 4 Metropolis-Hasting (MH) Algorithm

initialize $\mathbf{x}^0$.
for $t = 0$ to $T - 1$ **do**
  generate a sample uniformly from $[0, 1]$: $u \sim \mathcal{U}_{[0,1]}$.
  generate a sample $\mathbf{x}^* \sim q(\mathbf{x}^*|\mathbf{x}^t)$.
  if $u < \min\{1, \frac{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}$ **then**
    $\mathbf{x}^{t+1} = \mathbf{x}^*$.
  else
    $\mathbf{x}^{t+1} = \mathbf{x}^t$.
  end if
end for

Algorithm 5 Gibbs Sampling Algorithm

initialize $\mathbf{x}^0$.
for $t = 0$ to $T - 1$ **do**
  randomly select
  sample x_k^* from $p(x_k^*|x_{[1,\cdots,d]/k}^t)$.
  $x_k^{t+1} = x_k^*$.
end for

□

2.2.3.3 Gibbs Sampling Algorithm

The Gibbs sampling is a special case of MH with the proposal distribution defined as the cyclic conditional distributions among $\{x_1, x_2, \cdots, x_d\}$.

- the proposal distribution: $q(\mathbf{x}^*|\mathbf{x}^t) = p(x_k^*|x_{[1,\cdots,d]/k}^t)$;
- acceptance probability is:

$$\begin{aligned}\mathcal{A}(\mathbf{x}^*; \mathbf{x}^t) &= \min\{1, \frac{p(\mathbf{x}^*)q(\mathbf{x}^t|\mathbf{x}^*)}{p(\mathbf{x}^t)q(\mathbf{x}^*|\mathbf{x}^t)}\} \\ &= \min\{1, \frac{p(\mathbf{x}^*)p(x_{[1,\cdots,d]/k}^t|x_k^*)}{p(\mathbf{x}^t)p(x_k^*|x_{[1,\cdots,d]/k}^t)}\} \\ &= \min\{1, \frac{p(x_k^*)p()p(x_{[1,\cdots,d]/k}^t|x_k^*)}{p(\mathbf{x}^t)p(x_k^*|x_{[1,\cdots,d]/k}^t)}\}\}\end{aligned}$$

2.3 3D Part-Based Shape Modeling with Spatial Latent Dirichlet Markov Random Fields

VI. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *3D Object Class Geometry Modeling with Spatial Latent Dirichlet Markov Random Fields*, In Proceedings of the

35th German Conference on Pattern Recognition (GCPR13), pp 51-60, 2013, Springer.

3D Object Class Geometry Modeling with Spatial Latent Dirichlet Markov Random Fields*

Hanchen Xiong Sandor Szedmak Justus Piater

Institute of Computer Science, University of Innsbruck
{*hanchen.xiong, sandor.szedmak, justus.piater*}*@uibk.ac.at*

Abstract. This paper presents a novel part-based geometry model for 3D object classes based on latent Dirichlet allocation (LDA). With all object instances of the same category aligned to a canonical pose, the bounding box is discretized to form a 3D space dictionary for LDA. To enhance the spatial coherence of each part during model learning, we extend LDA by strategically constructing a Markov random field (MRF) on the part labels, and adding an extra spatial parameter for each part. We refer to the improved model as spatial latent Dirichlet Markov random fields (SLDMRF). The experimental results demonstrate that SLDMRF exhibits superior semantic interpretation and discriminative ability in model classification to LDA and other related models.

1 Introduction

During the past decades, computer vision has made remarkable progress in visual object understanding, e.g. classification, pose estimation and segmentation, etc. However, most previous study of object modeling is based on 2D images, in which appearance is the main and only information source for various tasks, so most attention is focused on increasing the robustness of algorithms to lighting changes, intra-class appearance variation and viewpoint variation [4]. Meanwhile, 3D geometry properties of objects have been rarely exploited and used to increase the expressiveness of object models. Recently, pioneering work [7,13] has attempted to add 3D geometric information to object models, demonstrating that the accuracy and robustness of such algorithms can be enhanced with extra 3D geometry clues. However, there still exists an obvious gap between 2D appearance modeling and 3D geometry modeling with respect to their interpretation and representation abilities, and it has been advocated [7,13] that robust 3D geometry modeling is highly desirable. Motived by this gap and desire, this paper puts forward a novel 3D object class geometry model in the light of state-of-the-art techniques developed in machine learning and computer graphics. Part-based models have displayed merits in 2D appearance modeling [4] for handling partial occlusion, our 3D geometry model is likewise part-based and inherits these strengths. The training data of our algorithm are collections of 3D

* The research has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreement no. 270273, Xperience.

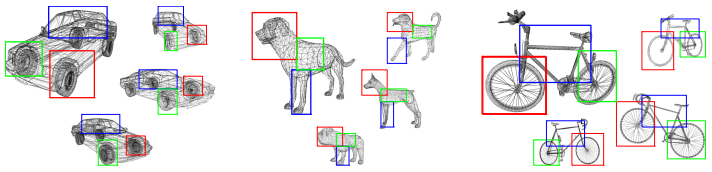


Fig. 1. Different object instances of the same class should share similar 3D structure of composing parts, although their parts can slightly vary from one instance to another.

models of different instances which belong to the same category (Figure 1). The basic underlying principle of our modeling is the concept that different object instances of the same class should share similar 3D structure of composing parts, although their parts can slightly vary from one instance to another. For example, all bicycles are composed of a frame and two wheels, and the geometric relation between these three parts are similar across different instances (Figure 1). In this paper, 3D objects are represented by point cloud data (PCD), which is a general and popular representation of 3D shapes and can easily be converted from other data formats (e.g. meshes). First, for each class, different PCDs of object instances are aligned using point cloud registration methods. Secondly, the main learning step is inspired by latent Dirichlet allocation (LDA) [1] and computer graphics [5]. LDA is a state-of-the-art machine learning tool for discovering latent topics within document collections. Here we apply LDA by considering each object point cloud as a document, and each part as a topic. With the bounding box volume discretized into a 3D grid dictionary, the part can be mined out as a multinomial distribution over the discrete 3D space, and each object is a multinomial distribution over parts. However, standard LDA ignores the spatial coherence, which is of great importance in our task but not generally taken into account in natural language applications. Based on discoveries in computer graphics [5] and other work on LDA [8,11], we develop a spatial latent Dirichlet Markov random field (SLDMRF) model with extra undirected links between topic labels and spatial parameters. The proposed SLDMRF can co-segment all point clouds simultaneously under a prior of coherence of correspondence, spatial continuity and spatial smoothness. According to our empirical results (section 3), compared to standard LDA and other related models, SLDMRF can achieve much more consistent and semantically meaningful segmentations of 3D point clouds, and the learned class geometry models display better discriminative ability in classification.

1.1 Related work

The starting point of 3D geometry modeling in visual object understanding is the difficulty in dealing with appearance variation due to different viewpoints. There have been several attempts to embed 3D geometric information into object models [2,3,7,13], and all of them have reported improvement in accuracy and robustness, although different 3D geometry information are exploited and modelled in their work. In [2] 3D object shapes are probabilistically modelled as continuous

distributions in $\mathbb{R}^3$ with a kernel density estimator (KDE). However, that work explicitly addresses neither category-level tasks nor semantic segmentation. Objects are considered as Gausssaian mixtures and expectation-maximization (EM) is applied to estimate corresponding Gaussian parameters and weights. One observation of Gaussian-mixture-based segmentation is that discovered parts rarely display good semantic interpretability since usually the part geometry is too complex to be modelled as a Gaussian (section 3.1). Other work attempts to improve the expressiveness of object geometry models in different ways. For example, Detry et al. [3] represent objects as hierarchically-organized spatial distributions of distinct feature types, but did not seek to produce semantically-meaningful segmentations. Other models [7,13] extract 3D geometry information at the class level. However, in [7] the segmentation is again based on Gaussian mixtures and EM, and most [13] do not model object classes in a part-based manner to avoid segmentation. Meanwhile, another thread of segmentation-based visual modeling is the application of Latent Dirichlet allocation (LDA) in computer vision [10,11,8]. LDA was originally developed to discover hidden topics in text corpora by clustering words into different topics [1]. Standard LDA, however, ignores spatial coherence, which is problematic in vision applications. Therefore, spatial LDA (SLDA) [11] and Latent Dirichlet Markov random fields (LDMRF) [8] were put forward to produce better, spatially-coherent segmentations. In addition, with higher emphasis of the smoothness of parts and consistent correspondences, 3D segmentation in computer graphics [5] constructs graphs with neighboring intra-links and correspondence inter-links among objects, and min-cut is used on graphs for segmentation.

The main contribution of this paper is an extension of LDA for 3D object class geometry modeling, which we refer to as Spatial Latent Dirichlet Markov Random Fields(SLDMRF). The proposed model is built with inspiration from recent advances in different fields [1,11,8,5], and it yields superior interpretability and representational capability in modeling 3D object class geometry.

2 3D Object Class Geometry Modeling

With the point cloud representations of 3D object shapes, the alignment of different instances of the same class is achieved with point cloud registration algorithms. While any suitable registration procedure can be used, we adopted a novel method [12] since it is very efficient and robust to non-rigid transformation, which suits the case of intra-category shape variation. An example of aligning dogs is displayed in Figure 2.

2.1 Latent Dirichlet Allocation

LDA [1] is a generative model that utilizes the information of co-occurring words to find out hidden topics shared by documents. In LDA, each document is considered as a finite mixture of topics; each topic is a finite mixture of words. The graphical model of LDA is shown in Figure 3(a). The generative process of LDA

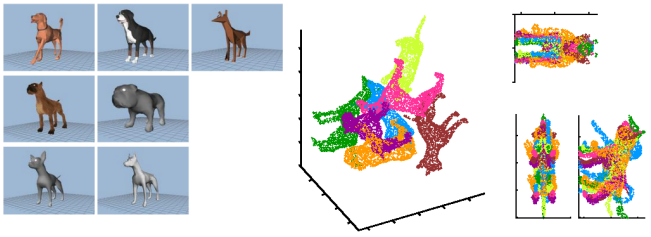


Fig. 2. Alignment of different dog instances by point cloud registration [12]. Left: original 3D shapes of different dog instances; middle: point clouds generated from the shapes on the left; right: three views (top, profile, front) of the point clouds (middle) after alignment.

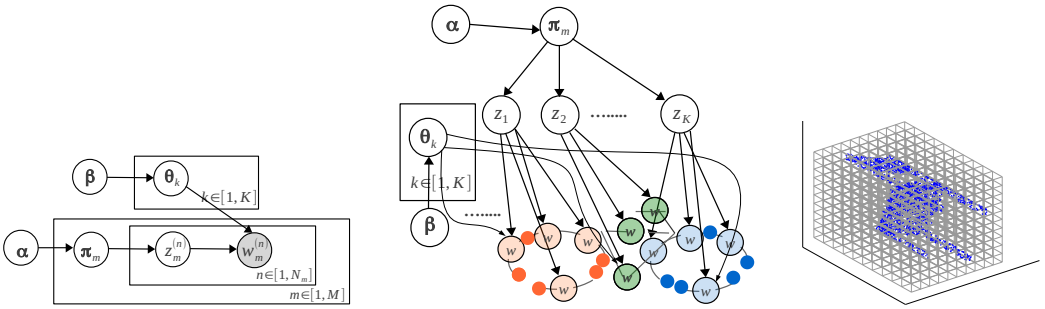


Fig. 3. (a) Graphical model of LDA; (b) Application of LDA to model 3D object categories; (c) Construction of 3D dictionary by discretizing the 3D space of the bounding box.

is as follows: (1) for each topic $k \in [1, K]$, a multinomial parameter $\boldsymbol{\theta}_k$ over words is sampled from Dirichlet prior $\boldsymbol{\beta}$; (2) for each document $m \in [1, M]$, a multinomial parameter $\boldsymbol{\pi}_m$ over K topics is sampled from Dirichlet prior $\boldsymbol{\alpha}$; (3) for each word $w_m^{(n)}$, $n \in [1, N_m]$ in document m, a topic label $z_m^{(n)}$ is first sampled from multinomial distribution $z_m^{(n)} \sim \text{Multinomial}(\boldsymbol{\pi}_m)$, then the word $w_m^{(n)}$ is sampled from the multinomial distribution parametrized with $\boldsymbol{\theta}_{z_m^{(n)}}$, $w_m^{(n)} \sim \text{Multinomial}(\boldsymbol{\theta}_{z_m^{(n)}})$. Hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ define the Dirichlet priors governing the parameters of multinomial distributions. Usually $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are set in a symmetric manner and using low values [6]. In [10], LDA is applied on a collection of images. Each image is considered as a document, objects correspond to topics, and visual words are generated using vector quantization on extracted features. In our case, however, LDA is utilized for 3D object class geometry modeling with objects of the same category as documents, and parts shared by different instances correspond to topics (Figure 3(b)).

3D Dictionary. In our task, LDA is expected to work effectively under the assumption that different objects of the same category should share very similar structure. Therefore, when LDA is applied on each collection of categorical object

point clouds, the co-occurring patterns are the 3D space occupied by 3D points. In each collection, all instances can be aligned to a canonical pose, based on which the 3D space of the bounding box is discretized into a grid, where each block represents a 3D word. Therefore, when transferring point clouds to corresponding documents, word w_k will replace 3D point x_i if x_i lies within block w_k. In this way, the discovered part actually is a distribution over 3D space, and a category is a mixture of these distributions. The concept of dictionary discretization is illustrated in Figure 3(c).

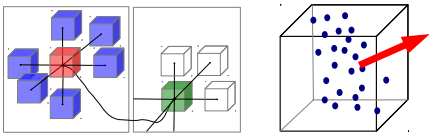
For label inference and parameters learning in LDA, a collapsed Gibbs sampling [6] can be formulated as

$$q_{\mathrm{LDA}}(z_m^{(n)} = k) \propto \frac{\mathbf{N}^{(k)}_{-mn,w_m^{(n)}} + \beta_{w_m^{(n)}}}{\sum_w^W (\mathbf{N}^{(k)}_{-mn,w} + \beta_w)} \cdot (\mathbf{N}^{(m)}_{-mn,k} + \alpha_k) \tag{1}$$

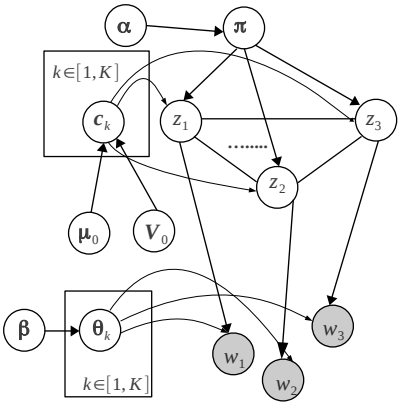
where $\mathbf{N}^{(k)}_{-mn,w}$ is the number of words in the corpus with value w assigned to topic k excluding the nth word in document m, and $\mathbf{N}^{(m)}_{-mn,k}$ is the number of words in document m assigned to topic k excluding the nth word in document m. From (1), it can be seen that LDA prefers to cluster together those words that often co-occur in the same document. Therefore, simply applying LDA on the 3D dictionary, unfortunately, is not expected to work because it misses a lot of spatial and correspondence information, which is not meaningful in the text processing case: (1) *Spatial coherence* is an important issue when LDA is applied in vision applications [11,8]. For example, in all point clouds of dogs, 3D words located in the hip and in the head will always co-occur. So by using (1), the hip and head of dogs can be clustered into a part, which is a spatially (of course also semantically) unreasonable segmentation. (2) *Correspondence coherence* is likewise important. LDA can find synonyms by finding their co-occurring patterns in documents. However, the "synonyms" in the 3D dictionary are identified by spatial correspondence. For example, in Figure 2, the legs of different dogs can rarely match exactly due to different species or standing poses. However, since all legs are close and correspond to each other, they should be clustered into the same part.

2.2 Spatial latent Dirichlet Markov random field

To enhance the spatial coherence, in Spatial LDA (SLDA) [11] 2D images are decomposed into small overlapping regions, which are used as documents to ensure that the pixels belonging to one part should be close to each other. Latent Dirichlet Markov random fields (LDMRF) [8], on the other hand, construct Markov random fields on the part label variables to enhance the local spatial coherence. However, both of them ignore the correspondences across the segmentations of different instances. Inspired by these improved versions of LDA and consistent co-segmentation in computer graphics [5], we put forward a novel spatial latent Dirichlet Markov random field (SLDMRF) that inherits virtues from both SLDA and LDMRF. However, rather than being a simple combination of SLDA and LDMRF, the proposed SLDMRF goes beyond them in several ways.



(a) Left: for each word (red block), there are two types of connections in SLDMRF: neighboring spatial connections (blue blocks) and correspondence connections (green block); right: the normal vector of each word in the document is estimated by using the points lying within the word.



(b) Graphical model of SLDMRF: compared to the standard LDA (Figure 3(a)), there are extra directed links between topic labels and spatial Gaussian parameters $\mathbf{c}_k$.

Fig. 4. SLDMRF modeling

First, instead of going through all small overlapping sub-volumes as SLDA does, we explicitly model the positions of all parts by parameters $\mathbf{c}_k$ such that 3D words that share the same label k are likely to be close to $\mathbf{c}_k$. Second, similarly to LDMRF, SLDMRF constructs a Markov random field on the neighboring label variables. However, different from LDMRF, we assign different potential functions based on the prior that the segmentation boundaries should be located at the point where abrupt curvature changes take place. The potential function is defined as

$$g(z_i, z_j) = \delta(z_i, z_j) \exp(|\langle o_i, o_j \rangle|) \tag{2}$$

where $\delta(z_i, z_j)$ equals 1 when z_i and z_j are neighbors, and 0 otherwise (Figure 4(a)), and o_i, o_j are the normal vectors estimated by using the points lying within word i and j respectively (Figure 4(a)). Last but not least, SLDMRF enhance the correspondences of segmentation across different instances. Inspired by the co-segmentation used in [5], we construct inter-connections between corresponding parts across different objects, and correspondences are matched by finding the nearest neighbors in other objects after alignment. In this way the segmentation can be more consistent within a category. The potential function $g(z_m^{(i)}, z_n^{(j)})$ for correspondence connections is set in the same way as spatial connections (2); $\delta(z_m^{(i)}, z_n^{(j)})$ is 1 if $z_m^{(i)}$ and $z_n^{(j)}$ are nearest neighbours of each other, and 0 otherwise. Because the part weights are already taken into account by LDA (parameter $\boldsymbol{\pi}$), the labels within the Markov random fields are modeled as:

$$p(\mathbf{Z}) \propto \exp\left(\textstyle\sum_{(i,j)} g(z_i, z_j)\right) \tag{3}$$

The graphical model of SLDMRF is presented in Figure 4(b). Hyperparameters $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$ (similar to $\boldsymbol{\alpha}, \boldsymbol{\beta}$) specify the Gaussian prior of part po-

sitions $\mathbf{c}_k \sim \mathcal{N}(\cdot; \boldsymbol{\mu}_0, \mathbf{V}_0)$. Given the part position $\mathbf{c}_k$, the label is sampled as $z_i \sim \mathcal{N}(\hat{w}_i; \mathbf{c}_k, \boldsymbol{\Lambda})$, where $\hat{w}_i$ denotes the 3D coordinates of word w_i. Since we do not expect the label distribution to be truly Gaussian, $\boldsymbol{\Lambda}$ is set relatively large. The joint probability of 3D words in SLDMRF $p(\{w_m^{(n)}\}_{m=1,n=1}^{M,N_m}, |\boldsymbol{\alpha}, \boldsymbol{\beta})$ is:

$$\frac{1}{\mathbf{Q}} \prod_{m=1}^{M} \prod_{n=1}^{N_m} \Bigg(\int_{\boldsymbol{\pi}_m} \int_{\boldsymbol{\theta}_{z_m^{(n)}}} \int_{\mathbf{c}_{z_m^{(n)}}} p(\boldsymbol{\pi}_m|\boldsymbol{\alpha}) \sum_{z_m^{(n)}=1}^{K} \Big(p(z_m^{(n)}|\boldsymbol{\pi}_m) p(w_m^{(n)}|\boldsymbol{\theta}_{z_m^{(n)}}) \\ \mathcal{N}(w_m^{(n)}; \mathbf{c}_{z_m^{(n)}}, \boldsymbol{\Lambda}) \mathcal{N}(\mathbf{c}_{z_m^{(n)}}; \boldsymbol{\mu}_0, \mathbf{V}_0) \Big) \times \prod_{x,y \in \tilde{z}_m^{(n)}} \sqrt{\exp(g(z_m^{(n)}, z_x^{(y)}))} \Bigg) \tag{4}$$

where $x, y \in \tilde{z}_m^{(n)}$ denotes the set of all word labels (the yth word in the xth document) connected with $z_m^{(n)}$, i.e. $\delta(z_m^{(n)}, z_x^{(y)}) = 1$, and $\mathbf{Q}$ is the normalization term induced by Markov random fields.

2.3 Inference and learning

Similar to the inference and learning in LDA, based on (4), we can develop a collapsed Gibbs sampler by integrating out $\boldsymbol{\pi}_m, \boldsymbol{\theta}_{z_m^{(n)}}, \mathbf{c}_{z_m^{(n)}}$ in SLMRF. The sampler can be more easily interpreted as a "combined" sampler by using clues from LDA, MRF and Guassian mixtures:

$$q^*(z_m^{(n)} = k) \propto q_{\text{LDA}}(z_m^{(n)} = k) \cdot q_M(z_m^{(n)} = k) \cdot q_c(z_m^{(n)} = k) \tag{5}$$

where $q_{\text{LDA}}(z_m^{(n)} = k)$ is the collapsed Gibbs sampler of LDA (1),

$$q_M(z_m^{(n)} = k) \propto \frac{\exp\left(\sum_{(z_j, z_m^{(n)})} g(z_j, z_m^{(n)} = k)\right)}{\sum_h \exp\left(\sum_{(z_j, z_m^{(n)})} g(z_j, z_m^{(n)} = h)\right)} \tag{6}$$

is the Gibbs sampler based on the Markov random field, and

$$q_c(z_m^{(n)} = k) \propto \frac{\mathcal{N}(\hat{w}_m^{(n)}; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Lambda}_l^{(k)})}{\sum_h \mathcal{N}(\hat{w}_m^{(n)}; \boldsymbol{\mu}_l^{(h)}, \boldsymbol{\Lambda}_l^{(h)})} \tag{7}$$

is a collapsed Gibbs sampler of Gaussian mixtures, with

$$\boldsymbol{\Lambda}_l^{(k)^{-1}} = \boldsymbol{\Lambda}_0^{-1} + l\boldsymbol{\Lambda}^{-1} \qquad \boldsymbol{\mu}_l^{(k)} = \boldsymbol{\Lambda}_l^{(k)^2} \left(\frac{\boldsymbol{\mu}_0}{\boldsymbol{\Lambda}_0^2} + \frac{l\overline{\hat{w}^{(k)}}}{\boldsymbol{\Lambda}^2} \right) \tag{8}$$

where l is the number of words which are labeled with k, and $\overline{\hat{w}^{(k)}}$ is the mean of 3D coordinates of words which are assigned to part k until the current iteration. Similar to [6], parameters $\{\boldsymbol{\pi}_m\}_{m=1}^M$, $\{\boldsymbol{\theta}_k\}_{k=1}^K$ can be estimated after the convergence of Gibbs sampling:

$$\boldsymbol{\pi}_m^{(k)} = \frac{\mathbf{N}_k^{(m)} + \alpha_k}{\sum_{k=1}^K (\mathbf{N}_k^{(m)} + \alpha_k)} \qquad \boldsymbol{\theta}_k^{(w)} = \frac{\mathbf{N}_w^{(k)} + \beta_w}{\sum_w^W (\mathbf{N}_w^k + \beta_w)} \tag{9}$$

Since hyperparameter $\boldsymbol{\alpha}$, in our case, is categorical part weight, we estimate it by simply compute the average of π_m: $\boldsymbol{\alpha} = \frac{1}{M} \sum_{m=1}^M \boldsymbol{\pi}_m$. In addition, parameters $\{\mathbf{c}_k\}_{k=1}^K$ are read out as $\{\boldsymbol{\mu}_l^k\}_{k=1}^K$ (8).

3 Experiments

To evaluate the proposed model, 5 object classes (cars, bikes, dogs, motorcycles, airplanes) from the Princeton shape benchmark (PSB) [9] database are used. Since 3D shapes in the PSB are represented as triangulated meshes, we convert them to point clouds by uniformly sampling points within triangles.

3.1 3D Object class geometry modeling

For comparison, LDA [1], LDMRF [8] and Gaussian Mixtures (GM) models are tested on the same data (aligned point clouds of categorical instances). Since LDMRF requires manual interference (semi-supervision), to avoid human bias during comparison, here we construct the same Markov random fields for both LDMRF and SLDMRF so that LDMRF can also work in an unsupervised manner. All four models are implemented with Gibbs sampling for label inference and learning. To ensure fairness, the same part number and iteration number (200) is applied. A test example of motorcycle modeling is presented in Figure 5. LDA does not find consistent and meaningful parts because of the intra-class variation (each object is labeled as a part since LDA only focuses on co-occurring patterns). LDMRF, on the other hand, discovers some locally continuous and consistent segments on different objects. However, the global spatial coherence of parts is poor; parts are shattered. GM establishes more globally obvious segmentation pattern by finding more consistent and meaningful parts. Nevertheless, GM ignores local spatial coherence, so parts are not well segmented; they tend to be of blob shape and to overlap each other. By contrast, SLDMRF produces best convincing segmentations in terms of consistence, local and global spatial coherence and semantic meaning. The SLDMRF modeling results of other four object classes are illustrated in Figure 6.

3.2 Geometry model classification

To illustrate the parts learned by SLDMRF is more accurate, and thus more discriminative, we conduct quantitative comparisons on classification task. Since LDA and LDMRF are far from being qualified for practical part-based modeling, here we are only concerned with the comparison between GM and SLDMRF. 3D shapes of 5 object classes are divided into training (70%) and test sets (30%). The model learning is conducted in the same way as in section 3.1. Although Markov random fields and spatial parameters greatly assists in segmentation and model learning of SLDMRF, they are not used in the final category models. A learned bicycle model is shown in Figure 6(e). It can be seen that the part position information and neighboring correlation are already encoded in the categorical part parameter $\boldsymbol{\theta}$. Therefore, for the sake of simplicity and computation feasibility, we only use learned LDA as our 3D object category models for classification. To test an object M^*, it is first aligned to the canonical poses of different class models. In SLDMRF case, the likelihood that M^* belongs to a

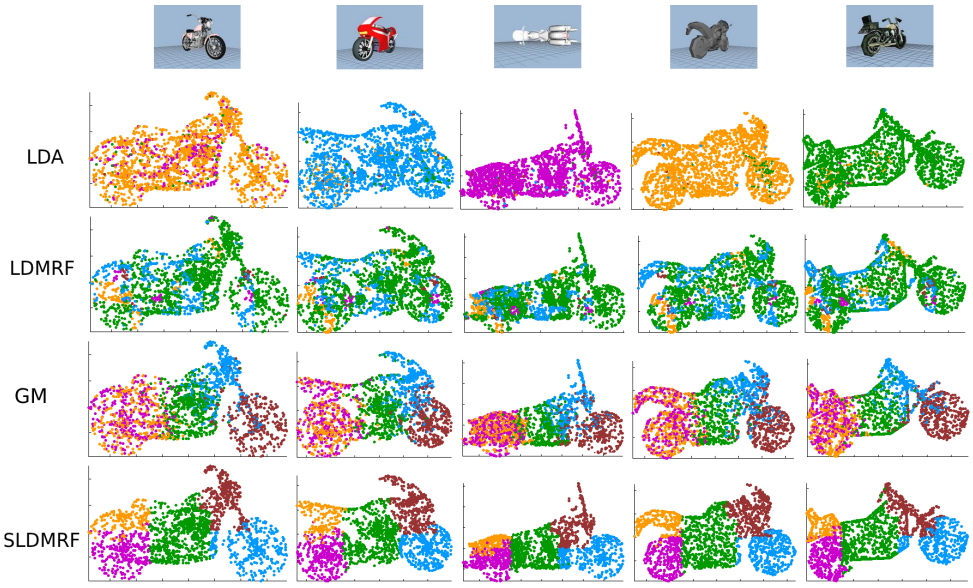


Fig. 5. Comparison of segmentation by using LDA, LDMRF, GM and SLDMRF, SLDMRF qualitatively yields more reasonable segmentations than the others.

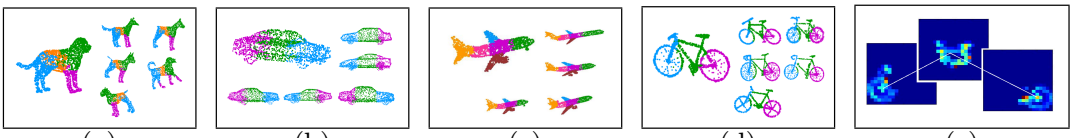


Fig. 6. SLDMRF modeling of dogs (a), cars (b), airplanes (c) and bikes (d); (e): the learned part parameters $\boldsymbol{\theta}_k$ of bikes.

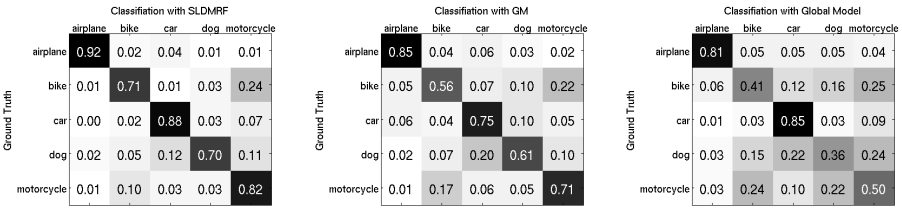
class $x \in \{\text{cars, bikes, dogs, motorcycles, airplanes}\}$ is computed as:

$$p(M^*|\mathcal{M}_x) = \prod_{i=1}^{|M^*|} \left\{ \sum_k \int_{\boldsymbol{\pi}} p(w_i|\boldsymbol{\theta}_k) p(k|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{\alpha}) \right\} \tag{10}$$

where $|M^*|$ is the number of points in object M^*. By contrast, in the GM case:

$$p(M^*|\mathcal{M}_x) = \prod_{i=1}^{|M^*|} \left\{ \sum_k \mathcal{N}(w_i; \boldsymbol{\theta}_k) \pi_k \right\} \tag{11}$$

Since no other prior knowledge is given, the classification can be done in a maximum-likelihood fashion. A global model learned with one single multinomial distribution on 3D dictionary is also provided as baseline for comparison. The classification performances of GM, SLDMRF and global model are evaluated using confusion matrices. The comparison in in Figure 7 demonstrates that SLDMRF is superior to GM with respect to discriminative ability in classification.



(a) (b) (c)

Fig. 7. The classification confusion matrcices of 5 object classes with SLDMRF (a), GM (b) and global model (c).

4 Conclusion and Discussion

We improved LDA model for geometry modeling with better semantic interpretation and promising discriminative capabilities. Meanwhile, learning and application of the model require good initial alignment, which is difficult for noisy and partial occluded 3D point cloud in practice. So a promising future work is to cooperate 3D geometry model with 2D image models to describe both structure and appearance, which thus enhance model's expressiveness and practical value.

References

1. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Detry, R., Piater, J.: Continuous Surface-Point Distributions for 3D Object Pose Estimation and Recognition. In: ACCV (2010)
3. Detry, R., Pugeault, N., Piater, J.: A Probabilistic Framework for 3D Visual Object Representation. PAMI 31(10), 1790–1803 (10 2009)
4. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part-Based Models. PAMI 32(9), 1627–1645 (2010)
5. Golovinskiy, A., Funkhouser, T.A.: Consistent segmentation of 3D models. Computers and Graphics 33, 262–269 (2009)
6. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences 101(Suppl. 1), 5228–5235 (April 2004)
7. Jörg Liebelt and Cordelia Schmid: Multi-View Object Class Detection with a 3D Geometric Model. In: CVPR (2010)
8. Mackey, L.: Latent Dirichlet Markov Random Fields for Semi-supervised Image Segmentation and Object Recognition (2007)
9. Shilane, P., Min, P., Kazhdan, M.M., Funkhouser, T.A.: The Princeton Shape Benchmark. In: SMI. pp. 167–178. IEEE Computer Society (2004)
10. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering object categories in image collections. In: ICCV (2005)
11. Wang, X., Grimson, E.: Spatial Latent Dirichlet Allocation. In: NIPS (2007)
12. Xiong, H., Szedmak, S., Piater, J.: Efficient,General Point Cloud Registration with Kernel Feature Maps. In: Canadian Conf. on Computer and Robot Vision (2013)
13. Yan, P., Khan, S.M., Shah, M.: 3D model based object class detection in an arbitrary view. In: ICCV (2007)

Chapter 3

Graph-Based Structural Output Learning

"To learn without thinking is blindness; to think without learning is idleness. "

Confucius

3.1 Conditional Random Field

For instance, in a multi-label learning case, input $\mathbf{x} \in \mathbb{R}^D$ is a D-dimensional real-value vector, output $\mathbf{y} \in \mathbb{B}^L$ is a L-dimensional binary vector with a component $y_{l:l\in[1:L]}$ denoting the presence of the lth label. A *Markov random field* (MRF) $p(\mathbf{x}, \mathbf{y})$ can be defined by a domain expert, or learned from existing training data, with a graph structure and associated potential functions, which are supposed to reconstruct the generative process of $\{\mathbf{x}, \mathbf{y}\} : p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$. As for the prediction task, the conditional probability $p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{y})}{\sum_{\mathbf{y}} p(\mathbf{x},\mathbf{y})}$ can be derived, or more straightforwardly, $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$ (*e.g.* by using max-product algorithm). However, it was realised that for the task $\mathbf{x} \to \mathbf{y}$, the only desired component is $p(\mathbf{y}|\mathbf{x})$; it is wast to expend effort in modeling $p(\mathbf{x})$, which is not relevant anyway. Consequently, *conditional random field* (CRF) was proposed (Lafferty et al., 2001) as a discriminative model by learning only the conditional probability $p(\mathbf{y}|\mathbf{x})$. One strength of CRFs is that it can include arbitrarily complex features (*e.g.* higher order dependencies) of $\mathbf{x}$, which

is intractable in MRFs. According to empirical study in (Kumar and Hebert, 2003a, Lafferty et al., 2001), CRFs outperform MRFs in many practical applications.

At first, the definition of conditional random fields (CRFs) give as (Lafferty et al., 2001):

Definition 3.1. Let $G = (V, E)$ be a graph (V are vertices and E are edges) such that $\mathbf{y} = \{y_v\}_{v \in V}$, so that y is indexed by the vertices of G. Then $(\mathbf{x}, \mathbf{y})$ is a conditional random field in case, when conditioned on $\mathbf{x}$, the random variables $\{y_v\}_{v \in V}$ obey the Markov property with respect to the graph G.

Assume that the joint conditional probability $p(\mathbf{y}|\mathbf{x})$ is defined in an exponential form:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}))}{\mathbf{Z}(\mathbf{w})} \tag{3.1}$$

where $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$, $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^K$ is a collection of K predefined, task-specific potential functions (or feature vectors) $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = [\phi_1(\mathbf{x}, \mathbf{y}), \cdots \phi_K(\mathbf{x}, \mathbf{y})]^\top$, $\mathbf{w} \in \mathbb{R}^K$ is the parameter vector, $\mathbf{Z}(\mathbf{w})$ is the partition function for normalization $\mathbf{Z}(\mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}))$.

As a probabilistic model, one CRF is usually learned via *maximum likelihood estimation* (MLE) or *maximum a posterior* (MAP) (Kumar and Hebert, 2003a, Lafferty et al., 2001, Vishwanathan et al., 2006). Meanwhile, inspired by the margin notion which has been exploited in successful binary classifiers, *e.g.* Support Vector Machine (SVM), an alternative discriminative training scheme of CRFs is maximizing the margin between the $p(\mathbf{y}|\mathbf{x})$ of the desired $\mathbf{y}^*$ and the best runner-up $\mathbf{y}^{**}$ in their log domains. The CRFs trained with this margin are referred to as *Max-Margin Markov Networks* (M^3Ns) (Taskar et al., 2003). On the one hand, similarly to regular SVM, *kernel tricks* can be used in the dual form of M^3N, in which way, the original inputs and/or outputs are implicitly mapped into a higher dimensional *reproducing kernel Hilbert space* (RKHS) (*e.g.* polynomial kernels used in (Taskar et al., 2003)). On the other hand, different from regular SVM, the *quadratic programming* (QP) problem associated with M^3Ns (in both primal and dual form) has exponential number of constraints. Although it can be simplified by exploiting interesting properties within forest-structured outputs (Taskar et al., 2003), for highly connected graphs, the learning is intractable. Obviously, although CRF and M^3Ns offer almost equivalent modeling capabilities for a given machine learning task, two rather different loss functions are employed in them. However, it can be revealed that MAP and max-margin learning schemes are, to large extent, related.

3.1.1 Maximum Likelihood Estimation

Given a training dataset $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{M}$, a CRF can be learned via maximum likelihood estimation (MLE):

$$\begin{aligned} \mathbf{w}^* &= \arg\max_{\mathbf{w}} \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \\ &= \arg\max_{\mathbf{w}} \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \{\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \log \textstyle\sum_{\mathbf{y}\in\mathcal{Y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}))\} \end{aligned} \tag{3.2}$$

Meanwhile, usually, in order to enhance the generalization properties, a prior on $p(\mathbf{w}) \propto \exp(-\frac{\mathbf{w}^2}{\sigma^2})$ is added in (3.2) to penalize the complexity of $\mathbf{w}$, which results in the maximum a posterior (MAP) solution:

$$\begin{aligned} \mathbf{w}^* &= \arg\max_{\mathbf{w}} -\tfrac{\mathbf{w}^2}{\sigma^2} + \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \{\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \log \textstyle\sum_{\mathbf{y}\in\mathcal{Y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}))\} \\ &= \arg\min_{\mathbf{w}} \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{M} \textstyle\sum_{i=1}^{M} \mathcal{L}_i^{MAP} \end{aligned} \tag{3.3}$$

where $\mathcal{L}_i^{MAP}$ is the negative log-likelihood of the ith training instance:

$$\mathcal{L}_i^{MAP} = \log \sum_{\mathbf{y}\in\mathcal{Y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y})) - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \tag{3.4}$$

and $C = \frac{\sigma^2}{2}$ is a trade-off parameter to balance the regularization term $\frac{1}{2}||\mathbf{w}||^2$ and average loss $\frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i^{MAP}$. A good property of (3.3) is that it is convex with respect to $\mathbf{w}$.

Proposition 3.2. *The objective function of maximum a posterior in learning CRFs is convex.*

Proof. The second order derivative of $\log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}))$ with respect to $\mathbf{w}$ is:

$$\frac{d^2 \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}))}{d\mathbf{w}^2} = \mathbf{Cov}_{p(\mathbf{y}|\mathbf{x}^{(i)};\mathbf{w})} \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}) \tag{3.5}$$

where $\mathbf{Cov}_{p(\boldsymbol{\rho})} f(\boldsymbol{\rho})$ denotes the covariance of $f(\boldsymbol{\rho})$ under the probability $p(\boldsymbol{\rho})$. Since the covariance function is always positive semidefinite, $\log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}))$ is convex. In addition, $-\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is linear (thus also convex), and $\frac{1}{2}||\mathbf{w}||^2$ is convex. Therefore, (3.3) (the sum of convex components) is convex. □

Because of the convexity of the objective function, any local minimum is also global minimum, and gradient descent can be employed to find the unique solution $\mathbf{w}^*$ with the negative gradient computed as:

$$\Delta \mathbf{w}_t = \frac{C}{M} \sum_{i=1}^{M} (\boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbb{E}_{p(\mathbf{y}|\mathbf{x}^{(i)};\mathbf{w}_t)} \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y})) - \mathbf{w}_t \tag{3.6}$$

where $\mathbb{E}_{p(\boldsymbol{\rho})} f(\boldsymbol{\rho})$ is the expectation of $f(\boldsymbol{\rho})$ under the probability $p(\boldsymbol{\rho})$. The difficulty of computing (3.6) is the expectation term because its complexity grows exponentially. For example, in multi-label learning case introduced in section 3.1, $|\mathcal{Y}| = 2^L$. Therefore, when $|\mathcal{Y}|$ is relatively large in practice, some approximations are used, *e.g.* by using *mean field* (MF), *loopy belief propagation* (LBP) or *pseudo likelihood* (PL) (Kumar and Hebert, 2003a, Lafferty et al., 2001, Vishwanathan et al., 2006, Yedidia et al., 2005).

3.1.2 Max-Margin Markov Network

The max-margin principle used in binary classifier, *e.g.* Support Vector Machine (SVM), has been proven theoretically and empirically (Vapnik, 1995) superior to others. The margin notion was generalized to learn structured outputs in (Taskar et al., 2003) and (Tsochantaridis et al., 2004). More concretely, they proposed to learn CRFs by maximizing the margin between the $p(\mathbf{y}|\mathbf{x})$ of the desired $\mathbf{y}^*$ and the best runner-up $\mathbf{y}^{**}$ in log domain. The CRFs trained with maximum margin are referred to as max-margin Markov networks (M^3N) (Taskar et al., 2003). The objective function of M^3N can be written out as:

$$\begin{aligned} & \arg\min_{\mathbf{w},\xi_{i:i\in[1,M]}} \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{M}\textstyle\sum_{i=1}^{M}\xi_i \\ s.t. \quad & \forall i, \forall y \in \mathcal{Y}/\mathbf{y}^{(i)} : \xi_i \geq 0, \mathbf{w}^\top(\boldsymbol{\phi}(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \boldsymbol{\phi}(\mathbf{x}^{(i)},\mathbf{y})) \geq 1 - \tfrac{\xi_i}{d(\mathbf{y}^{(i)},\mathbf{y})} \end{aligned} \tag{3.7}$$

where ξ_i are slack variables for the relaxation of max-margin, and $d(\mathbf{y}^{(i)},\mathbf{y}) \geq 0$ is a measure of dissimilarity between $\mathbf{y}^{(i)}$ and $\mathbf{y}$ (*e.g.* hamming distance used in the multilable learning case). The term $\frac{\xi_i}{d(\mathbf{y}^{(i)},\mathbf{y})}$ is to rescale slack variables by the dissimilarities between corresponding $\mathbf{y}^{(i)}$ and others. Obviously, (3.7) is a quadratic programming (QP) problem with exponential number of constraints. Similarly to SVM, (3.7) can be converted to its dual form, of which some interesting properties can be exploited to simplify the computation for forest-structured outputs (*i.e.* singly-connected graphs). However, for highly-connected graphs, exact learning is intractable. An advantage of the dual form of M^3N is that it enables the use of *kernel methods*, which can implicitly project $\boldsymbol{\phi}(\mathbf{x},\mathbf{y})$ to a higher dimensional Hilbert feature space. Defining $\boldsymbol{\phi}(\mathbf{x},\mathbf{y})$ with kernel methods was more exploited in Structural SVM (Tsochantaridis et al., 2004) (see section **??**). Usually quadratic or cubic kernels are used and yield promising results (Taskar et al., 2003). Correspondingly, explicitly feature map of $\boldsymbol{\phi}(\mathbf{x},\mathbf{y})$ can be constructed via two-degree and three-degree polynomials in the primal form of M^3Ns.

The second constraint in (3.7) can be written in a more compact form: $\max_{\mathbf{y}\in\mathcal{Y}}\{d(\mathbf{y}^{(i)},\mathbf{y}) - \mathbf{w}^\top(\boldsymbol{\phi}(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \boldsymbol{\phi}(\mathbf{x}^{(i)},\mathbf{y}))d(\mathbf{y}^{(i)},\mathbf{y}\} \leq \xi_i$. In addition, we can rescale $\mathbf{w}$ ($\mathbf{w} \leftarrow \mathbf{w}d(\mathbf{y}^{(i)},\mathbf{y})$) . Then, instead of rescaling slack variables, we rescale margin to be the

same level as $d(\mathbf{y}^{(i)}, \mathbf{y})$. Therefore, Eq. (3.7) can be reformulated in a similar form as Eq. (3.3):

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{M}\sum_{i=1}^{M} \mathcal{L}_i^{Margin} \tag{3.8}$$

where

$$\mathcal{L}_i^{Margin} = \max_{\mathbf{y}\in\mathcal{Y}}\{d(\mathbf{y}^{(i)}, \mathbf{y}) + \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y})\} - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \tag{3.9}$$

3.2 Training Undirected Graphical Models with Persistent Sequential Monte Carlo

II. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Towards Maximum Likelihood: Learning Undirected Graphical Models using Persistent Sequential Monte Carlo*, The 6th Asian Conference on Machine Learning (ACML2014),

JMLR: Workshop and Conference Proceedings 39:1–16, 2014 ACML 2014

Towards Maximum Likelihood: Learning Undirected Graphical Models using Persistent Sequential Monte Carlo

Hanchen Xiong HANCHEN.XIONG@UIBK.AC.AT
Sandor Szedmak SANDOR.SZEDMAK@UIBK.AC.AT
Justus Piater JUSTUS.PIATER@UIBK.AC.AT
Institute of Computer Science, University of Innsbruck
Technikerstr. 21a, A-6020 Innsbruck, Austria

Abstract

Along with the emergence of algorithms such as persistent contrastive divergence (PCD), tempered transition and parallel tempering, the past decade has witnessed a revival of learning undirected graphical models (UGMs) with sampling-based approximations. In this paper, based upon the analogy between Robbins-Monro's stochastic approximation procedure and sequential Monte Carlo (SMC), we analyze the strengths and limitations of state-of-the-art learning algorithms from an SMC point of view. Moreover, we apply the rationale further in sampling at each iteration, and propose to learn UGMs using *persistent sequential Monte Carlo* (PSMC). The whole learning procedure is based on the samples from a long, persistent sequence of distributions which are actively constructed. Compared to the above-mentioned algorithms, one critical strength of PSMC-based learning is that it can explore the sampling space more effectively. In particular, it is robust when learning rates are large or model distributions are high-dimensional and thus multi-modal, which often causes other algorithms to deteriorate. We tested PSMC learning, also with other related methods, on carefully-designed experiments with both synthetic and real-world data, and our empirical results demonstrate that PSMC compares favorably with the state of the art.

Keywords: Sequential Monte Carlo, maximum likelihood learning, undirected graphical models.

1. Introduction

Learning undirected graphical models (UGMs), or Markov random fields (MRF), has been an important yet challenging machine learning task. On the one hand, thanks to its flexible and powerful capability in modeling complicated dependencies, UGMs are prevalently used in many domains such as computer vision, natural language processing and social analysis. Undoubtedly, it is of great significance to enable UGMs' parameters to be automatically adjusted to fit empiric data, *e.g.* maximum likelihood (ML) learning. A fortunate property of the likelihood function is that it is concave with respect to its parameters (Koller and Friedman, 2009), and therefore gradient ascent can be applied to find the unique maximum. On the other hand, learning UGMs via ML in general remains intractable due to the presence of the partition function. Monte Carlo estimation is a principal solution to the problem. For example, one can employ Markov chain Monte Carlo (MCMC) to obtain samples from the model distribution, and approximate the partition function with the samples. However, the sampling procedure of MCMC is very inefficient because it usually requires a large number of steps for the Markov chain to reach equilibrium. Even though in some cases where efficiency can be ignored, another weakness of MCMC estimation is that it yields large estimation variances. A more practically feasible alternative is MCMC maximum likelihood (MCMCML;

Geyer 1991); see section 2.1. MCMCML approximates the gradient of the partition function with importance sampling, in which a proposal distribution is initialized to generate a fixed set of MCMC samples. Although MCMCML increases efficiency by avoiding MCMC sampling at every iteration, it also suffers from high variances (with different initial proposal distributions). Hinton (2002) studied *contrastive divergence* (CD) to replace the objective function of ML learning. This turned out to be an efficient approximation of the likelihood gradient by running only a few steps of Gibbs sampling, which greatly reduces variance as well as the computational burden. However, it was pointed out that CD is a biased estimation of ML (Carreira-Perpinan and Hinton, 2005), which prevents it from being widely employed (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010). Later, a *persistent* version of CD (PCD) was put forward as a closer approximation of the likelihood gradient (Tieleman, 2008). Instead of running a few steps of Gibbs sampling from training data in CD, PCD maintains an almost persistent Markov chain throughout iterations by preserving samples from the previous iteration, and using them as the initializations of Gibbs samplers in the current iteration. When the learning rate is sufficiently small, samples can be roughly considered as being generated from the stationary state of the Markov chain. However, one critical drawback in PCD is that Gibbs sampling will generate highly correlated samples between consecutive weight updates, so mixing will be poor before the model distribution gets updated at each iteration. The limitations of PCD sparked many recent studies of more sophisticated sampling strategies for effective exploration within data space (section 3). For instance, Salakhutdinov (2010) studied *tempered transition* (Neal, 1994) for learning UGMs. The strength of tempered transition is that it can make potentially big transitions by going through a trajectory of intermediary Gibbs samplers which are smoothed with different temperatures. At the same time, *parallel tempering*, which can be considered a parallel version of tempered transition, was developed by Desjardins et al. (2010) for training restricted Boltzmann machines (RBMs). Contrary to a single Markov chain in PCD and tempered transition, parallel tempering maintains a pool of Markov chains governed by different temperatures. Multiple tempered chains progress in parallel and are mixed at each iteration by randomly swapping the states of neighbouring chains.

The contributions of this paper are twofold. The first is theoretic. By linking Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) and sequential Monte Carlo (SMC), we cast PCD and other state-of-the-art learning algorithms into a SMC-based interpretation framework. Moreover, within the SMC-based interpretation, two key factors which affect the performance of learning algorithms are disclosed: *learning rate* and *model complexity* (section 4). Based on this rationale, the strengths and limitations of different learning algorithms can be analyzed and understood in a new light. The second contribution is practical. Inspired by the understanding of learning UGMs from a SMC perspective, and the successes of global tempering used in parallel tempering and tempered transition, we put forward a novel approximation-based algorithm, *persistent SMC* (PSMC), to approach the ML solution in learning UGMs. The basic idea is to construct a long, persistent distribution sequence by inserting many tempered intermediary distributions between two successively updated distributions (section 5). According to our empirical results on learning two discrete UGMs (section 6), the proposed PSMC outperforms other learning algorithms in challenging circumstances, *i.e.* large learning rates or large-scale models.

2. Learning Undirected Graphical Models

In general, we can define undirected graphical models (UGMs) in an energy-based form:

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{\exp\left(-E(\mathbf{x};\boldsymbol{\theta})\right)}{\mathbf{Z}(\boldsymbol{\theta})} \tag{1}$$

$$\textit{Energy function:} \quad E(\mathbf{x};\boldsymbol{\theta}) = -\boldsymbol{\theta}^\top \phi(\mathbf{x}) \tag{2}$$

with random variables $\mathbf{x} = [x_1, x_2, \ldots, x_D] \in \mathcal{X}^D$ where x_d can take N_d discrete values, $\phi(\mathbf{x})$ is a K-dimensional vector of sufficient statistics, and parameter $\boldsymbol{\theta} \in \mathbb{R}^K$. $\mathbf{Z}(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))$ is the partition function for global normalization. Learning UGMs is usually done via maximum likelihood (ML). A critical observation of UGMs' likelihood functions is that they are concave with respect to $\boldsymbol{\theta}$, therefore any local maximum is also global maximum (Koller and Friedman, 2009), and gradient ascent can be employed to find the optimal $\boldsymbol{\theta}^*$. Given training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^M$, we can compute the derivative of average log-likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^M \log p(\mathbf{x}^{(m)};\boldsymbol{\theta})$ as

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \underbrace{\mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x}))}_{\psi^+} - \underbrace{\mathbb{E}_{\boldsymbol{\theta}}(\phi(\mathbf{x}))}_{\psi^-}, \tag{3}$$

where $\mathbb{E}_{\mathcal{D}}(\xi)$ is the expectation of ξ under the empirical data distribution $p_{\mathcal{D}} = \frac{1}{M}\sum_{m=1}^M \delta(\mathbf{x}^{(m)})$, while $\mathbb{E}_{\boldsymbol{\theta}}(\xi)$ is the expectation of ξ under the model probability with parameter $\boldsymbol{\theta}$. The first term in (3), which is often referred to as *positive phase* ψ^+, can be easily computed as the average of $\phi(\mathbf{x}^{(m)}), \mathbf{x}^{(m)} \in \mathcal{D}$. The second term in (3), also known as *negative phase* ψ^-, however, is not trivial because it is a sum of $\prod_{d=1}^D N_d$ terms, which is only computationally feasible for UGMs of very small size. Markov chain Monte Carlo (MCMC) can be employed to approximate ψ^-, although it is usually expensive and leads to large estimation variances. The underlying procedure of ML learning with gradient ascent, according to (3), can be envisioned as a behavior that iteratively pulls down the energy of the data space occupied by $\mathcal{D}$ (positive phase), but raises the energy over all data space $\mathcal{X}^D$ (negative phase), until it reaches a balance ($\psi^+ = \psi^-$).

2.1. Markov Chain Monte Carlo Maximum Likelihood

A practically feasible approximation of (3) is Markov chain Monte Carlo maximum likelihood (MCMCML; Geyer 1991). In MCMCML, a proposal distribution $p(\mathbf{x};\boldsymbol{\theta_0})$ is set up in the same form as (1) and (2), and we have

$$\frac{\mathbf{Z}(\boldsymbol{\theta})}{\mathbf{Z}(\boldsymbol{\theta}_0)} = \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} \tag{4}$$

$$= \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))}{\exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} \times \frac{\exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} \tag{5}$$

$$= \sum_{\mathbf{x}} \exp\left((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \phi(\mathbf{x})\right) p(\mathbf{x};\boldsymbol{\theta}_0) \tag{6}$$

$$\approx \frac{1}{S}\sum_{s=1}^S w^{(s)} \tag{7}$$

Algorithm 1 MCMCML Learning Algorithm

Input: training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate η; gap L between two successive proposal distribution resets

$t \leftarrow 0$, initialize the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$
while ! stop criterion **do**
  if $(t \mod L) == 0$ **then**
    (Re)set the proposal distribution as $p(\mathbf{x}; \boldsymbol{\theta}_t)$
    Sample $\{\bar{\mathbf{x}}^{(s)}\}$ from $p(\mathbf{x}; \boldsymbol{\theta}_t)$
  end if
  Calculate $w^{(s)}$ using (8)
  Calculate gradient $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$ using (9)
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$
  $t \leftarrow t + 1$
end while

Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$

where $w^{(s)}$ is

$$w^{(s)} = \exp\left((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \phi(\bar{\mathbf{x}}^{(s)})\right), \tag{8}$$

and the $\bar{\mathbf{x}}^{(s)}$ are sampled from the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$. By substituting $\mathbf{Z}(\boldsymbol{\theta}) = \mathbf{Z}(\boldsymbol{\theta}_0)\frac{1}{S}\sum_{s=1}^{S} w^{(s)}$ into (1) and average log-likelihood, we can compute corresponding gradient as (note $\mathbf{Z}(\boldsymbol{\theta}_0)$ will be eliminated since it corresponds to a constant in the logarithm)

$$\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\boldsymbol{\theta}_0}(\phi(\mathbf{x})), \tag{9}$$

where $\mathbb{E}_{\boldsymbol{\theta}_0}(\xi)$ is the expectation of ξ under a *weighted* empirical data distribution $p_{\boldsymbol{\theta}_0} = \sum_{s=1}^{S} w^{(s)}\delta(\bar{\mathbf{x}}^{(s)}) / \sum_{s=1}^{S} w^{(s)}$ with data sampled from $p(\mathbf{x}; \boldsymbol{\theta}_0)$. From (9), it can be seen that MCMCML does nothing more than an importance sampling estimation of ψ^- in (3). MCMCML has the nice asymptotic convergence property (Salakhutdinov, 2010) that it will converge to the exact ML solution when the number of samples S goes to infinity. However, as an inherent weakness of importance sampling, the performance of MCMCML in practice highly depends on the choice of the proposal distribution, which results in large estimation variances. The phenomenon gets worse when it scales up to high-dimensional models. One engineering trick to alleviate this pain is to reset the proposal distribution, after a certain number of iterations, to the recently updated estimation $p(\mathbf{x}; \boldsymbol{\theta}^{estim})$ (Handcock et al., 2007). Pseudocode of the MCMCML learning algorithm is presented in Algorithm 1.

3. State-of-the-art Learning Algorithms

Contrastive Divergence (CD) is an alternative objective function of likelihood (Hinton, 2002), and turned out to be de facto a cheap and low-variance approximation of the maximum likelihood (ML) solution. CD tries to minimize the discrepancy between two Kullback-Leibler (KL) divergences, $KL(p^0|p_{\boldsymbol{\theta}}^{\infty})$ and $KL(p_{\boldsymbol{\theta}}^{n}|p_{\boldsymbol{\theta}}^{\infty})$, where $p^0 = p(\mathcal{D}; \boldsymbol{\theta})$, $p_{\boldsymbol{\theta}}^{n} = p(\bar{\mathcal{D}}_n; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_n$ denoting the data sampled after n steps of Gibbs sampling with parameter $\boldsymbol{\theta}$, and $p_{\boldsymbol{\theta}}^{\infty} = p(\bar{\mathcal{D}}_{\infty}; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_{\infty}$ denoting the

data sampled from the equilibrium of a Markov chain. Usually $n = 1$ is used, and correspondingly it is referred to as the CD-1 algorithm. The negative gradient of CD-1 is

$$-\frac{\partial\big(CD_1(\mathcal{D};\boldsymbol{\theta})\big)}{\partial\boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\bar{\mathcal{D}}_1}(\phi(\mathbf{x})) \tag{10}$$

where $\mathbb{E}_{\bar{\mathcal{D}}_1}(\xi)$ is the expectation of ξ under the distribution $p^1_{\boldsymbol{\theta}}$. The key advantage of CD-1 is that it efficiently approximates ψ^- in the likelihood gradient (3) by running only one step Gibbs sampling. While this local exploration of sampling space can avoid large variances, CD-1 was theoretically (Carreira-Perpinan and Hinton, 2005) and empirically (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010) proved to be a biased estimation of ML .

Persistent Contrastive Divergence (PCD) is an extension of CD by running a nearly persistent Markov chain. For approximating ψ^- in likelihood gradient (3), the samples at each iteration are retained as the initialization of Gibbs sampling in the next iteration. The mechanism of PCD was usually interpreted as a case of Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) with Gibbs sampling as transitions. In general SAP, if the learning rate η is sufficiently small compared to the mixing rate of the Markov chain, the chain can be roughly considered as staying close to the equilibrium distribution (i.e. PCD$\rightarrow$ML when $\eta \rightarrow 0$). Nevertheless, Gibbs sampling as used in PCD heavily hinders the exploration of data space by generating highly correlated samples along successive model updates. This hindrance becomes more severe when the model distribution is highly multi-modal. Although multiple chains (mini-batch learning) used in PCD can mitigate the problem, we cannot generally expect the number of chains to exceed the number of modes. Therefore, at the late stage of learning, PCD usually gets stuck in a local optimum, and in practice, small and linearly-decayed learning rates can improve the performance (Tieleman, 2008).

Tempered Transition was originally developed by Neal (1994) to generate relatively big jumps in Markov chains while keeping reasonably high acceptance rates. Instead of standard Gibbs sampling used in PCD, tempered transition constructs a sequence of Gibbs samplers based on the model distribution specified with different temperatures:

$$p_h(\mathbf{x};\boldsymbol{\theta}) = \frac{\exp(-E(\mathbf{x};\boldsymbol{\theta})\beta_h)}{\mathbf{Z}(h)} \tag{11}$$

where h indexes temperatures $h \in [0, H]$ and β_H are inverse temperatures $0 \leq \beta_H < \beta_{H-1} < \cdots \beta_0 = 1$. In particular, β_0 corresponds to the original complex distribution. When h increases, the distribution gets more flat, where Gibbs samplers can more adequately explore. In tempered transition, a sample is generated with a Gibbs sampler starting from the original distribution. It then goes through a trajectory of Gibbs sampling through sequentially tempered distributions (11). A backward trajectory is then run until the sample reaches the original distribution. The acceptance of the final sample is determined by the probability of the whole forward-and-backward trajectory. If the trajectory is rejected, the sample does not move at all, which is even worse than local movements of Gibbs sampling, so β_H is set relatively high (0.9 in Salakhutdinov 2010) to ensure high acceptance rates.

Parallel Tempering, on the other hand, is a "parallel" version of Tempered Transition, in which smoothed distributions (11) are run with one step of Gibbs sampling in parallel at each iteration. Thus, samples native to more uniform chains will move with larger transitions, while samples native

to the original distribution still move locally. All chains are mixed by swapping samples of randomly selected neighbouring chains. The probability of the swap is

$$r = \exp\left((\beta_h - \beta_{h+1})(E(\mathbf{x}_h) - E(\mathbf{x}_{h+1}))\right) \tag{12}$$

Although multiple Markov chains are maintained, only samples at the original distribution are used. In the worst case (there is no swap between β_0 and β_1), parallel tempering degrades to PCD-1. β_H can be set arbitrarily low (0 was used by Desjardins et al. 2010).

4. Learning as Sequential Monte Carlo

Before we delve into the analysis of the underlying mechanism in different learning algorithms, it is better to find a unified interpretation framework, within which the behaviors of all algorithms can be more apparently viewed and compared in a consistent way. In most previous work, PCD, tempered transition and parallel tempering were studied as special cases of Robbins-Monro's stochastic approximation procedure (SAP; Tieleman and Hinton 2009; Desjardins et al. 2010; Salakhutdinov 2010). These studies focus on the interactions between the mixing of Markov chains and distribution updates. However, we found that, since the model changes at each iteration, the Markov chain is actually not subject to an invariant distribution, the concept of the mixing of Markov chains is fairly subtle and difficult to capture based on SAP.

Alternatively, Asuncion et al. (2010) exposed that PCD can be interpreted as a sequential Monte Carlo procedure by extending MCMCML to a particle filtered version. To have an quick overview of sequential Monte Carlo More and how it is related to learning UGMs, we first go back to Markov chain Monte Carlo maximum likelihood (MCMCML; section 2.1) and examine it in an extreme case. When the proposal distribution in MCMCML is reset at every iteration as the previously updated estimation, *i.e.* $L = 1$ in Algorithm 1 and the proposal distribution is left as $p(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ at the tth iteration, the weights will be computed as $w^{(s)} = \exp(\boldsymbol{\theta_t} - \boldsymbol{\theta_{t-1}})^\top \phi(\bar{\mathbf{x}}^{(s)})$. Since the parameters $\boldsymbol{\theta}$ do not change very much along iterations, it is not necessary to generate particles[1] from proposal distributions at each iteration. Instead, a set of particles are initially generated and reweighted sequentially for approximating the negative phase. However, if the gap between two successive $\boldsymbol{\theta}$ is relatively large, particles will degenerate. Usually, the effective sampling size (ESS) can be computed to measure the degeneracy of particles, so if ESS is smaller than a pre-defined threshold, resampling and MCMC transition are necessary to recover from it. The description above notably leads to *particle filtered MCMCML* (Asuncion et al., 2010), which greatly outperforms MCMCML with small amount of extra computation.

More interestingly, it was pointed out that PCD also fits the above sequential Monte Carlo procedure (*i.e.* importance reweighting + resampling + MCMC transition) with uniform weighting for all particles and Gibbs sampling as MCMC transition. Here we extend this analogy further to general Robbins-Monro's SAP, into which tempered transition and parallel tempering are also categorized, and write out a uniform interpretation framework of all learning algorithms from SMC perspective (see Algorithm 2). Note that all particle weights are uniformly assigned; resampling has no effect and can be ignored. In addition, the MCMC transition step is forced to take place at every iteration, believing that the particle set is always degenerated.

It is also worth noting that when we are applying algorithms in Algorithm 2, we are not interested in particles from any individual target distribution (which is usually the purpose of SMC).

1. From now on, we use "particles" to fit SMC terminology, it is equivalent to "samples" unless mentioned otherwise.

Algorithm 2 Interpreting Learning as SMC

Input: training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate η

Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0), t \leftarrow 0$
Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
while ! stop criterion **do**
  // `importance reweighting`
  Assign $w^{(s)} \leftarrow \frac{1}{S}, \forall s \in S$
  // `resampling is ignored because it has no effect`
  // `MCMC transition`
  switch (algorithmic choice)
  case CD**:**
    generate a brand new particle set $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling from $\mathcal{D}$
  case PCD**:**
    evolve particle set $\{\bar{\mathbf{x}}_{t}^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling
  case Tempered Transition**:**
    evolve particle set $\{\bar{\mathbf{x}}_{t}^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with tempered transition
  case Parallel Tempering**:**
    evolve particle set $\{\bar{\mathbf{x}}_{t}^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with parallel tempering
  end switch
  // `update distribution`
  Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
  $t \leftarrow t + 1$
end while

Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$

Instead, we want to obtain particles faithfully sampled from all sequence distributions. It can be easily imagined that one badly sampled particle set at tth iteration will lead to a biased incremental update $\Delta\boldsymbol{\theta}_t$. Consequently, the learning will go to a wrong direction even though the later sampling is perfectly good. In other words, we are considering all sequentially updated distributions $p(\mathbf{x}; \boldsymbol{\theta}_t)$ as our target distributions.

In practice, the performance of SMC highly depends on the construction of sequential distributions. In our learning case, sequential distributions are learned by iterative updates, therefore, learning and sampling are somehow entangled. As we mentioned earlier, particles will degenerate when the gap between successive sequential distributions is large. Checking ESS followed by resampling and MCMC transition can help to some extent. However, in many practical cases where real-world distributions are extremely complex, more consideration on MCMC transition is due. In our case of learning UGMs, the gap can be intuitively understood as the product of *learning rate* η and *model complexity* $\mathcal{O}(\boldsymbol{\theta})$. Therefore, we believe that learning rate and model complexity[2] are two key factors to challenge learning algorithms.

2. Here we consider the multimodality of a distribution as its complexity, *i.e.* smooth distributions are less complex than multi-modal distributions.

Within this SMC-based interpretation, we can see that four algorithms differ from each other at MCMC transitions, which is an important component in SMC (Schäfer and Chopin, 2013). In PCD, a one-step Gibbs sampler is used as MCMC transition. As for tempered transition, a Metropolis-Hasting (MH) move based on forward-and-backward sequence of Gibbs samplers of different temperatures is employed. Likewise, parallel tempering also uses a MH move. This move is generated by swapping particles native to the distributions of different temperatures. By contrast, in CD, a brand new particle set is generated by running one-step Gibbs sampling from training data, which is actually not a MCMC transition. When the learning rate is small and two successive distributions are smooth (*e.g.* at the early stage of learning or when the model is of low dimension), PCD, tempered transition and parallel tempering can traverse sampling space sufficiently well. However, when the learning rate is large or two sequential distributions exhibt multiple modes (*e.g.* at the late stage of learning or when the model is high-dimensional), highly correlated particles from the one-step Gibbs sampler's local movement cannot go through the gap between two distributions. Tempered transition and parallel tempering, instead, are more robust to the large gap since it moves closer to the later distribution by making use of many globally-tempered intermediary distributions. The worst case is CD, which always samples particles within the vicinity of training data $\mathcal{D}$. So it will eventually drop $\mathcal{D}$ down into an energy well surrounded by barriers set up by their proximities.

5. Persistent Sequential Monte Carlo

It was explained that learning UGMs can be interpreted as a SMC procedure. Here we propose to apply this rationale further in learning UGMs with a deeper construction of sequential distributions. The basic idea is very simple; given particles from $p(\mathbf{x};\boldsymbol{\theta}_t)$, many sub-sequential distributions are inserted to construct a sub-SMC for obtaining particles from $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Inspired by global tempering used in parallel tempering and tempered transition, we build sub-sequential distributions $\{p_h(\mathbf{x};\boldsymbol{\theta}_{t+1})\}_{h=0}^{H}$ between $p(\mathbf{x};\boldsymbol{\theta}_t)$ and $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$ as follows:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_t)^{1-\beta_h} p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{13}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. In this way, the length of the distribution sequence will be extended in SMC. In addition, obviously, $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_t)$ while $p_0(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Therefore, the whole learning can be considered to be based on a long, persistent sequence of distributions, and therefore the proposed algorithm is referred to as *persistent SMC* (PSMC). An alternative understanding of PSMC can be based on using standard SMC for sampling $p(\mathbf{x};\boldsymbol{\theta}_t)$ at each iteration. In standard SMC case, the sub-sequential distributions are:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{14}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. The schematic figures of standard SMC and PSMC are presented in Figure 1 where we can see a prominent difference between them, the continuity from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ to $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Intuitively, PSMC can be seen as a linked version of SMC by connecting $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ and $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. In addition, in our implementation of PSMC, to ensure adequate exploration, only half of the particles from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ are preserved to the next iteration; the other half particles are randomly initialized with a uniform distribution $\mathcal{U}^D$ (Figure 1(*b*)).

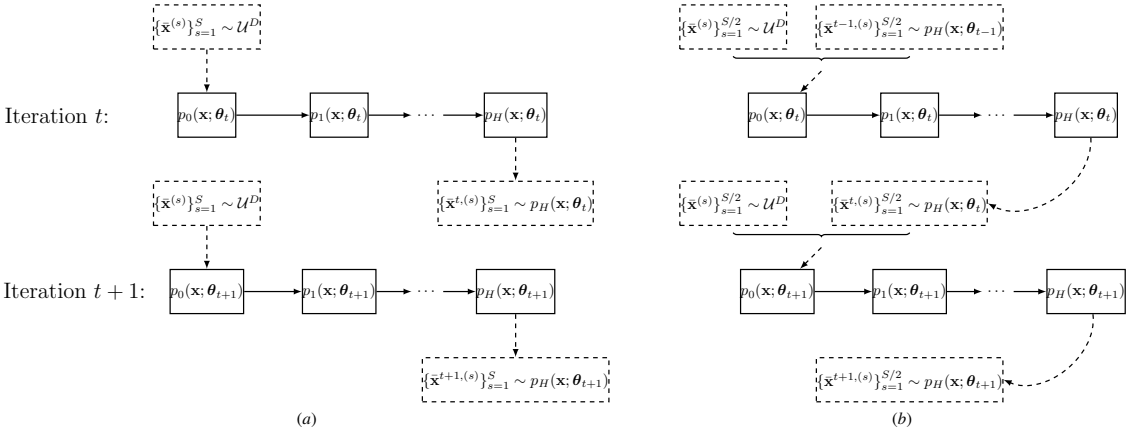


Figure 1: The schematic figures of (a) standard sequential Monte Carlo and (b) persistent sequential Monte Carlo for learning UGMs. Solid boxes denote sequential distributions and solid arrows represent the move (resampling and MCMC transition) between successive distributions. Dashed boxes are particle sets and dashed arrows mean feeding particles into a SMC or sampling particles out of a distribution.

One issue arising in PSMC is the number of β_h, *i.e.* H, which is also a problem in parallel tempering and tempered transition[3]. Here, we employed the bidirectional searching method (Jasra et al., 2011). When we construct sub-sequential distributions as (13), the importance weighting for each particle is

$$\begin{aligned} w^{(s)} &= \frac{p_h(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})}{p_{h-1}(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})} \\ &= \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_t)\right)^{\Delta\beta_h} \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})\right)^{-\Delta\beta_h} \end{aligned} \tag{15}$$

where $\Delta\beta_h$ is the step length from β_{h-1} to β_h, *i.e.* $\Delta\beta_h = \beta_h - \beta_{h-1}$. We can also compute the ESS of a particle set as (Kong et al., 1994)

$$\sigma = \frac{(\sum_{s=1}^{S} w^{(s)})^2}{S\sum_{s=1}^{S} w^{(s)2}} \in \left[\frac{1}{S}, 1\right] \tag{16}$$

Based on (15) and (16), we can see that, when a particle set is given, ESS σ is actually a function of $\Delta\beta_h$. Therefore, assuming that we set the threshold of ESS as σ^*, we can then find the biggest $\Delta\beta_h$ by using bidirectional search (see Algorithm 3) . Usually a small particle set is used in learning (mini-patch scheme), so it will be quick to compute ESS. Therefore, with a small amount of extra computation, the gap between two successive βs and the length of the distribution sequence in PSMC can be actively determined, which is a great advantage over the manual tunning in parallel tempering and tempered transition. By integrating all pieces together, we can write out a pseudo code of PSMC as in Algorithm 4.

3. Usually, there is no systematic way to determine the number of β_h in parallel tempering and tempered transition, and it is selected empirically.

Algorithm 3 Finding $\Delta\beta_h$

Input: a particle set $\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S}$, β_{h-1}

$l \leftarrow 0$, $u \leftarrow \beta_{h-1}$, $\alpha \leftarrow 0.05$
while $|u - l| \geq 0.005$ and $l \leq \beta_{h-1}$ **do**
  compute ESS σ by replacing $\Delta\beta_h$ with $-\alpha$ according to (16)
  if $\sigma < \sigma*$ **then**
    $u \leftarrow \alpha$, $\alpha \leftarrow (l + \alpha)/2$
  else
    $l \leftarrow \alpha$, $\alpha \leftarrow (\alpha + u)/2$
  end if
end while

Output: Return $\Delta\beta_h = \max(-\alpha, -\beta_{h-1})$

Algorithm 4 Learning with PSMC

Input: a particle set $\{\mathbf{x}^{(m)}\}_{m=1}^{M}$, learning rate η

Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0), t \leftarrow 0$
Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
while ! stop criterion **do**
  $h \leftarrow 0$, $\beta_0 \leftarrow 1$
  while $\beta_h < 1$ **do**
    assign importance weights $\{w^{(s)}\}_{s=1}^{S}$ to particles according to (15)
    resample particles based on $\{w^{(s)}\}_{s=1}^{S}$
    compute the step length $\Delta\beta_h$ according to Algorithm 3
    $\beta_{h+1} = \beta_h + \delta\beta$
    $h \leftarrow h + 1$
  end while
  Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
  $t \leftarrow t + 1$
end while

Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$

6. Experiments

In our experiments, PCD, parallel tempering (PT), tempered transition (TT), standard SMC and PSCM were empirically compared on 2 different discrete UGMs, *i.e.* fully visible Boltzmann machines (VBMs) and restricted Boltzmann machines (RBMs). As we analyzed in section 4, large learning rate and high model complexity are two main challenges for learning UGMs. Therefore, two experiments were constructed to test the robustness of algorithms to different learning rates and model complexities separately. On one hand, one VBM was constructed with small size and tested with synthetic data. The purpose of the small-scale VBM is to reduce the effect of model complexity. In addition, the exact log-likelihood can be computed in this model. On the other hand, two RMBs were used in our second experiment, one is medium-scale and the other is large-scale.

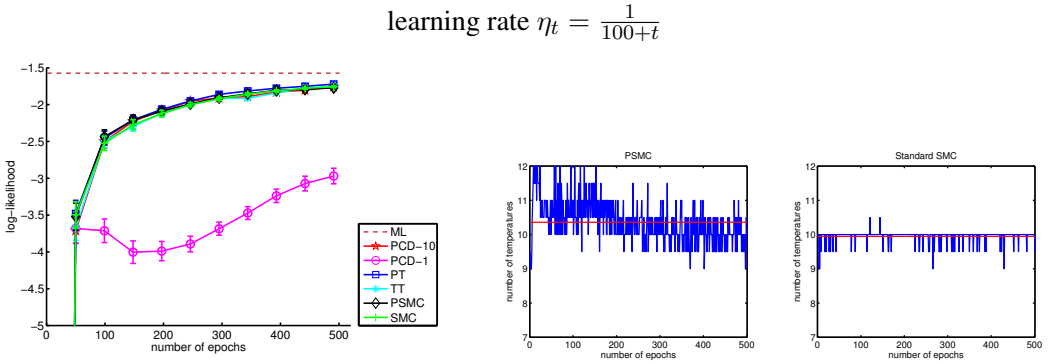


Figure 2: The performance of algorithms with the first learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

They were applied on a real-world database MNIST[4]. In this experiment, the learning rate was set to be small to avoid its effect. In both experiments, mini-patch of 200 data instances were used. When PSMC and SMC were run, $\sigma^* = 0.9$ was used as the threshold of ESS. We recorded the number of βs at each iteration in PSMC, and computed the average value H. In order to ensure the fairness of the comparison, we offset the computation of different algorithms. In PT, H βs were uniformly assigned between 0 and 1. In TT, similarly, H βs were uniformly distributed in the range $[0.9, 1]$[5]. Two PCD algorithms were implemented, one is with one-step Gibbs sampling (PCD-1) and the other is with H-step Gibbs sampling (PCD-H). In the second experiment, the computation of log-likelihoods is intractable, so here we employed an annealing importance sampling (AIS)-based estimation proposed by Salakhutdinov and Murray (2008). All methods were run on the same hardware and experimental conditions unless otherwise mentioned.

6.1. Experiments with Different Learning Rates

A Boltzmann machine is a kind of stochastic recurrent neural network with fully connected variables. Each variable takes binary value $\mathbf{x} \in \{-1, +1\}^D$. Using the energy representation (2), parameters $\boldsymbol{\theta}$ correspond to $\{\mathbf{W} \in \mathbb{R}^{D\times D}, \mathbf{b} \in \mathbb{R}^{D\times 1}\}$ and $\phi(\mathbf{x}) = \{\mathbf{x}\mathbf{x}^\top, \mathbf{x}\}$. Here we used a fully visible Boltzmann machine (VBM), and computed the log-likelihood to quantify performances. In this experiment, a small-size VBM with only 10 variables is used to avoid the effect of model complexity. For simplicity, $W_{ij}{}_{i,j\in[1,10]}$ were randomly generated from an identical distribution $\mathcal{N}(0, 1)$, and 200 training data instances were sampled. Here we tested all learning algorithms with 3 different learning rate schemes: (1) $\eta_t = \frac{1}{100+t}$, (2) $\eta_t = \frac{1}{20+0.5\times t}$, (3) $\eta_t = \frac{1}{10+0.1\times t}$. The learning rates in the three schemes were at different magnitude levels. The first one is smallest, the second is intermediate and the last one is relative large. For the first scheme, 500 epochs were run, and the log-likelihood *vs.* number of epochs plots of different learning algorithms are presented in

4. http://yann.lecun.com/exdb/mnist/index.html
5. In our experiment, we used a TT similar to that used by Salakhutdinov (2010) by alternating between one Gibbs sampling and one tempered transition.

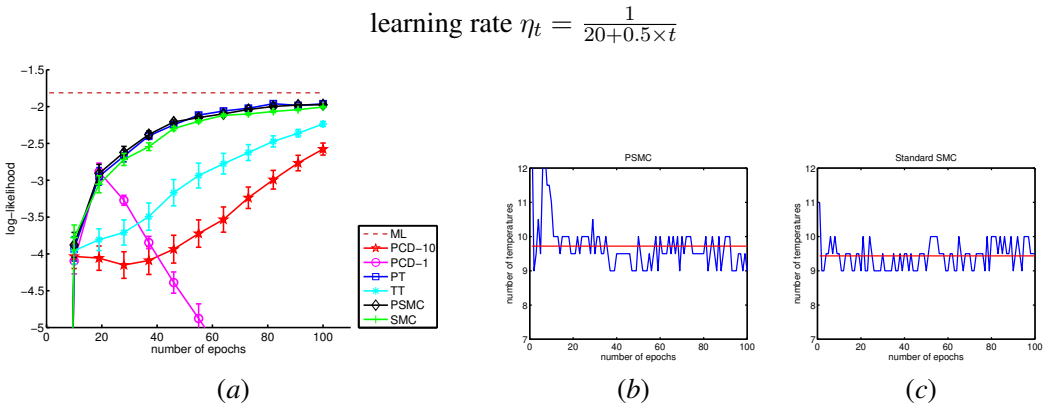


Figure 3: The performance of algorithms with the second learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

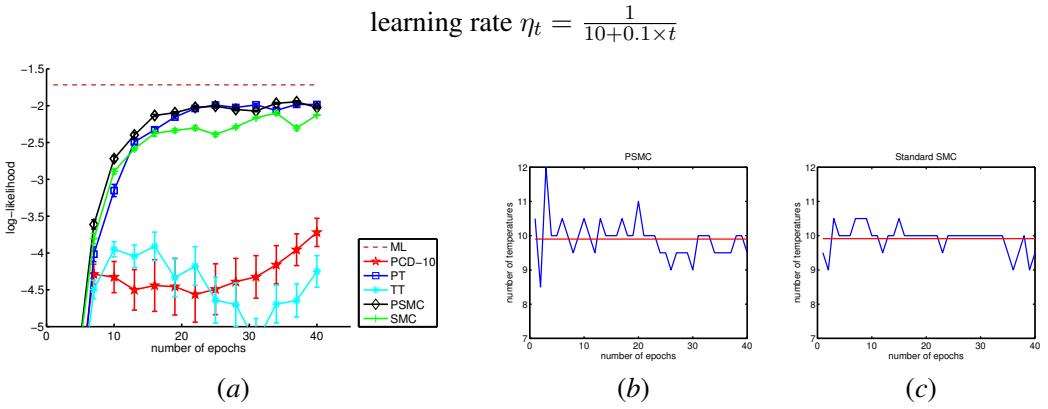


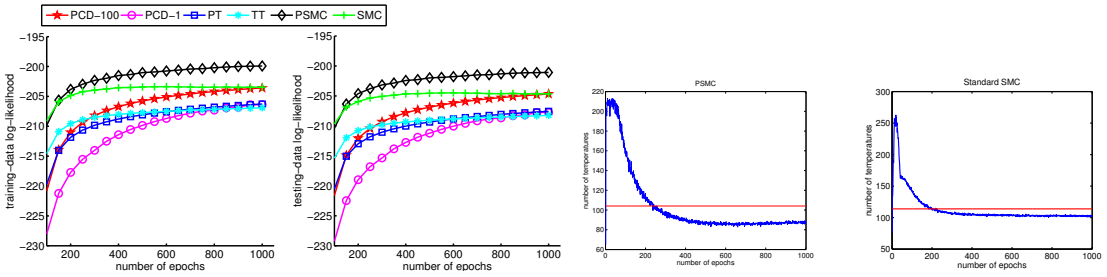
Figure 4: The performance of algorithms with the third learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

Figure 2(*a*). The number of βs in PSMC and SMC are also plotted in Figures 2(*b*) and 2(*c*) respectively. We can see that the mean value H in PSMC is around 10, which is slightly higher than the one in SMC. For the second and third learning rate schemes, we ran 100 and 40 epochs respectively. All algorithms' performances are shown in Figure 3(*a*) and 4(*a*). We found that the number of βs in PSMC and SMC are very similar to those of the first scheme (Figures 3(*b*), 3(*c*), 4(*b*) and 4(*c*)). For all three schemes, 5 trials were run with different initial parameters, and the results are presented with mean values (curves) and standard deviations (error bars). In addition, maximum likelihood (ML) solutions were obtained by computing exact gradients (3). For better quantitative comparison, the average log-likelihoods based on the parameters learned from six algorithms and three learn-

LEARNING UNDIRECTED GRAPHICAL MODELS USING PERSISTENT SEQUENTIAL MONTE CARLO

Models		(Avg.) Log-Likelihoods					
(Size)	Learning rate schemes	PCD-1	PCD-H	PT	TT	SMC	PSMC
VBM	$\eta_t = \frac{1}{100+t}$	-1.693	-1.691	$\mathbf{-1.689}$	-1.692	-1.692	-1.691
(15)	$\eta_t = \frac{1}{20+0.5\times t}$	-7.046	-2.612	-1.995	-2.227	-2.069	$\mathbf{-1.891}$
	$\eta_t = \frac{1}{10+0.1\times t}$	-25.179	-3.714	-2.118	-4.329	-2.224	$\mathbf{-1.976}$
	MNIST						
RBM	training data	-206.3846	-203.5884	206.2819	-206.9033	-203.3672	$\mathbf{-199.9089}$
(784×10)	testing data	-207.7464	-204.6717	206.2819	-208.2452	-204.4852	$\mathbf{-201.0794}$
RBM	training data	-176.3767	-173.0064	-165.2149	-170.9312	-678.6464	$\mathbf{-161.6231}$
(784×500)	testing data	-177.0584	-173.4998	-166.1645	-171.6008	-678.7835	$\mathbf{-162.1705}$

Table 1: Comparison of Avg.log-likelihoods with parameters learned from different learning algorithms and conditions.



(*a*) (*b*) (*c*)

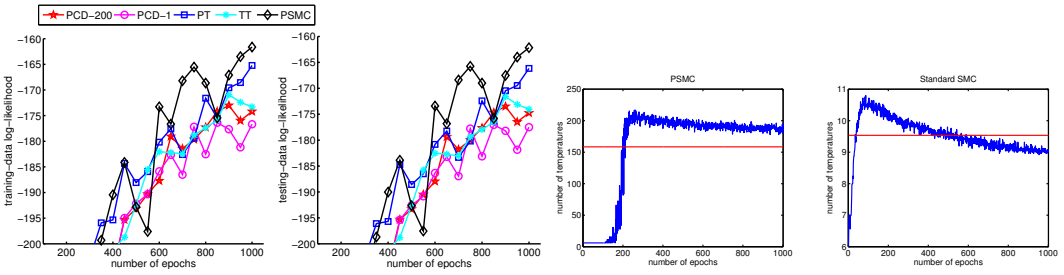
Figure 5: The performance of algorithms on the medium-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

ing rate schemes are listed in the upper part of Table 1. The results of the first experiment can be summarized as follows:

1. When the learning rate was small, PT, TT, SMC, PSMC and PCD-10 worked similarly well, outperforming PCD-1 by a large margin.

2. When the learning rate was intermediate, PT and PSMC still worked successfully, which were closely followed by SMC. TT and PCD-10 deteriorated, while PCD-1 absolutely failed.

3. When the learning rate went to relatively large, the fluctuation patterns were obvious in all algorithms. Meanwhile, the performance gaps between PSMC and other algorithms was larger. In particular, TT and PCD-10 deteriorated very much. Since PCD-1 failed even worse in this case, its results are not plotted in Figure 4(*a*).



(*a*) (*b*) (*c*)

Figure 6: The performance of algorithms on the large-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

6.2. Experiments with Models of Different Complexities

In our second experiment, we used the popular restricted Boltzmann machine to model handwritten digit images (with the MNIST database). RBM is a bipartite Markov network consisting of a visible layer and a hidden layer, it is a "restricted" version of Boltzmann machine with only interconnections between the hidden layer and the visible layer. Assuming that the input data are binary and N_v-dimensional, each data point is fed into the N_v units of the visible layer $\mathbf{v}$, and N_h units in hidden layer $\mathbf{h}$ are also stochastically binary variables (latent features). Usually, $\{0, 1\}$ is used to represent binary values in RBMs to indicate the activations of units.The energy function $E(\mathbf{v}, \mathbf{h})$ is defined as $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{h}^\top \mathbf{b} - \mathbf{v}^\top \mathbf{c}$, where $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}$, $\mathbf{b} \in \mathbb{R}^{N_v \times 1}$ and $\mathbf{c} \in \mathbb{R}^{N_h \times 1}$. Although there are hidden variables in the energy function, the gradient of likelihood function can be written out in a form similar to (3) (Hinton, 2002). Images in the MNIST database are 28×28 handwritten digits, *i.e.* N_v=784. To avoid the effect of learning rate, in this experiment, a small learning rate scheme $\eta_t = \frac{1}{100+t}$ was used and 1000 epochs were run in all learning algorithms. Two RBMs were constructed for testing the robustness of learning algorithms to model complexity, one medium-scale with 10 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784\times 10}$), the other large-scale with 500 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784\times 500}$)[6]. Similarly to the first experiment, we first ran PSMC and SMC, and recorded the number of triggered βs at each iteration and their mean values (Figure 5(*b*), 5(*c*), 6(*b*) and 6(*c*)). For the medium-scale model, the number of βs in PSMC and SMC are similar (around 100). However, for the large-scale model, the mean value of $|\{\beta_0, \beta_1, \cdots\}|$ is 9.6 in SMC while 159 in PSMC. The reason for this dramatic change in SMC is that all 200 particles initialized from the uniform distribution were depleted when the distribution gets extremely complex. For other learning algorithms, H was set 100 and 200 in the medium- and large-scale cases, respectively. Since there are 60000 training images and 10000 testing images in the MNIST database, we plotted both training-data log-likelihoods and testing-data log-likelihoods as learning progressed (see Figure 5(*a*) and 6(*a*)). More detailed quantitative comparison can be seen in the lower part of Table 1. Similarly, we conclude the results of the second experiments as follows:

6. Since a small-scale model was already tested in the first experiment, we did not repeat it here.

1. When the scale of RBM was medium, PSMC worked best by reaching the highest training-data and testing-data log-likelihoods. SMC and PCD-100 arrived the second highest log-likelihoods, although SMC converged much faster than PCD-100. PT, TT and PCD-1 led to the lowest log-likelihoods although PT and TT raised log-likelihoods more quickly than PCD-1.

2. When the scale of RBM was large, all algorithms displayed fluctuation patterns. Meanwhile, PSMC still worked better than others by obtaining the highest log-likelihoods. PT ranked second, and TT ranked third, which was slightly better than PCD-200. PCD-1 ranked last. SMC failed in learning the large-scale RBM, so its results are not presented in Figure 6(*a*).

7. Conclusion

A SMC interpretation framework of learning UGMs was presented, within which two main challenges of the learning task were disclosed as well. Then, a persistent SMC (PSMC) learning algorithm was developed by applying SMC more deeply in learning. According to our experimental results, the proposed PSMC algorithm demonstrates promising stability and robustness in various challenging circumstances with comparison to state-of-the-art methods. Meanwhile, there still exist much room for improvement of PSMC, *e.g.* using adaptive MCMC transition (Schäfer and Chopin, 2013; Jasra et al., 2011), which suggests many possible directions for future work. Besides, although PSMC is expected to approach the maximum likelihood solution in learning UGMs, sometimes maximizing the posterior function is more desirable (*e.g.* when the prior is available), so it is also interesting to extend PSMC for maximum a posteriori learning.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

Arthur U. Asuncion, Qiang Liu, Alexander T. Ihler, and Padhraic Smyth. Particle filtered MCMC-MLE with connections to contrastive divergence. In *ICML*, 2010.

Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On Contrastive Divergence Learning. In *AISTATS*, 2005.

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *AISTATS*, 2010.

Charles J. Geyer. *Markov Chain Monte Carlo Maximum Likelihood.* Compuuting Science and Statistics:Proceedings of the 23rd Symposium on the Interface, 1991.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris. statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, pages 1548–7660, 2007.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Ajay Jasra, David A. Stephens, Arnaud Doucet, and Theodoros Tsagaris. Inference for lévy-driven stochastic volatility models via adaptive sequential monte carlo. *Scandinavian Journal of Statistics*, 38:1–22, 2011.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Augustine Kong, Jun S. Liu, and Wing H. Wong. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association*, 89(425):278–288, March 1994.

Radford Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1994.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Ann.Math.Stat.*, 22:400–407, 1951.

Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In *NIPS*, 2010.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *ICML*, 2008.

Christian Schäfer and Nicolas Chopin. Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23(2):163–184, 2013.

T. Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *ICML*, pages 1064–1071, 2008.

T. Tieleman and G.E. Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. In *ICML*, pages 1033–1040. ACM New York, NY, USA, 2009.

3.2.1 Training Conditional Random Fields for Image Annotation and Image Segmentation

In this section we present some evaluations and comparison of different learning algorithms on two practical tasks: *multi-label learning* and *image segmentation*. Different from previous experiments where generative models were learned, here we trained discriminative models. Therefore, two conditional random fields were employed. Generally speaking, let us denote $\mathbf{x}$ as input and $\mathbf{y} \in \mathcal{Y}$ as output, and our target is to learn an UGM:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}^\top \phi(\mathbf{y}, \mathbf{x}))}{\mathbf{Z}} \tag{3.10}$$

where the partition function $\mathbf{Z}$ is

$$\mathbf{Z} = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{y}, \mathbf{x})) \tag{3.11}$$

where $\phi(\mathbf{y}, \mathbf{x})$ is defined based on task-oriented dependency structure. Note that the partition function $\mathbf{Z}$ is computed by marginalizing out only $\mathbf{y}$ because our interest is a conditional distribution. Six algorithms were implemented: PCD-H, PCD-1, PT, TT, SMC and PSMC. Similar setups were used for all algorithms as the previous section. Learning rate $\eta_t = \frac{1}{10+0.1*t}$ was used and 100 iterations were run. For each input $\mathbf{x}$, the size of particle set $\{\hat{\mathbf{y}}^{(s)}\}$ is 200. Similar to other supervised learning schemes, a regularization $\frac{1}{2}||\boldsymbol{\theta}||^2$ was added and a trade-off parameters was tunned via k folder cross-validation ($k = 4$).

It is worth mentioning that better results can be expected in both experiments by running more iterations, using better learning rates or exploiting feature engineering. However, our purpose here is to compare different learning algorithms under the some conditional instead of defeat state-of-the-art results in multi-label learning study and image segmentation study respectively. Therefore, we saved our effort in tunning algorithms and constructing sophisticated features.

3.2.1.1 Multi-Label Learning

In multi-label learning, inter-label dependency is rather critical. Assume that input $\mathbf{x} \in \mathbb{R}^d$ and there are L labels (*i.e.* $\mathbf{y} \in \{-1, +1\}^L$), here we modeled all pairwise dependencies among L labels, and therefore the constructed conditional random field is:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{y}^\top \mathbf{W}_E \mathbf{y} + \mathbf{y}^\top \mathbf{W}_v \mathbf{x})}{\mathbf{Z}} \tag{3.12}$$

	Precision(%)	Recall(%)	F1(%)
PCD-1	57.7	59.3	58.5
PCD-5	70.3	72.6	71.4
TT	70.0	67.5	68.7
PT	**72.2**	77.1	74.6
SMC	71.7	75.1	73.4
PSMC	71.9	**78.5**	**75.1**

TABLE 3.1: A comparison of six learning algorithms on multi-label learning.

where $\mathbf{W}_E \in \mathbb{R}^{L\times L}$ captures pairwise dependencies among L labels while $\mathbf{W}_v \in \mathbb{R}^{L\times d}$ reflects the dependencies between input $\mathbf{x}$ and all individual labels. In the test phase, with learned $\mathbf{W}_E$ and $\mathbf{W}_v$, for a test input $\mathbf{x}^\dagger$, we predict the corresponding $\mathbf{y}^\dagger$ with 100 rounds of gibbs sampling based on (3.12).

In our experiment, we used popular *scene* database (Boutell et al., 2004), where scene images are associated with a few semantic labels. In the database, there are 1121 training instances and 1196 test instances. In total there are 6 labels ($L = 6$) and a 294 dimensional feature were extracted from each image ($\mathbf{x} \in \mathbb{R}^{294}$). Readers are referred to Boutell et al. (2004) for more details about the database and feature extraction.

We evaluated the performance of multi-label learning using *precision* (P), *recall* (R), and the *F1* measure (F). For each label, the precision is computed as the ratio of the number of images assigned the label correctly over the total number of images predicted to have the label, while the recall is the number of images assigned the label correctly divided by the number of images that truly have the label. Then precision and recall are averaged across all labels. Finally, the F1 measure is calculated as $F = 2\frac{P\times R}{P+R}$. The results of all six algorithms are presented in Table **??**. The average number temperatures in PSMC is around 5, so PCD-5 was implemented. Also 5 temperatures were use in PT and TT. We can see that PSMC yields the best F1 measure 75.1, followed by PT and SMC with 74.6 and 73.4 respectively. The results of PCD-5 and TT are relative worse, while PCD-1 is the worst.

3.2.1.2 Image Segmentation

Image segmentation essentially is a task to predict the semantic label of all image pixels or blocks. Inter-label dependences within neighbourhood are usually exploited in image segmentation. For instance, by dividing an image into equal size and non-overlapping blocks, the label of a block does not only depend on the appearance of the block, but also the labels of its neighbouring blocks. For simplicity, here we only consider binary labels. In addition, we assume that blocks and inter-label dependencies are position

irrelevant. Therefore, a conditional random filed can be constructed as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{u,v\in E} y_u W_e y_v + \sum_{v\in V} y_v \mathbf{w}_v^\top \mathbf{x}_v)}{\mathbf{Z}} \tag{3.13}$$

where $y_v \in \{-1, +1\}$, E denotes the set of all edges connecting neighbouring blocks, $W_e \in \mathbb{R}$ encodes the dependency between neighbouring labels, V denotes the set of all block's labels and $\mathbf{w}_v \in \mathbb{R}^{d\times 1}$ encodes the dependency between block label and its appearance which is represented by a d-dimensional feature $\mathbf{x}_v \in \mathbb{R}^d$. Similar to the multi-label experiment, desired labels are predicted via 100 round gibbs sampling in test phase.

In our experiment, we used the binary segmentation database from Kumar and Hebert (2003b), where each image is divided into non-overlapping blocks of size 16×16 and each block is annotated with either "man-made structure" (MS) or "nature structure" (NS). Overall, there are 108 training images and 129 test images. The training set contains 3004 MS blocks and 36269 NS blocks, while the test set contains 6372 MS blocks and 43164 NS blocks. For each block, its appearance is represented by a 3-dimensional features which includes the mean of gradient magnitude, the 'spikeness' of the count-weighted histogram of gradient orientations, the angle between the most frequent orientation and the second most frequent orientation. The feature was designed to fit this specific application. More explanation of the database and feature design can be found in Kumar and Hebert (2003b).

We found that the average number of temperatures in PSMC is 20, therefore PCD-20 was run and 20 temperatures were used in TT and PT. We quantify the segmentation performance of six algorithms with confusion matrices, which are presented in Figure 3.1. We can see that PSMC outperforms all others (by checking the diagonal entries of confusion matrices). For qualitative comparison, an example image and corresponding segmentations are shown in Figure 3.2. It can be seen that the segmentation by PSMC is closer to the ground truth compared to others.

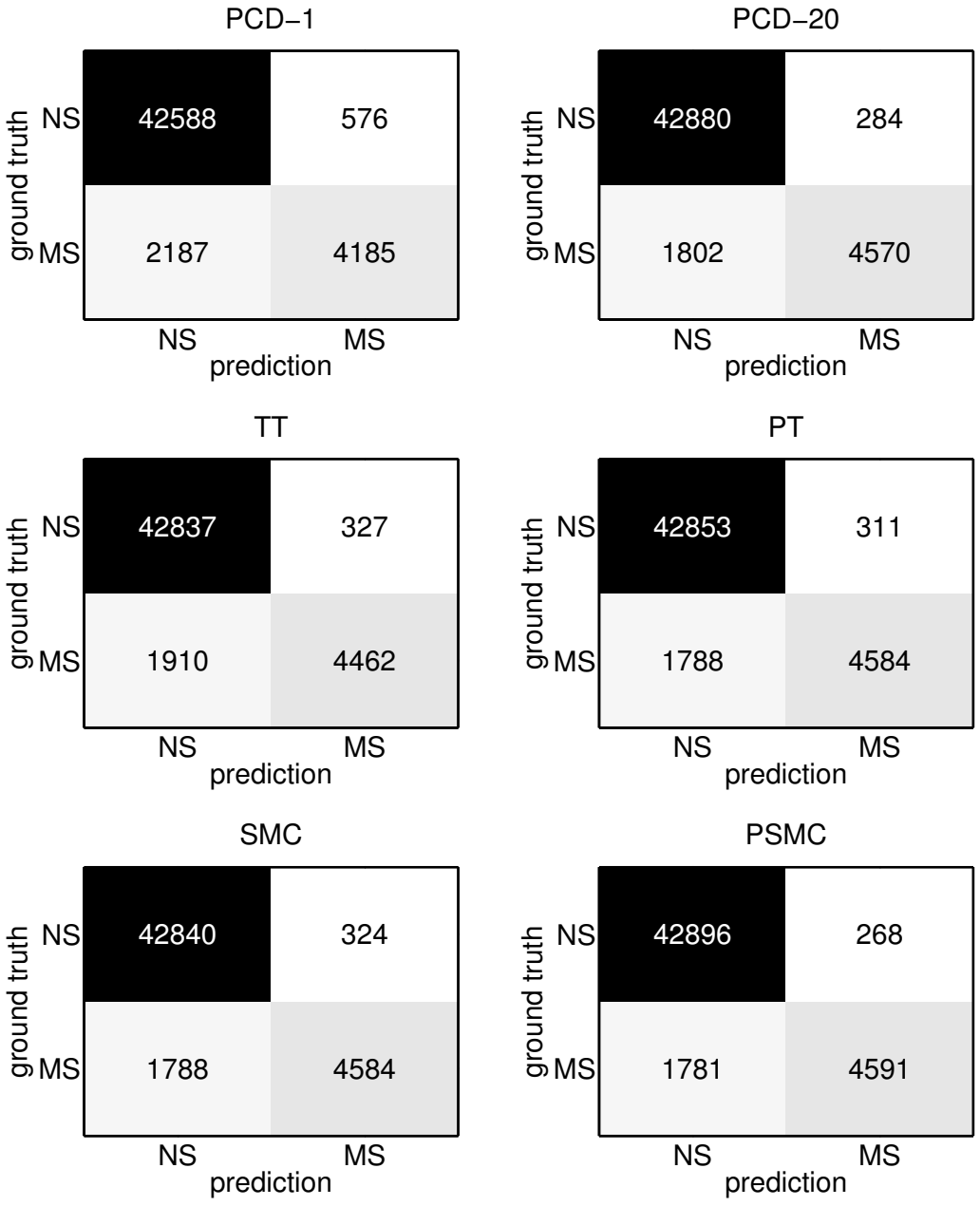


FIGURE 3.1: Confusion matrices of binary segmentation by six algorithms.

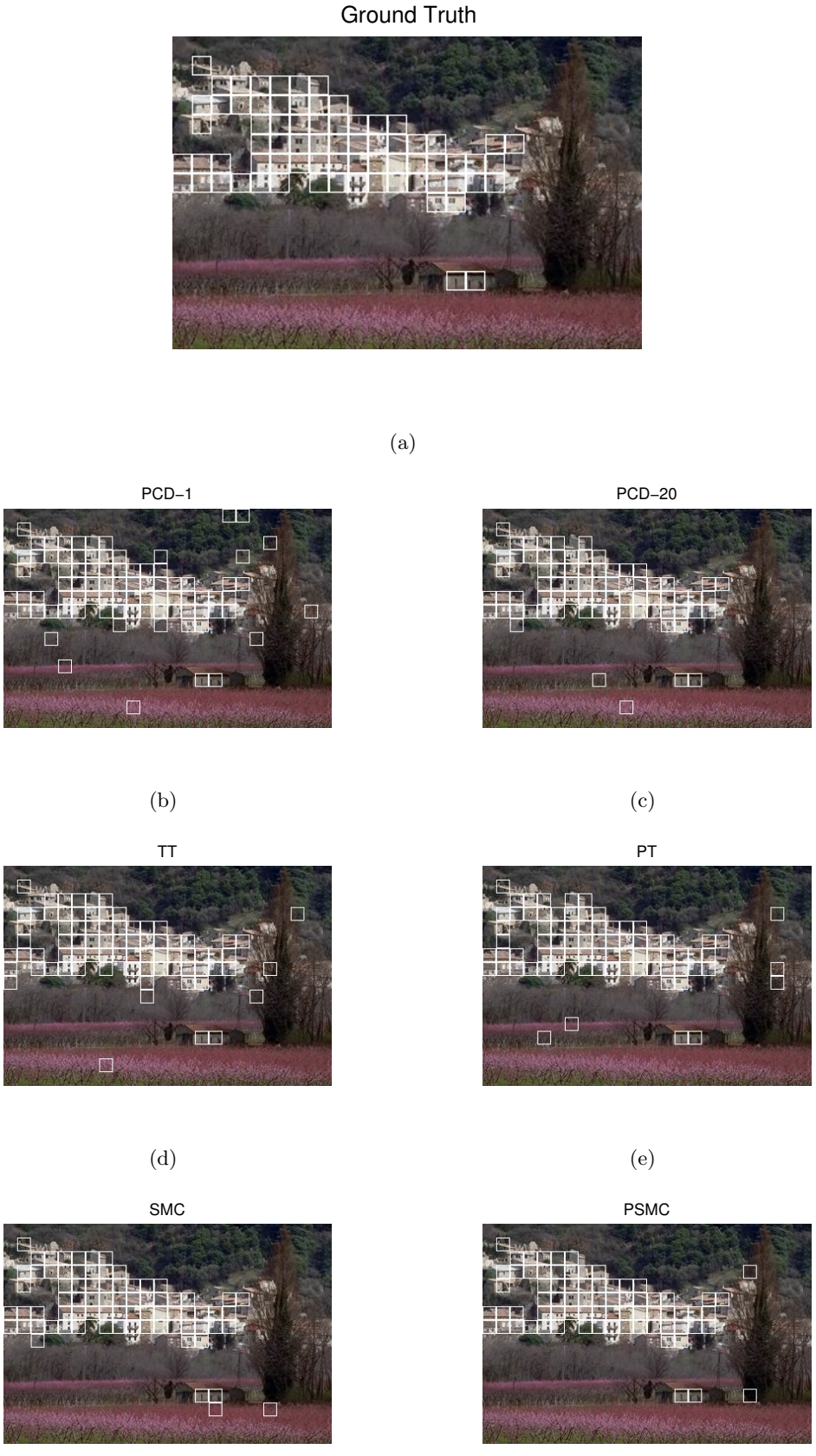


FIGURE 3.2: An example image and corresponding segmentations by six algorithms.

Chapter 4

Kernel-Based Structural Output Learning

"Nothing is more practical than a good theory."

Vladimir Vapnik

4.1 Joint SVM

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

4.1.1 Structural SVM for Multi-Label Learning

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper.

Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

4.1.2 Joint SVM: Output Kernel Learning and Regularization

I. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Implicit Learning of Simpler Output Kernels for Multi-Lable Prediction*, NIPS workshop on Representation and Learning Methods for Complex Outputs (NIPS-RLCO2014).

III. **Hanchen Xiong**, Sandor Szedmak, Justus Piater. *Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels*, Neurocomputing Journal (Accepted).

Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels

Hanchen Xiong[1], Sandor Szedmak, Justus Piater

Institute of Computer Science, University of Innsbruck, Technikerstr.21a, A-6020 Innsbruck, Austria

Abstract

This paper studies how joint training of multiple support vector machines (SVMs) can improve the effectiveness and efficiency of automatic image annotation. We cast image annotation as an output-related multi-task learning framework, with the prediction of each tag's presence as one individual task. Evidently, these tasks are related via dependencies between tags. The proposed joint learning framework, which we call *joint SVM*, is superior to other related models in its impressive and flexible mechanisms in exploiting the dependencies between tags: first, a linear output kernel can be implicitly learned when we train a joint SVM; or, a pre-designed kernel can be explicitly applied by users when prior knowledge is available. Also, a practical merit of joint SVM is that it shares the same computational complexity as one single conventional SVM, although multiple tasks are solved simultaneously. Although derived from the perspective of multi-task learning, the proposed joint SVM is highly related to structured-output learning techniques, *e.g.* max-margin regression [1], structural SVM [2]. According to our empirical results on several image-annotation benchmark databases, our joint training strategy of SVMs can yield substantial improvements, in terms of both accuracy and efficiency, over training them independently. In particular, it compares favorably with many other state-of-the-art algorithms. We also develop a "perceptron-like" online learning scheme for joint SVM to enable it to scale up better to huge data in real-world practice.

Keywords: image annotation, multi-label learning, output kernels, maximum margin
2014 MSC: 00-01, 99-00

1. Introduction

Automatic image annotation is an important yet challenging machine learning task. The importance is based on the fact that the number of images grows increasingly fast on the internet, and most of them have no link to semantic tags (or keywords, labels). Therefore, automatic annotation is of great significance to generate meaningful meta-data for organizing image collections, and in particular, retrieving images from textual queries. The challenges are usually considered from two classical perspectives [3]: first, *semantic-gap*, *i.e.* the gap from low-level image features to textual tags is large and there exist no reliable way to extract dependencies between them; secondly, *absence of correspondence*, *i.e.* for each tag associated with one image, there is no corresponding region annotated, which hinders learning worse. Meanwhile, when considering contemporary image annotation, one more difficulty to bear is big data. The image data on internet is usually presented in large volumes (million or billion level), so the desired learning method should be capable of working on large-scale data with high learning and prediction efficiency. One straight-forward yet naive strategy is to consider each tag's presence as a binary classification problem. Then, multiple binary classifiers, *e.g.* support vector machines (SVMs), can be trained independently for different tags. This method, however, will suffer from high computational complexity in both training and prediction

Email addresses: hanchen.xiong@uibk.ac.at (Hanchen Xiong), sandor.szedmak@uibk.ac.at (Sandor Szedmak), justus.piater@uibk.ac.at (Justus Piater)
[1]Corresponding author

phases when the number of tags is relatively large. In addition, independently learning multiple SVMs is not expected to work well because it ignores the dependencies between the presences of tags [4], which is a phenomenal characteristic of image annotation tasks (*e.g.*, sky and cloud often co-occur).

In this paper, we propose to interpret image annotation as the learning of multiple related tasks. However, different from most existing multi-task learning frameworks [5] in which tasks are related through their *inputs*, our joint learning method focuses on the relation between *outputs*. Our strategy is motivated by two intuitions. First, by connecting multiple SVM classifiers together, the dependencies between their outputs (the presences of tags), presumably, can be more easily encoded. Secondly, if the outputs of multiple SVMs can be merged into a single vector entity, the optimization problem can be established and solved over vectors, greatly reducing the computational complexity. These two objectives, surprisingly, can be easily achieved by summing up the objectives and constraints in different SVMs, plus an appropriately designed kernel on outputs. We refer to the proposed training strategy as *joint SVM*. The key strength of joint SVM is that it can flexibly offer two mechanisms to exploit the dependencies between tags: first, when there is no prior knowledge on the dependencies, a linear output kernel can be implicitly learned when we train a joint SVM; or a pre-designed, prior-oriented, kernel can be applied on outputs when prior knowledge is available (see section 4). In addition, as we will see in section 3, the training of joint SVM shares almost the same computation complexity as a single regular SVM, which is a practical merit when the number of tags is relative large. Interestingly, although derived from the perspective of multi-task learning, the proposed joint SVM highly relates to structured-output learning techniques, such as max-margin regression [1], structural SVM [2] or max-margin Markov network (M^3N) [6]. More connections between them will be exploited in section 5. In addition, to enable joint SVM to scales up to huge data (million or billion level) in real-world practice, we develop a "perceptron-like" online learning algorithm for joint SVM in section 6. In our experiment (section 7), we tested joint SVM on several benchmark image-annotation databases, with comparison against independent SVMs and other results reported in state-of-the-art algorithms. The experimental results show that our joint SVM can gain impressive improvement over training SVMs independently. In particular, it compares favorably with many other state-of-the-art algorithms.

2. Related Work

Prior to our work, there exist many literatures on image annotation in computer vision and machine learning communities [7, 8, 9, 10, 3, 11, 12, 13, 14, 4, 15, 16]. Roughly speaking, all algorithms can be categorized into generative methods or discriminative methods according to how the relevance between image features and textual tags are modeled. On one hand, generative methods, mostly inspired by linguistic translation studies, model the generating or formating procedure of visual features and tags, then tags prediction from a novel image is inferred by leveraging co-occurrence statistics between visual features and tags in training data. Continuous Relevance Model (CRM) [7], Correlation Latent Dirichlet Allocation (CorLDA) [8] and Multiple Bernoulli Relevance Model (MBRM) [9] belong to the generative category. However, one drawback of these method is that usually some statistical assumptions (*e.g.* conditional independence) are imposed on models, which restricts their modeling capabilities. Furthermore, another practical obstacle of most generative methods is the intractability of inference in tag prediction phase, therefore, usually some approximation techniques are applied. On the other hand, discriminative methods directly model the tag-predicting function, out of which TagProp [10], JEC [3] are metric-learning based approaches, rank-SVM [17], LM-K [18] are rank-learning based approaches, M3L [4] and Multi-Label Relationship Learning (MLRL) [16] are maximum-margin based approaches. One notable issue, and also difficulty, in discriminative methods is the dependencies between output tags, of which many state-of-the-art studies [4, 11, 18] have being aware. In several recent studies [10, 3, 11], discriminative methods were reported to displayed empirically superior performance than generative ones on image annotation task. More comparison and analysis on different representative methods can be found in up-to-date reviews [3, 4, 11].

The proposed joint SVM in this paper is a maximum-margin based, discriminative learning framework. Although joint SVM displays strong connections with structured-output learning, the staring point of our work is to improve the annotation performance by exploiting the relationship between individual tag-predictors. A conceptually-related work was concurrently, but independently from us, presented in MLRL

[16], of which the authors explicitly model the relationship as a covariance matrix in matrix-variate normal distribution over individual model parameters. In contrast, in joint SVM, the dependency between different tags are encoded in output kernels. In this sense, our work is also similar to LM-K [18] and M3L [4]. Interestingly, when the output kernel is linear, it is equivalent to the explicit relationship learning in MLRL. Meanwhile, more sophisticated output kernel can flexibly be constructed and utilized in joint SVM, to afford nonlinear, higher-order dependencies, although they are not always of help in practice.

3. Joint Learning of Multiple SVMs

3.1. Support Vector Machines and Input Kernels

In the past two decades, support vector machines (SVMs) have displayed remarkable successes in various application domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [19] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are ith and jth input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Most popularly, a Gaussian (or radial basis function) kernel

$$K_\phi^{Gau}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 / 2\sigma^2\right) \tag{2}$$

is employed because its corresponding RKHS is an unnormalized Gaussian density function:

$$\phi^{Gau}(\mathbf{x}^{(i)}) \propto \mathcal{N}(\tau; \mathbf{x}^{(i)}, \sigma) \tag{3}$$

which is of infinite dimension, and thus greatly improves the representational capability of input data. Another popular kernel function is Polynomial kernel:

$$K_\phi^{Pol}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + c\right)^d \tag{4}$$

In particular, when the degree $d = 1$ and constant term $c = 0$, Polynomial is a simple inner product. Meanwhile, in 2-degree ($d = 2$) Polynomial kernel, corresponding feature map is:

$$\phi^{Pol}(\mathbf{x}) = [x_d^2, \cdots, x_1^2, \sqrt{2}x_d x_{d-1}, \cdots, \sqrt{2}x_2 x_1, \sqrt{2c}x_d, \cdots, \sqrt{2c}x_1, c]^\top \tag{5}$$

Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$ of one binary classification problem, the primal form of training SVM is written

$$\begin{array}{rl} \arg\min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^m \xi^{(i)} \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)})\right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}$ is the normal vector of a linear hyperplane in $\mathcal{H}_\phi$ (here and later we use $\mathcal{H}$ as $dim(\mathcal{H})$ for simplicity when we denote dimensionality) , $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is trade-off parameter de between training error and max-margin regularization. Eq.(6) differs from usual SVM formulation slightly at the absence of a bias term. Here we ignore the bias since it can be absorbed in $\mathbf{w}$ [2]. Actually, eliminating the bias is more critical in predicting multiple dependent labels, check [4] for

[2]When a Polynomial kernel is used, a bias term is already in its corresponding feature map. When a Gaussian kernel is used, an input vector can be augmented with one extra constant.

detailed explanations. The computational advantage of kernels become obvious when the primal form of SVM (Eq.(6)) is reformulated to its dual form by introducing Lagrange multipliers α_i for each constraints:

$$\begin{array}{rl} \arg\min_{\alpha_1,\alpha_2,\ldots,\alpha_m} & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y^{(i)}y^{(j)}K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t.} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{7}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^m \alpha_i y^{(i)}\phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \operatorname{sgn}\left(\mathbf{w}^\top\phi(\hat{\mathbf{x}})\right) = \operatorname{sgn}\left(\sum_{i=1}^m \alpha_i y^{(i)}K_\phi(\mathbf{x}^{(i)},\hat{\mathbf{x}})\right). \tag{8}$$

It can be seen that the kernel function K_ϕ enables the learning of a high-dimensional (even infinite) $\mathbf{w}$ without explicit computation in $\mathcal{H}_\phi$. Eq.(8) shows that the kernel function yields a similarity measurement between two input instances, and the prediction is working as a weighted-sum of all outputs in the training data.

3.2. Joint SVM

The automatic image annotation task seeks to predict the presence of tags given an input image. Assume d-dimensional visual features are extracted from input images and there are T tags in a pre-defined dictionary, then the annotation learning task is to seek a function $f : \mathbf{x} \in \mathbb{R}^d \to \{-1,+1\}^T$. If we consider prediction of each tag's occurrence as a binary classification problem, we can list as many SVMs as the number of tags. Similar to other multi-task learning frameworks [5], we connect the learning tasks of different SVMs by simply summing up their objectives and constraints respectively in the primal form

$$\begin{array}{rl} \min & \frac{1}{2}\sum_{t=1}^T ||\mathbf{w}_t||^2 + C\sum_{t=1}^T\sum_{i=1}^m \xi_t^{(i)} \\ \text{w.r.t.} & \mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi\times 1} \\ \text{s.t.} & \sum_{t=1}^T y_t^{(i)}\left(\mathbf{w}_t^\top\phi(x^{(i)})\right) \ge T - \sum_{t=1}^T \xi_t^{(i)} \end{array} \tag{9}$$

where t indexes different tags or learning tasks, and T is the total number of tags. By denoting $\mathbf{y}^{(i)} = [y_1^{(1)},\ldots,y_T^{(i)}]$ and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T};\ldots;\frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (Eq.(9)) as a joint SVM:

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{T\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{10}$$

where $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$, and $\bar{\xi}^{(i)} = \frac{1}{T}\sum_{t=1}^T \xi_t^{(i)}$. Eq.(10) is referred to as *joint SVM*. One rationale of Eq.(10) is that within the joint form of objectives and constraints, learning easy tasks can help the learning of challenging tasks. For example, if training data $(\mathbf{x}^{(i,)}, y_p^{(i)})$ can be easily classified correctly in the pth task (*i.e.*, $y^{(i)}(\mathbf{w}_p^\top\mathbf{x}^{(i)})/T > \frac{1}{T}$), it can offer some 'freedom' to other challenging tasks before violating constraint $\left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle_{\mathcal{H}} \ge 1$. Meanwhile, a more critical strength of Eq.(10) is that a linear output kernel is implicitly learned and absorbed in the model parameters $\mathbf{W}$. More rigorous explanation will be presented later in section 4.1. In addition, another key functionality joint SVM can afford is that we can also, based on our prior knowledge, explicitly define kernel functions on outputs $\mathbf{y}$ to improve their representational power (*e.g.* dependencies). Assume the kernel function defined on outputs are $K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})$ (the output kernel will be explained later) and the corresponding feature map is $\psi : \{-1,+1\}^T \to \mathcal{H}_\psi$, then Eq.(10) is modified to

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{11}$$

Similarly to a single conventional SVM, joint SVM Eq.(11) can be converted to its dual form

$$\begin{array}{rl} \arg\min_{\alpha_1,\cdots,\alpha_m} & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i\alpha_j K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{12}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)})\phi(\mathbf{x}^{(i)})^\top$. It can be seen, with the kernel matrix on outputs pre-computed, that the computational complexity of joint learning (Eq.(12)) is the same as the learning of one single SVM (Eq.(7)), which is a great advantage in efficiency.

Given a test input $\hat{\mathbf{x}}$, the prediction $\phi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^{m} \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{13}$$

Meanwhile, one computational issue is that there is no direct way (say, by inverting Eq.(13)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned} \hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \sum_{i=1}^m \alpha_i \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \end{aligned} \tag{14}$$

In general, there is no closed-form solution to Eq.(14), so usually approximate dynamic programming (ADP) is applied in searching for the optimum $\hat{\mathbf{y}}^*$. Here, we employ a simpler yet effective strategy. Since the number of tags associated with one image is rather small, most of the $\mathbf{y}$ in $\{+1, -1\}^T$ space are bad solutions. Therefore, when the training data size is large, the most likely solutions of Eq.(14), presumably, are covered by the outputs in training data $\{\mathbf{y}\}_{i=1}^m$. Consequently, we can find the optimum $\hat{\mathbf{y}}^*$ via a similar neighbour-based label transferring theme as [3, 10]:

$$\hat{\mathbf{y}}^* = \left(\sum_{k=1}^{K} \mathbf{y}^{(k)} w_k \right) / \sum_{k=1}^{K} w_k \tag{15}$$

$$w_j = \sum_{i=1}^{m} \alpha_i \beta_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) \tag{16}$$

where $k = \{j \in [1, m] : w_j > 0\}$ and maximum $K = 10$ neighbours are taken into account. Since α_i are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ were already computed in the training phase, only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing. Thus, the complexity in predicting is $\mathcal{O}(m)$.

4. Implicit and Explicit Linear Output Kernels on Tag-Sets

To transform the pairwise and triplet-wise dependencies between tags into the inner product of two outputs containing those tags, 2-degree and 3-degree Polynomial kernels are tried in [18] and it was reported that 2-degree is better than 3-degree. In [4, 16, 20], linear feature maps were exploited also for pairwise dependencies. In particular, linear output kernels and models were simultaneously learned in [16, 20], while the output kernel in [4] is pre-computed as a correlation matrix over output vectors. In this paper, based on the experience from previous literatures, we also only focus pairwise dependencies and study linear kernels (although higher-order kernels will also be tried in our experiments, and the performance among different kernels can be checked in section 7). Here we adopted strategies both in [16, 20] and in [4]. At first, we present that the linear output kernel can be implicitly, but more simply compared to [16, 20], learned when we train a joint SVM. Secondly, we developed a novel pre-designed linear kernel function, which can be seen as a replacement of the kernel with correlation matrix used in [4].

4.1. Implicit linear output kernel learning

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$matrix $\mathbf{P}$, via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{17}$$

where $\mathbf{\Omega} = \mathbf{P}^\top\mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top\mathbf{W}$, we can rewrite joint SVM (Eq.(11)) as:

$$\begin{array}{ll} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i\in\{1,\ldots,m\} \end{array} \tag{18}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in Eq.(18) will be pointless. In [20] one extra regularization on $\mathbf{\Omega}$, $\frac{1}{2}||\mathbf{\Omega}||_F^2$, was added into the objective function, while $||P||_F = 1$ was used in [16]. By contrast, a pseudo regularization on $\mathbf{P}$ is used in [11] via the re-construction loss from manually-corrupted data and $\mathbf{P}$. Here we apply a simpler strategy by using a compact regularizer, $\frac{1}{2}\mathbf{W}^\top\mathbf{\Omega}\mathbf{W}$, resulting in:

$$\begin{array}{ll} \arg\min_{\mathbf{U}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{U}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i\in\{1,\ldots,m\} \end{array} \tag{19}$$

Remarkably, Eq.(19) is equivalent to Eq.(11) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a simple joint SVM with no explicit kernel on outputs.

4.2. Odds-ratio based kernel

In this paper, we also explicitly design an odds-ratio based kernel over tag-sets to capture pairwise dependencies. The dependency between tags measures how much the appearance of one tag increases or decreases the chance of another tag to occur in the same label set. At first, we can estimate the probability of co-occurrence of two labels, w_r and w_s, form training data:

$$P(w_r, w_s) = \frac{\sum_{i=1,\ldots,m} \mathbf{y}_r^{(i)} = 1 \text{ and } \mathbf{y}_s^{(i)} = 1}{m}. \tag{20}$$

according to which, we can compute the odds ratio, a measure, of the dependency between those words by the well known formula [21]:

$$O_{rs} = \frac{P(w_r, w_s)P(\overline{w_r}, \overline{w_s})}{P(w_r, \overline{w_s})P(\overline{w_r}, w_s)}, \tag{21}$$

where $\overline{w_r}$ means the complement of w_r (counting those sample items where w_r does not occur). Then the odds ratio is symmetrized by taking its logarithm, where the 0 value expresses the independence and the positive (or negative) value corresponds to higher (or lower) co-occurrence of those words than the random case. The higher of the magnitude of the log-odds-ratio shows stronger deviation from the independence.

$$Q_{rs} \leftarrow \log(O_{rs}) \tag{22}$$

The odds-ratio based kernel on a pair of outputs can then be computed:

$$K_\psi^{Odd}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top}\mathbf{Q}\mathbf{y}^{(j)} \tag{23}$$

where $\mathbf{Q}$ is the log-odds-ratio matrix with $\mathbf{Q}_{rs} = Q_{rs}$.

5. Relation to Structured-Output Learning

Interestingly, although derived from a rather different starting point, our joint SVM (Eq.(11)) is the same as Maximum Margin Regression (MMR) [1], wherein the motivation is to seek a linear operator in arbitrary tensor product space $\psi(\mathbf{y}^{(i)}) \otimes \phi(\mathbf{x}^{(i)})$. In addition, Eq.(11) is also related to structural SVM [6, 2] by sharing the same objective, yet with different constraints. An empirical comparison of these two methods on hierarchical-label learning is in [22]. The solution of the MMR stands close to the Minimum Description Length Principle, see for example in [23], by providing a highly compressed description to complex learning problems. In particular, when a linear output kernel and Hamming loss function are used in structural SVM. Structural SVM can be converted to a rather similar formulation as joint SVM by decomposing Hamming loss and $\mathbf{y}$ element-wisely. The detailed derivation was presented in [4].

6. Online Learning of Joint SVM

In real-word applications, the number of images can be very huge and beyond the memory storage and computing capacities of normal PCs. For instance, millions of images are uploaded to Facebook^{TM} and Flicker^{TM} every day. Obviously, the computation of kernel matrix for even daily volume is impractical. The formulation of joint SVM also suggests an implementation of a "perceptron-like" algorithm . For simplicity, here we present the case where no output kernel is applied. We aim to demonstrate the transparency of the formulation of joint SVM, which allows us to inherit most of the machine learning techniques developed earlier. Consider the optimization problem in Eq.(10) when only the error term is minimized

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_{\mathcal{H}_y}) \\ \text{subject to} \quad & \{\mathbf{W} | \mathbf{W} : \mathcal{H}_x \to \mathcal{H}_y, \mathbf{W} \text{ a linear operator}\}, \end{aligned} \tag{24}$$

where λ is a prescribed margin, and the function $h(u)$ denotes the Hinge loss, that is

$$h(u) = \begin{cases} u & \text{if } u > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The error function that we are going to minimize has subgradient with respect to $\mathbf{W}$ and this can be computed independently in an incremental way for each term occurring in the summation Eq.(24). The reader can consult to [24] and [25] for details of incremental subgradient methods. The term-wise subgradient is equal to

$$\partial h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y})|_{\mathbf{W}} = \begin{cases} -\mathbf{y}^{(i)}\phi(x^{(i)})^T & \text{if } \lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

We can define the learning speed with a step size, denoted by s, and we obtain the "perceptron-like" algorithm given in Figure 1. In that algorithm $\mathbf{W}^{norm}$ denotes the L_2 normalized linear operator.

Input of the learner: The sample S, step size s
Output of the learner: $\mathbf{W} \in \mathbb{R}^{\mathcal{H}_y \times \mathcal{H}_x}$
Initialization: $t = 0$; $\mathbf{W}_t = \mathbf{0}$; $\mathbf{W}_t^{norm} = \mathbf{0}$; $\|\mathbf{W}_t\| = 0$
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input-output pair: $(\mathbf{x}_i, \mathbf{y}_i)$
    $\beta_i = \langle \mathbf{y}_i, \mathbf{W}_t^{norm}\boldsymbol{\phi}(x_i)\rangle_{\mathcal{H}_y}$
    if $\beta_i < \lambda$ **then**
      $\mathbf{W}_{t+1} = \mathbf{W}_t + s\mathbf{y}_i\boldsymbol{\phi}(x_i)^T$
      $t = t + 1$
      $\|\mathbf{W}_{t+1}\|^2 = \|\mathbf{W}_t\|^2 + s^2\|\mathbf{y}_i\|^2\|\phi(\mathbf{x}_i)\|^2 + 2s\beta_i$
      $\mathbf{W}_{t+1}^{norm} = \mathbf{W}_{t+1}/\|\mathbf{W}_{t+1}\|$
    end if
  end for
until

(27)

Figure 1: Primal "perceptron-like" online learning algorithm for joint SVM.

The departure from the original perceptron algorithm, see for example in [26] and [27], is very moderate. Here we need to learn a matrix realizing the projection of the input vectors into the output space. The incremental subgradient based update employs the direct product of the corresponding output and input vectors to update the projection matrix. Furthermore a normalization step is also included as a certain regularization step, similar approach is proposed in [28].

A dual version of perceptron algorithm can be derived to learn vector outputs. Assume $\mathbf{W}$ is expressible by the training instances, then we have the optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \sum_{j=1}^{m} \alpha_j \overbrace{\langle \mathbf{y}^{(i)}, \mathbf{y}^{(j)} \rangle}^{\kappa_{ij}^{y}} \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}^{(i)}), \boldsymbol{\phi}(\mathbf{x}^{(j)}) \rangle}^{\kappa_{ij}^{\phi}}) \\ \text{subject to} \quad & \alpha_j \geq 0, \ j = 1, \ldots, m, \end{aligned} \tag{28}$$

The partial derivatives for $\alpha_i, \ k = 1, \ldots, m$ equals to

$$\partial h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi})|_{\alpha_i} = \begin{cases} -\kappa_{ij}^{y} \kappa_{ij}^{\phi} & \text{if } h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Finally the corresponding dual perceptron algorithm is formulated according to Figure 2. An analogue of

Input of the learner: The training set S, step size s,
Output of the learner: $(\alpha_j), \ j = 1, \ldots, m$,
Initialization: $\alpha_j = \mathbf{0}; \ j = 1, \ldots, m$,
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input: $\mathbf{x}^{(i)} \in \mathbb{R}^n$;
    if $\langle \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) < \ \lambda$ **then**
      for $j = 1, 2, \ldots, m$ **do**
        $\alpha_j = \alpha_j + s \kappa_{ij}^{y} \kappa_{ij}^{\phi}$
      endif
    end if
  end for
until

(30)

Figure 2: Dual "perceptron-like" online learning algorithm for joint SVM.

the standard Novikoff theorem provides an upper bound on the number of updates and a lower bound on the achievable margin in the primal formulation. Here we follow the derivation that was presented in [29]. Let us define the margin for perceptron learner as

$$\gamma(\mathbf{W}, S, \phi) = \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) \in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W} \boldsymbol{\phi}(\mathbf{x}^{(i)}) \rangle_F}{\|\mathbf{W}\|_F}. \tag{31}$$

Then we can claim the following statement not assuming the normalization step in the algorithm:

Theorem 1. *Let $S = \{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\} \subset (\mathcal{Y} \times \mathcal{X}), \ i = 1, \ldots$ be a sample set independently and identically drawn from an unknown distribution and let $\phi : \mathcal{X} \to \mathcal{H}_\phi$ be an embedding into a Hilbert space, furthermore assume that $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$ for all i, and that the learning rate, the step size, s is a fixed positive real number. Suppose there exists a linear operator $\mathbf{W}^*$ such that $\|\mathbf{W}^*\|_F = 1$ and*

$$\gamma(\mathbf{W}^*, S, \phi) \geq \Gamma, \tag{32}$$

and the algorithm stops when the functional margin 1 is achieved.

1. *Then the number of updates made by Algorithm (1) is bounded by*

$$t \leq \frac{1}{\Gamma^2} \left(1 + \frac{2}{s}\right). \tag{33}$$

2. *Then for the solution* $\mathbf{W}_t$ *in Algorithm (1) we have*

$$\gamma(\mathbf{W}_t, S, \phi) \geq \frac{\Gamma}{s+2}. \tag{34}$$

Proof 1. 1. *Following the proof of the original Novikoff theorem [30], we first upper bound the norm of the matrix* $\mathbf{W}_t$ *obtained after* t *updates:*

$$\begin{aligned} \|\mathbf{W}_t\|_F^2 &= \|\mathbf{W}_{t-1}\|_F^2 + 2s\langle \mathbf{y}^{(i)}\mathbf{W}_{t-1}\boldsymbol{\phi}(x^{(i)})\rangle_{\mathcal{H}_y} + s^2\|\mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T\|_F^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2\|\mathbf{y}^{(i)}\|^2\|\boldsymbol{\phi}(x^{(i)})\|^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2 \\ &\leq ts(s+2). \end{aligned} \tag{35}$$

We now provide a reverse inequality for the inner product with $\mathbf{W}^\star$*:*

$$\begin{aligned} \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T, \mathbf{W}^\star\right\rangle_F \\ &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}, \mathbf{W}^\star\boldsymbol{\phi}(x^{(i)})\right\rangle_{\mathcal{H}_y} \\ &\geq \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\Gamma \\ &\geq ts\Gamma. \end{aligned}$$

Then we can create the squeezing inequality:

$$ts(s+2)\|\mathbf{W}^\star\|_F^2 \geq \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2. \tag{36}$$

implying the result.

2. *Taking the bound Eq.(33) for* t *and substituting into Eq.(35) we arrive at*

$$\|\mathbf{W}_t\|_F \leq \frac{s+2}{\Gamma}. \tag{37}$$

Then for the margin we have

$$\gamma(\mathbf{W}_t, S, \phi) \geq \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W}_t\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_F}{\|\mathbf{W}_t\|_F} \tag{38}$$

$$\geq \frac{1}{\|\mathbf{W}_t\|_F} \tag{39}$$

$$\geq \frac{\Gamma}{s+2}, \tag{40}$$

which proves the statement.

125 Sparsity bounds [31] can also be used to translate this bound on the number of updates into a corresponding bound on the generalization of the resulting classifier.

All results included in this paper are assumed the normalization conditions, $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$, of Theorem 1. By forcing the normalization of $\|\mathbf{W}_t\|$ for all t in Algorithm 1 allows us to simplify and sharpen the proof of Theorem 1. In this case Expression (35) collapses into a identity of both sides of the equation, therefore instead of (36) we have

$$1 = \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2, \tag{41}$$

from which we gain that

$$t \leq \frac{1}{s\Gamma}, \tag{42}$$

Dataset	labels	Number of training instances	test instances	average labels
Corel5k	260	4500	500	3.3965
Espgame	268	18689	2081	4.6859
Iaprtc12	291	17665	1962	5.7187

Table 1: Statistics of three benchmark datasets.

and in case of the margin we can write

$$\gamma(\mathbf{W}_t, S, \phi) \quad \geq \quad 1, \tag{43}$$

which statements are significantly stronger than those appearing in the general case. The price that we need to pay for this result is the slower algorithm.

In comparing our algorithm with other online learning schemes of maximum margin based learning methods, e.g. SVM, (see some realizations in [32] and [33]), we need to bear in mind that our methods learns to predict all components of the label vector within one optimization problem. Those methods which can deal only with binary classification problems have to solve as many binary label subproblems as the number of labels independently, therefore their overall computational complexity turns to be significantly higher than our approach.

7. Experiment

7.1. Databases

In our experiments, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies [7, 8, 9, 10, 3, 11] with performance evaluations reported therein. Therefore, we can easily compare our method with others. Statistics of three benchmark datasets are summarized in Table 1. Readers are referred to [3] for more details of three datasets.

7.2. Feature Extraction

In our experiment, we worked with 15 visual features extracted in [10]. More concretely, they contain one Gist descriptor, six global color histograms and eight histograms of local bag-of-words texture features [3]. The description of 15 features are summarized in Table 2. Readers are referred to [10] for more detail on extracting these features. These features were also used in [10] and [11]. A similar visual feature set without layout was extracted and used in [3], while 30 visual feature with spatial layouts were used in [9].

7.3. Evaluation metric

In our experiment, we evaluated annotation performance using *precision (P), recall (R), F-1 measure (F)*, which were commonly used in previous studies. For each tag, the precision is computed as ratio between the number of images assigned the tag correctly and total number of images predicted to have the tag, while the recall is the number of images assigned the tag correctly, divided by the number of images which truly have the tag. Then precision and recall are averaged across all tags. At last, F1 measure is calculated as $F = 2\frac{P \times R}{P+R}$.

7.4. Model selection

In three original databases, training/test data are already divided in advance. Therefore, given a learned model, there exist no variance in prediction performance on fixed test data. Hyper-parameters in Gaussian kernels, polynomial kernels and odds-ratio based kernels are found by cross validation restricted to the training data, namely it is divided into validation test and validation training parts. Then the learner is trained only on the validation training items. At the end those values of the parameters have been chosen which maximize the $F1$ score on the validation test.

[3] All features are available on http://lear.inrialpes.fr/people/guillaumin/data.php.

Feature	Dimension	Source	Descriptor	Location	Layout
DenseHue	100	texture	hue	dense	no
DenseHueV3H1	300	texture	hue	dense	yes
DenseSift	1000	texture	sift	dense	no
DenseSiftV3H1	3000	texture	sift	dense	yes
Gist	512	-	holistic	-	-
HarrisHue	100	texture	Hue	Harris points	no
HarrisHueV3H1	300	texture	Hue	Harris points	yes
HarrisSift	1000	texture	sift	Harris points	no
HarrisSiftV3H1	3000	texture	sift	Harris points	yes
Hsv	4096	color	HSV	-	no
HsvV3H1	5184	color	HSV	-	yes
Lab	4096	color	LAB	-	no
LabV3H1	5184	color	LAB	-	yes
Rgb	4096	color	RGB	-	no
RgbV3H1	5184	color	RGB	-	yes

Table 2: Description of 15 visual features tried in our experiments.

	Corel5k			Espgame			iaprtc12		
Feature	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
DenseHue	33.3	26.0	29.2	28.5	16.4	20.8	26.7	17.5	21.1
DenseHueV3H1	38.1	30.7	34.0	32.9	18.8	23.9	31.8	21.0	25.3
DenseSift	40.2	32.2	35.8	33.3	24.6	28.3	38.4	26.5	31.4
DenseSiftV3H1	**43.7**	**34.6**	**38.6**	**35.2**	**26.3**	**30.1**	**40.5**	**28.3**	**33.3**
Gist	33.7	26.9	29.9	28.3	20.6	23.9	33.2	23.5	27.5
HarrisHue	31.0	24.6	27.4	27.4	16.3	20.4	27.6	18.3	22.0
HarrisHueV3H1	34.5	27.7	30.7	31.5	18.4	23.2	31.9	21.9	26.0
HarrisSift	39.9	32.1	35.6	33.2	25.5	28.9	39.4	26.9	32.0
HarrisSiftV3H1	40.2	33.4	36.5	**34.6**	**26.2**	**29.8**	**40.7**	**29.7**	**34.3**
Hsv	38.3	30.6	34.0	30.0	18.7	23.1	32.6	21.1	25.7
HsvV3H1	40.8	33.8	37.0	33.8	21.6	26.4	35.4	24.1	28.7
Lab	35.1	27.5	30.8	27.2	16.4	20.5	28.4	17.9	22.0
LabV3H1	39.7	30.7	34.6	30.0	18.9	23.1	32.7	20.8	25.4
Rgb	42.0	33.4	37.2	26.2	16.4	20.2	32.8	20.6	25.3
RgbV3H1	**42.1**	**34.5**	**38.0**	29.6	19.2	23.3	35.7	23.0	28.0

Table 3: Performance of joint SVM without explicit output kernel on different individual features.

7.5. Selecting optimal features

In [10, 11], all 15 features were used for predicting tags. However, we believe that there exist some redundancies in all 15 features. Also, some features might be weakly relevant to the annotation task. A more efficient way is to identify a few most relevant features and use them for prediction. To this end, we apply joint SVM without explicit output kernel on different features, and list their discriminative abilities in Table 3, in which the best and second runner-up features are highlighted with bold font. We can see that `DenseSfitV3H1` is consistently more reliable than other features in three datasets. In addition, `HarrisSiftV3H1` is also optimal or close to optimal in Espgame and Iaprtc12 respectively. However, `HarrisSiftV3H1` is inferior to `RgbV3H1` in Corel5k. Therefore, in our later experiments, we used `DenseSfitV3H1`+`RgbV3H1` on Corel5k, while `DenseSfitV3H1`+`HarrisSiftV3H1` on Espgame and Iaprtc12. We combined two features by simply concatenating one feature vector after the other one.

	Training Time (sec)	Testing Time (sec)	Testing Performance		
			Precision (%)	Recall (%)	F1 (%)
Independent SVMs (Gau)	6285.11	117.20	15.3	22.1	18.1
Independent SVMs (Pol)	4612.23	147.9	15.1	29.7	20.0
Joint SVM (Gau)	80.68	**6.92**	40.8	37.1	38.9
Joint SVM (Pol)	**76.48**	9.11	**48.5**	**38.0**	**42.6**

Table 4: Comparison between one joint SVM and multiple SVMs on Corel5k dataset. Two input kernels (Gaussian and 2-degree polynomial) are tried in both learners.

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM [9]	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC [3]	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp [10]	33.0	42.0	37.0	39.0	27.0	**32.0**	45.0	**34.0**	**39.0**
FastTag [11]	32.0	**43.0**	37.0	**46.0**	22.0	30.0	**47.0**	26.0	34.0
JSVM	**48.5**	38.0	**42.6**	32.7	**31.6**	**32.2**	42.2	29.4	34.6
JSVM+Odd	**48.8**	37.1	**42.2**	27.4	27.1	27.2	32.9	28.6	30.6
JSVM+Pol(2)	46.6	37.0	41.3	32.6	24.4	27.9	37.9	26.6	31.2
JSVM+Pol(3)	41.5	31.3	35.7	28.5	21.3	24.4	38.0	26.1	31.0
JSVM-Per	37.5	29.8	33.2	25.0	19.0	21.6	29.2	20.8	24.3

Table 5: Comparison between different versions of joint SVM and other related methods on three benchmark databases.

7.6. Comparison with Independent SVMs

At first, we applied both a joint SVM, and many independent SVMs on Corel5k dataset with the feature combination selected above. To ensure fairness, no user-designed kernel is used on output for the joint SVM (plain joint SVM), while Gaussian kernel and 2-degree polynomial kernel are tried for inputs in both learners. In the learning phase, the optimization problems (Eq.(7)) and (Eq.(12)) were solved with the same coordinate descent method [20]. In addition, the same cross-validation procedure is used for both many independent SVMs and the joint SVM to find the best hyper-parameters C, d, c, σ. To measure the efficiency, training and testing time were recorded as well. All experiments were run on the same simulation and hardware conditions (Python 3, Intel Core i7). The comparison of accuracy and efficiency between independent SVMs and joint SVM is presented in Table 4. While the learning and testing time of independent SVMs scale with the number of tags, the computation time of joint SVM approximately equals a SVM for single-tag classification. At the same time, in terms of accuracy, joint SVM also worked much better than independent SVMs. We can also see that 2-degree polynomial input kernel worked better than Gaussian input kernel for both learners.

7.7. Comparison with state-of-the-art

More intensive experiments of joint SVM were conducted with different pre-designed, explicit output kernels: odds-ratio based kernel (JSVM+Odd), 2-degree polynomial (JSVM+Pol(2)), 3-degree polynomial (JSVM+Pol(3)). Also, online learning algorithm of joint SVM (JSVM-Per) was also implemented. All configurations were run on all three datasets, with optimal feature combination and 2-degree polynomial kernel on inputs. The experimental results, together with the reported results from other related work, are presented in Table 5. We can see that plain joint SVM (JSVM) outperforms all other results on Corel5k and Espgame datasets, yielding the best results so far. JSVM is also the second best result on Iaprtc12 dataset. The results of JSVM+Odd and JSVM+Pol(2) are similar on all three datasets. It is worth noting that JSVM+Odd and JSVM+Pol(2) also worked better than previous methods by a large margin. Meanwhile, JSVM+Pol(3) is worse than JSVM+Pol(2). JSVM-Per's performance is inferior to other JSVM versions, although it is still better than two classic methods [9, 3].

7.8. Discussions

Based on our experiments, it seems that plain joint SVM (JSVM) works more robustly than the joint SVMs with explicit output kernels. In order to dig deeper to find an explanation, we can study the correlation matrices of output tag-sets in three datasets. In Figure 3, for each dataset, we plot the histograms (in log scale) of all correlation values in both training sets and testing sets. We can see that most entries in correlation matrices are 0, which means that the pairwise correlation (or roughly speaking, dependencies) is rather sparse. Although JSVM, JSVM+Odd both encode linear pairwise dependencies, it should be reminded that the implicit output kernel in JSVM is in regularization term, which implies that simpler output kernels (dependencies) are encouraged. However, JSVM+Odd does not have this preference. Therefore, JSVM can implicitly learned most simple output kernels when no more complex ones are needed. Analogously, the same principle can explain why JSVM+Pol(2), or even JSVM+Pol(3) led to worse results. If we look closer, we can observe that in Corel5k datasets, stronger correlations are displayed in its testing set, and correspondingly, the performance gaps between JSVM, JSVM+Odd and JSVM+Pol(2) are also rather small.

As for JSVM-Per, one reason of its inferiority is that the regularization is computed instance-wisely, which might conflict the global effect it is supposed to have. However, we gain tractability, for extremely large datasets, with acceptable accuracy cost. As a future direction work, we will investigate some alternative online regularization strategies.

8. Conclusions

A novel joint SVM was presented for automatic image tagging. It is superior to conventional SVMs based on our empirical results. In particular, it compares favorably with state-of-the-art methods. As possible future work directions, we would like to apply and improve joint SVM in other multi-label learning domains.

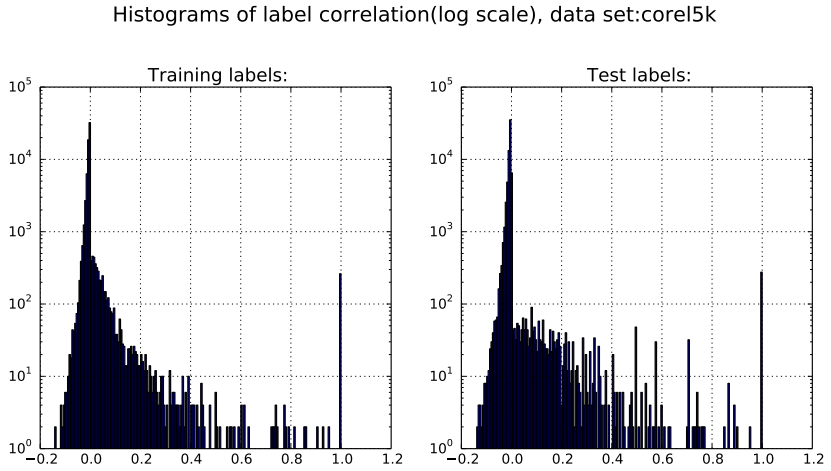
Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

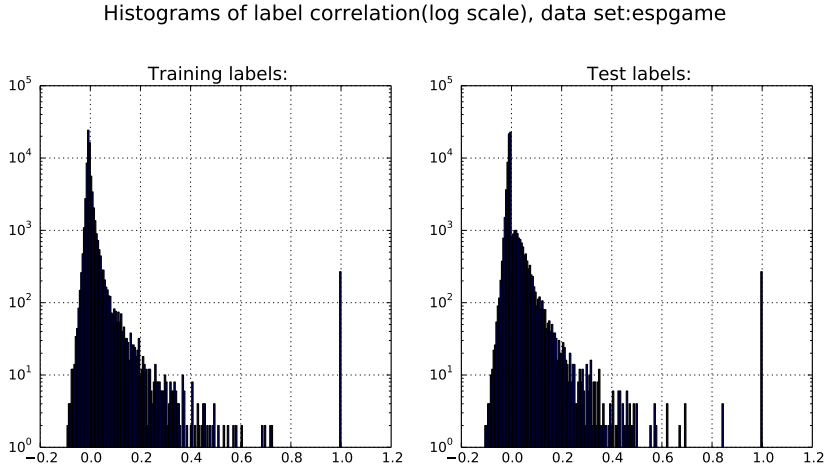
Reference

[1] S. Szedmak, J. Shawe-taylor, Learning via linear operators: Maximum margin regression, Tech. rep., University of Southampton, UK (2005).
[2] I. Tsochantaridis, T. Hofmann, T. Joachims, Y. Altun, Support vector machine learning for interdependent and structured output spaces, in: ICML, 2004.
[3] A. Makadia, V. Pavlovic, S. Kumar, Baselines for image annotation, International Journal of Computer Vision 90 (2010) 88–105.
[4] B. Hariharan, S. V. N. Vishwanathan, M. Varma, Efficient max-margin multi-label classification with applications to zero-shot learning, Machine Learning 88 (1-2) (2012) 127–155.
[5] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Machine Learning 73 (3) (2008) 243–272.
[6] B. Taskar, V. Chatalbashev, D. Koller, C. Guestrin, Learning structured prediction models: A large margin approach, in: ICML, 2005.
[7] V. Lavrenko, R. Manmatha, J. Jeon, A model for learning the semantics of pictures, in: NIPs, 2004.
[8] D. M. Blei, M. I. Jordan, Modeling annotated data, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, 2003.
[9] S. L. Feng, R. Manmatha, V. Lavrenko, Multiple bernoulli relevance models for image and video annotation, in: CVPR, 2004.
[10] M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: ICCV, 2009.
[11] M. Chen, A. Zheng, K. Q. Weinberger, Fast image tagging, in: ICML, 2013.
[12] D. R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Computation 16 (2004) 2639–2664.
[13] X. Qi, Y. Han, Incorporating multiple svms for automatic image annotation, Pattern Recogn. 40 (2) (2007) 728–741.

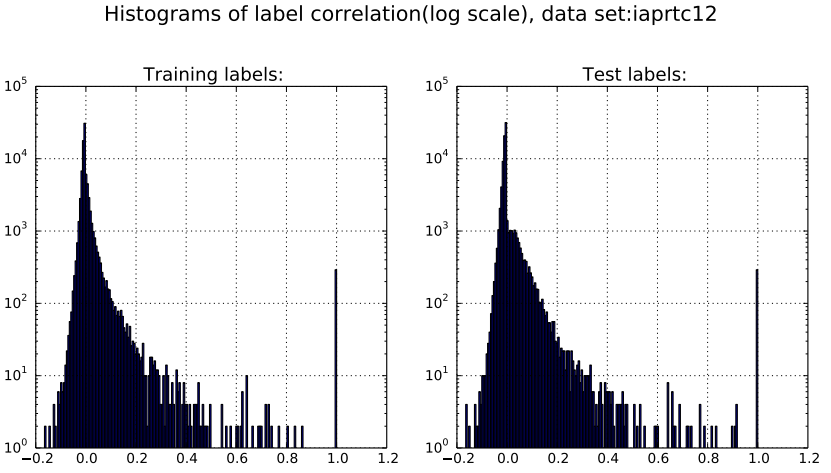
[14] S. Szedmák, T. D. Bie, D. R. Hardoon, A metamorphosis of canonical correlation analysis into multivariate maximum margin learning, in: ESANN, 2007.
[15] J. Rousu, C. Saunders, S. Szedmák, J. Shawe-Taylor, Kernel-based learning of hierarchical multilabel classification models, Journal of Machine Learning Research 7 (2006) 1601–1626.
[16] Y. Zhang, D.-Y. Yeung, Multilabel relationship learning, ACM Trans. Knowl. Discov. Data 7 (2) (2013) 1–30.
[17] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: NIPs, 2001.
[18] Y. Guo, D. Schuurmans, Multi-label classification with output kernels, in: ECML/PKDD, 2013.
[19] V. Vapnik, Statistical learning theory, Wiley, 1998.
[20] F. Dinuzzo, C. S. Ong, P. V. Gehler, G. Pillonetto, Learning output kernels with block coordinate descent, in: ICML, 2011.
[21] S. M. Hailpern, P. F. Visintainer, Odds ratios and logistic regression: further examples of their use and interpretation, Stata Journal 3 (3) (2003) 213–225.
[22] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, J. Rousu, Towards structured output prediction of enzyme function, in: BMC Proceedings, 2(4):S2, 2008.
[23] P. D. Grünwald, The Minimum Description Length Principle, MIT Press, 2007.
[24] D. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scienctific, 1999.
[25] K. Kiwiel, Convergence of approximate and incremental subgradient methods for convex optimization, Journal of Optimization 14, 3 (2004) 807–840.
[26] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
[27] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning and Games, Cambridge University Press, 2006.
[28] C. Gentile, A new approximate maximal margin classification algorithm, Journal of Machine Learning Research 2 (2001) 213–242.
[29] Y. Li, H. Zaragoza, R. Herbich, J. Shawe-Taylor, J. Kandola, The perceptron algorithm with uneven margins, in: Proceedings of the International Conference of Machine Learning (ICML'2002), 2002.
[30] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods, Cambridge University Press, New York, NY, USA, 2000.
[31] T. Graepel, R. Herbrich, J. Shawe-Taylor, Generalisation error bounds for sparse linear classifiers, in: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, Morgan Kaufmann Publishers Inc., 2000, pp. 298–303.
[32] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, Journal of Machine Learning Research 6(Sep) (2005) 1579–1619.
[33] S. Shalev-Shwartz, T. Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, Journal of Machine Learning Research 14(Feb) (2013) 567–599.



(a)



(b)



(c)

Figure 3: The histograms (in log scale) of all correlation values in both training sets and testing sets: (a) Corel5k, (b) Espgame (c) Iartc12.

Implicit Learning of Simpler Output Kernels for Multi-Label Prediction

Hanchen Xiong Sandor Szedmak Justus Piater
Institute of Computer Science, University of Innsbruck
Innsbruck, A-6020, Austria
{*hanchen.xiong, sandor.szedmak, justus.piater*}*@uibk.ac.at*

Abstract

It has been widely agreed that, in multi-label prediction tasks, capturing and utilizing dependencies among labels is quite critical. Therefore, a research tendency in multi-label learning is that increasingly more sophisticated dependency structures on labels (*e.g.* output kernels) are proposed. We show that, however, over-complex dependency structures will harm more than help learning when the underling dependency is relatively weak. To avoid overfitting on structures, a regularization on label-dependency is desirable. In this paper, we put forward a novel *joint-SVM* for multi-label learning. Compared to other discriminative learning schemes, joint-SVM has two strengths: at first, the complexity of training joint-SVM is almost the same as training a single regular SVM, which is quite efficient; secondly, in joint-SVM, a linear output kernel on multi-label is implicitly learned and a regularization on the output kernel is implicitly added, which enhances generalization ability. In our experimental results on image annotation, joint-SVM compares favorably state-of-the-arts methods.

1 Predict Multi-label as Structured Outputs

In the past two decades, support vector machines (SVMs) have displayed remarkable successes in various application domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [12] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are two input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$, the primal form of training SVM is:

$$\begin{aligned} \arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_{i=1}^m \xi^{(i)} \\ \text{s.t.} \quad & y^{(i)} \left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)}) \right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{2}$$

where $\mathbf{w}$ is a linear hyperplane in $\mathcal{H}_\phi$, $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is a trade-off parameter. (2) differs from usual SVM formulation slightly at the absence of a bias term. Here we ignore the bias since it can be absorbed in $\mathbf{w}$. The computational advantage of kernels become obvious when the primal form of SVM (2) is reformulated to its dual form:

$$\begin{aligned} \arg \min_{\alpha_1, \alpha_2, \ldots, \alpha_m} \quad & \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{s.t.} \quad & \forall i, 0 \leq \alpha_i \leq C \end{aligned} \tag{3}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \text{sgn}\left(\mathbf{w}^\top \phi(\hat{\mathbf{x}})\right) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y^{(i)} K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})\right). \tag{4}$$

We can denote $y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)})\right)$ in the constraints of (2) as a score function $F(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w})$, then for binary outputs $y^{(i)}$, $F\left(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w}\right) - F\left(\mathbf{x}^{(i)}, -y^{(i)}; \mathbf{w}\right) = 2 \times F\left(\mathbf{x}^{(i)}), y^{(i)}; \mathbf{w}\right)$. Also, a distance function between binary outputs can be denoted as $d(y^{(i)}, -y^{(i)}) = |y^{(i)} - (-y^{(i)})| = 2$. Then by replacing C with $\frac{C}{2}$, (2) can be rewritten as:

$$\begin{aligned} \arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_{i=1}^{m} \xi^{(i)} \\ \text{s.t.} \quad & \forall i, \underbrace{F\left(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w}\right) - F\left(\mathbf{x}^{(i)}, -y^{(i)}; \mathbf{w}\right)}_{\Delta_F(y^{(i)}, -y^{(i)})} \geq d(y^{(i)}, -y^{(i)}) - \xi^{(i)}, \xi^{(i)} \geq 0 \end{aligned} \tag{5}$$

which is a binary-output case of structural SVM [11] (see later). By using hinge-loss representation for $\xi^{(i)}$, (5) is:

$$\arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{m} \max\{0, d(y^{(i)}, -y^{(i)}) - \Delta_F(y^{(i)}, -y^{(i)})\} \tag{6}$$

Structural SVM [11] is an extension of SVM for structured-outputs, in which, however, the margin to be maximized is defined as the score gap between the desired output and the runner-up. Assume that structured outputs $\mathbf{y} \in \mathcal{Y}$, and the score function is linear in some *combined feature representation* of inputs and outputs $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$: $F(\mathbf{x}, \mathbf{y}; \mathbf{W}) = \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle$, then the objective function of structural SVM is:

$$\arg \min_{\mathbf{W} \in \mathbb{R}^{\mathbf{\Psi}}} \frac{1}{2}||\mathbf{W}||^2 + C \sum_{i=1}^{m} \max_{\mathbf{y}' \in \mathcal{Y}} \left\{ d(\mathbf{y}^{(i)}, \mathbf{y}') - \Delta_F(\mathbf{y}^{(i)}, \mathbf{y}') \right\} \tag{7}$$

where $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}') = F(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}) - F(\mathbf{x}^{(i)}, \mathbf{y}'; \mathbf{W})$ and $d(\mathbf{y}^{(i)}, \mathbf{y}')$ is a distance function defined on structured outputs. In multi-label scenario, given a set of T labels, then outputs are T-dimensional binary vector $\mathbf{y} = [y_1, \cdots, y_t, \cdots, y_T]^\top \in \mathbb{B}^T$. When we define the score function $F\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}\right) = \langle \mathbf{W}, \phi(\mathbf{x}^{(i)}) \otimes \mathbf{y}^{(i)} \rangle$, and use *Hamming distance* on outputs, then because of linear decomposability, (7) can be rewritten as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{\mathcal{H}_\phi \times \mathbb{R}^T}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \textstyle\sum_{i=1}^{m} \sum_{t=1}^{T} \max_{y'_t = \{-1, +1\}} \left\{ d(y_t^{(i)}, y'_t) - \Delta_F(y_t^{(i)}, y'_t) \right\} \\ \Downarrow & \\ \arg \min_{\mathbf{w}_1, \cdots, \mathbf{w}_T \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \textstyle\sum_{t=1}^{T} \left\{ \tfrac{1}{2}||\mathbf{w}_t||^2 + C \sum_{i=1}^{m} \max\left\{0, d(y_t^{(i)}, -y_t^{(i)}) - \Delta_F(y_t^{(i)}, -y_t^{(i)})\right\}\right\} \end{aligned} \tag{8}$$

where $\langle \cdot, \cdot \rangle_F$ denotes Frobenius product and $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$.

2 Joint SVM

It can be seen (by linking (6) and (8)) that, with linearly decomposable score functions and output distances, using structural SVM on multi-label learning is equivalent to learning T SVMs jointly. This is closely related to multi-task learning frameworks [1], where different learning tasks are connected by summing up their objectives and constraints respectively:

$$\begin{aligned} \min \quad & \tfrac{1}{2} \textstyle\sum_{t=1}^{T} ||\mathbf{w}_t||^2 + C \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_t^{(i)} \\ \text{w.r.t.} \quad & \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi \times 1} \\ \text{s.t.} \quad & \textstyle\sum_{t=1}^{T} y_t^{(i)} \left(\mathbf{w}_t^\top \phi(x^{(i)})\right) \geq T - \sum_{t=1}^{T} \xi_t^{(i)} \end{aligned} \tag{9}$$

By denoting $\mathbf{y}^{(i)} = [y_1^{(1)}, \ldots, y_T^{(i)}]$, and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T}; \ldots; \frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (9) as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^T \times \mathcal{H}_\phi} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \textstyle\sum_{i=1}^{m} \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{10}$$

which is referred to as *joint SVM*. When linear output kernels ($K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \langle \psi(\mathbf{y}^{(i)}), \psi(\mathbf{y}^{(j)}) \rangle$) [4, 7, 13] are applied on outputs, (10) will be:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{11}$$

Since the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ is still preserved, join SVM solves the same problem as structural SVM. However, one strength of joint SVM is that its training complexity is almost the same as a single SVM, by contrast to the exponential complexity in structural SVM. Similarly to regular SVM, joint SVM can be converted to its dual form

$$\begin{aligned} \arg \min_{\alpha_1, \cdots, \alpha_m} \quad & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i \alpha_j K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{s.t} \quad & \forall i, 0 \leq \alpha_i \leq C \end{aligned} \tag{12}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)}) \phi(\mathbf{x}^{(i)})^\top$. It can be seen that, with the kernel matrix on outputs precomputed, the computational complexity of joint SVM (12) is the same as the learning of one single SVM (3), which is a great advantage in efficiency. Meanwhile, when more general output kernels are used, then the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ will be violated, then joint SVM becomes a special case of max-margin regression [10], which seeks to learn linear operators $\mathbf{W} : \mathcal{H}_\phi \to \mathcal{H}_\psi$ from general $\phi(\mathbf{x}) \otimes \psi(\mathbf{y})$.

Given a test input $\hat{\mathbf{x}}$, the prediction $\psi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^m \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{13}$$

Meanwhile, there is no direct way (say, by inverting Eq.(13)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned} \hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y} \in \{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y} \in \{+1,-1\}^T} \sum_{i=1}^m \alpha_i \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \end{aligned} \tag{14}$$

In general, there is no closed-form solution to Eq.(14), so here we use a similar neighbour-based label transferring theme as [9, 6]:

$$\hat{\mathbf{y}}^* = \left(\sum_{k=1}^K \mathbf{y}^{(k)} w_k \right) \Big/ \sum_{k=1}^K w_k \qquad w_j = \sum_{i=1}^m \alpha_i \beta_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) \tag{15}$$

where $k = \{j \in [1, m] : w_j > 0\}$ and maximum $K = 10$ neighbours are taken into account. Since α_i are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ were already computed in the training phase, only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing. Thus, the complexity in predicting is $\mathcal{O}(m)$.

3 Implicit Learning and Regularization of Output Kernels

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$ matrix $\mathbf{P}$[3, 4, 7, 13], via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus the corresponding linear output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{16}$$

where $\mathbf{\Omega} = \mathbf{P}^\top \mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top \mathbf{W}$, we can rewrite joint SVM (11) as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{17}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in (17) will be pointless. Different regularizations on $\mathbf{P}$ have been proposed in previous work. In [4] one extra regularization on $\mathbf{\Omega}$, $\frac{1}{2}||\mathbf{\Omega}||_F^2$, was added into the objective function, while $||P||_F = 1$ was used in [13]. By

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM [5]	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC [9]	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp [6]	33.0	42.0	37.0	39.0	27.0	**32.0**	45.0	**34.0**	**39.0**
FastTag [3]	32.0	**43.0**	37.0	**46.0**	22.0	30.0	**47.0**	26.0	34.0
JSVM	**48.5**	38.0	**42.6**	32.7	**31.6**	**32.2**	42.2	29.4	34.6
JSVM+Pol(2)	46.6	37.0	41.3	32.6	24.4	27.9	37.9	26.6	31.2
JSVM+Pol(3)	41.5	31.3	35.7	28.5	21.3	24.4	38.0	26.1	31.0

Table 1: Comparison between different versions of joint SVM and other related methods on three benchmark databases. P, R and F1 denote precision, recall and F1 measure respectively.

contrast, a pseudo regularization on $\mathbf{P}$ is used in [3] via the re-construction loss from manually-corrupted data and $\mathbf{P}$. Similar to [4], we want to add a regularizer to control overfitting from output dependency-structures. Meanwhile, by merging regularization on $\mathbf{W}$ and $\mathbf{P}$, we obtain a more compact regularizer, $\frac{1}{2}\mathbf{W}^\top\mathbf{\Omega}\mathbf{W}$, resulting in:

$$\begin{aligned} \arg \min_{\mathbf{U}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{U}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i\in\{1,\ldots,m\} \end{aligned} \tag{18}$$

Remarkably, (18) is equivalent to (11) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a plain joint SVM with no explicit kernel on outputs. In addition, a regularization on the output kernel is also implicitly added.

4 Experiments

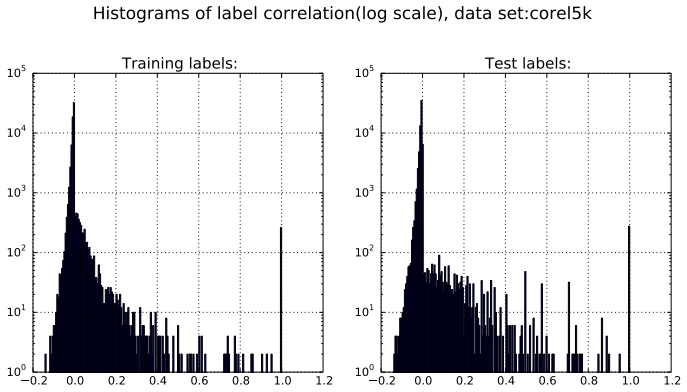
In our experiments, we evaluated the propose joint SVM on image annotation tasks. Here, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies [8, 2, 5, 6, 9, 3] with performance evaluations reported therein. Therefore, we can easily compare our method with others. We used the same visual features as in [6, 3]. Three types of joint SVMs with different output kernels are tested: plain joint SVM (JSVM), 2-degree polynomial (JSVM+Pol(2)) and 3-degree polynomial (JSVM+Pol(3)).

The experimental results, together with the reported results from other related work, are presented in Table 1. We can see that plain joint SVM (JSVM) outperforms all other results on Corel5k and Espgame datasets. JSVM is also the second best result on Iaprtc12 dataset. JSVM+Pol(2) also worked better than some old methods [5, 9]. Meanwhile, JSVM+Pol(3) is worse than JSVM+Pol(2).

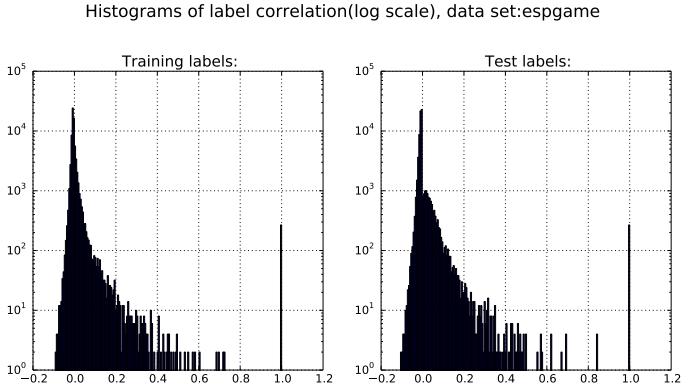
Discussions Based on our experiments, it seems that plain joint SVM (JSVM) works more robustly than the joint SVMs with explicit output kernels. In order to dig deeper to find an explanation, we can study the correlation matrices of output tag-sets in three datasets. In Figure 1, for each dataset, we plot the histograms (in log scale) of all correlation values in both training sets and testing sets. We found that most entries in correlation matrices are 0, which means that the pairwise correlation (or roughly speaking, dependencies) is rather sparse. Although JSVM, JSVM+Pol(2) both encode pairwise dependencies, it should be reminded that the implicit linear output kernel in JSVM is in regularization term, which implies that simpler output kernels (dependencies) are encouraged. However, JSVM+Pol(2) does not have this preference. Therefore, JSVM can implicitly learned most simple output kernels when no more complex ones are needed. Analogously, the same principle can explain why even JSVM+Pol(3) led to worse results.

5 Conclusions

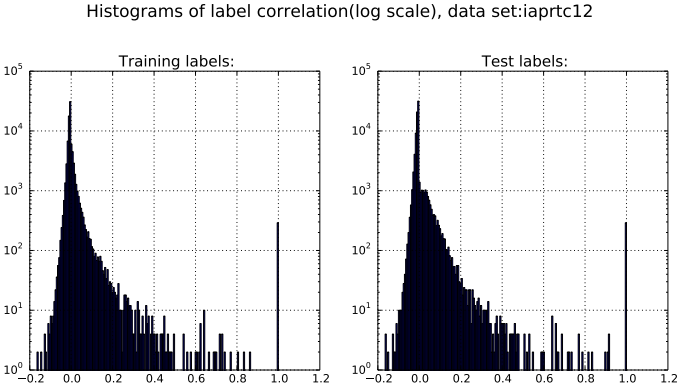
A novel joint SVM was presented for multi-label learning. One benefit of using joint SVM is that the learning and regularization of a linear output kernel are implicitly conducted. Moreover, both training joint SVM and predicting with joint SVM are efficient. As a possible work direction, we might investigate more interesting output kernel regularization schemes to fit different applications.



(a)



(b)



(c)

Figure 1: The histograms (in log scale) of all correlation values in both training sets and testing sets: (a) Corel5k, (b) Espgame (c) Iartc12.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] David M. Blei and Michael I. Jordan. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003.

[3] Minmin Chen, Alice Zheng, and Kilian Q. Weinberger. Fast image tagging. In *ICML*, 2013.

[4] Francesco Dinuzzo, Cheng Soon Ong, Peter V. Gehler, and Gianluigi Pillonetto. Learning output kernels with block coordinate descent. In *ICML*, 2011.

[5] S. L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *CVPR*, 2004.

[6] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[7] Bharath Hariharan, S. V. N. Vishwanathan, and Manik Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine Learning*, 88(1-2):127–155, 2012.

[8] Victor Lavrenko, R. Manmatha, and Jiwoon Jeon. A model for learning the semantics of pictures. In *NIPs*. 2004.

[9] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90:88–105, 2010.

[10] Sandor Szedmak and John Shawe-taylor. Learning via linear operators: Maximum margin regression. Technical report, University of Southampton, UK, 2005.

[11] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[12] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[13] Yu Zhang and Dit-Yan Yeung. Multilabel relationship learning. *ACM Trans. Knowl. Discov. Data*, 7(2):1–30, August 2013.

4.2 Homogeneity Analysis for Object-Action Relation Learning

VII. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Homogeneity Analysis for Object-Action Relations Reasoning in Kitchen Scenarios*, In Proceedings of 2nd Workshop on Machine Learning for Intelligent Systems (MLIS13), pp 37-44, 2013, ACM.

Homogeneity Analysis for Object-Action Relation Reasoning in Kitchen Scenarios *

Hanchen Xiong Sandor Szedmak Justus Piater
Institute of Computer Science, University of Innsbruck
Technikerstr.21a A-6020, Innsbruck, Austria
{hanchen.xiong,sandor.szedmak,justus.piater}@uibk.ac.at

ABSTRACT

Modeling and learning object-action relations has been an active topic of robotic study since it can enable an agent to discover manipulation knowledge from empirical data, based on which, for instance, the effects of different actions on an unseen object can be inferred in a data-driven way. This paper introduces a novel object-action relational model, in which objects are represented in a multi-layer, action-oriented space, and actions are represented in an object-oriented space. Model learning is based on homogeneity analysis, with extra dependency learning and decomposition of unique object scores into different action layers. The model is evaluated on a dataset of objects and actions in a kitchen scenario, and the experimental results illustrate that the proposed model yields semantically reasonable interpretation of object-action relations. The learned object-action relation model is also tested in various practical tasks (e.g. action effect prediction, object selection), and it displays high accuracy and robustness to noise and missing data.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Manipulation; I.2.6 [**Learning**]: Knowledge acquisition—*Object-action relation learning*

General Terms

Algorithms, Experimentation

Keywords

Homogeneity analysis, Object-action relation learning

1. INTRODUCTION

Manipulations of objects are core and indispensable functions in robotic systems to fulfill various practical tasks. However, because of the diversity of real-world objects in shape, material and other properties, manipulation design at the instance level is very effort-consuming and thus prohibitive. Learning principles or correlation patterns of different actions based on trial experiences is an appealing direction of robotics research. In other words, an agent can acquire knowledge of object-action relations in a data-driven manner by making use of a limited number of experiments. In addition, the study of object-action relations has also attracted attention within the cognition and psychology communities [5, 10], since it is expected to be related to how human beings accumulate knowledge by physically interacting with different objects. Humans begin to interact with their environment in their infancy, and in many interactions, two elements are involved: objects and actions. Actions are executed on objects with the humans' motor capabilities, and the effects of these actions are observed with their perception abilities. Based on such repeated interactions, human beings can quickly acquire object-action knowledge, and easily fulfill different actions on various objects by transferring such knowledge to novel objects. Although the exact mechanism of how the human brain organizes and learns object-action relations is still unknown, it has been pointed out that computational modeling of object-action relations can be a plausible perspective for the study of both robotics and human cognition.

Nevertheless, modeling and learning object-action relations has been a difficult task. The difficulties mainly stem from two sources. First, the structure of descriptions of both objects and actions can be very complex. The descriptions are derived from several sources, and the corresponding feature spaces are high-dimensional (i.e., objects and actions are characterized by large numbers of parameters). The second difficulty is due to the small number of experiments which can confirm the effects of different actions on objects. Even worse, in some cases, the experiments might provide contradicting outcomes. In consequence, the empirical data are rather sparse and noisy.

In this paper we put forward a novel model of object-action relations, in which objects are represented in a multi-layer action-oriented space, and actions are represented in an object-oriented space. The object-action relations are encoded in these two spaces, on which various reasoning tasks can be performed. The training data for the model are constructed from two sources, objects and the collection of effects (positive and negative) of different actions

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

MLIS'13, August 04 2013, Beijing, China

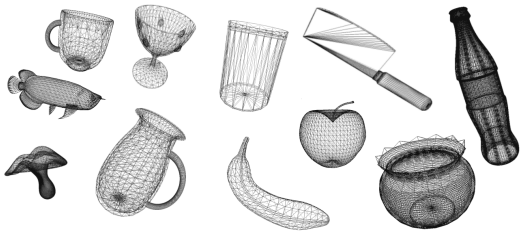


Figure 1: A sample set of kitchen objects

executed on objects. Two pieces of information are summarized in a structure called object-action profiles. Objects are represented by categorical indicators of basic properties and binary labels of various low- and high-level geometric features. Actions are represented by binary labels of object-dependent effects. The model is learned with *homogeneity analysis*. The strength of homogeneity analysis is that it can map multi-variate categorical/binary data to a homogeneous Euclidean space. Via these projections, objects and actions can be effectively represented with object scores and category quantifications. Basically, the object scores are computed as the average of quantifications of categories which they belong to, and category quantifications are computed as the geometric centroid of objects they belong to. These two projections are iteratively updated until convergence. Based on homogeneity analysis, action-category quantifications are represented in an object-oriented manner. The resulting object scores, however, do not fit our modeling scenario. The object scores are computed by treating all variables of object-action profiles equivalently, and therefore the scores are unique for all actions. By contrast, our model is designed to represent objects differently with respect to different actions. Therefore, we provide dedicated means to determine the dependencies between category quantifications of object and action variables, and decompose the object scores/representations into different action layers.

We present our model and associated learning/reasoning procedures in the context of object-action relations within a kitchen scenario (Figure 1). A database of typical kitchen objects and actions is constructed as well to evaluate our model. The experimental results demonstrate that the model yields semantically good interpretation of object-action relations by displaying reasonable dependencies and correlations between object and actions variables. In addition, the experiment with sparseness and noise added into training data highlights the robustness of our model to noisy and missing data.

1.1 Related Work

For object manipulation knowledge modelling, the concept of *affordance* [5] has been widely used [9, 8, 10, 3, 4] to link objects and actions in terms of object-action-effect triples. An affordance defines how an object "affords" a manipulation by an agent based on its motor abilities, and how this manipulability can be perceived by the agent [5]. For instance, the grasping affordance of a stone is much higher for a human being than a dog since human hands have better motor control of fingers than dogs' paws. More concretely, object affordances represent how can an agent interact with real-world environment by encoding the relations among actions, objects and effects. Although there have been numerous studies on how affordances can be modelled such that they can be effectively learned and utilized to assist practical robotic manipulation, the object/action affordance problem, at its base, is about how an agent can understand objects based on interactions with them by using its motor and perceptual capabilities. However, most previous studies are limited to one isolated object affordance (e.g. grasping). In some cases, multiple objects are involved and interact with each other within one manipulation. For example, a single action such as cutting involves two objects, the cutting tool (e.g. knives) and the object being cut (e.g. an apple). In [7], the affordance definition was extended to object relations. However, since only geometric relation (distance, angle of orientations) between multiple objects are used in [7], it still cannot model concepts such as cutting affordances for objects. Our work, by contrast, seeks to model general object-action relations. Our relational model connects objects and all possible actions that can be performed on them.

Our model is mainly inspired by [3], of which the basic assumption is that objects that share similar parts (e.g. rim, handles) should also hold similar grasping affordances. We extend [3] in two ways: first, we consider general object-action relations instead of only grasping affordances; second, the dependency of actions on different parts can be learned, in which way, for different actions, different co-occurring parts among objects will be considered for their action-effect reasoning.

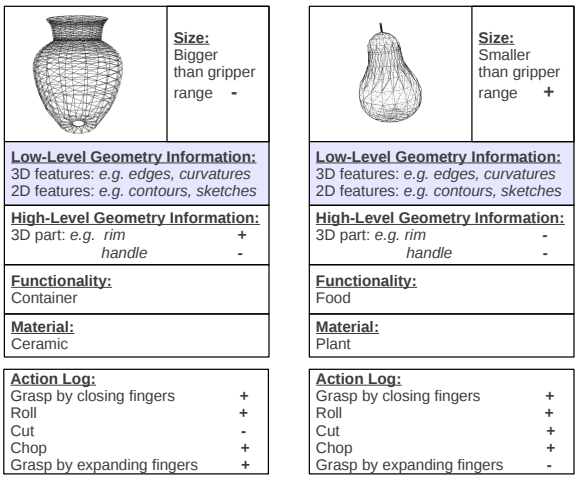
Other related work involves modeling of *sensorimotor coordination* [9], where a Bayesian network is employed to model multiple affordances associated with objects based on visual properties (e.g. color, size, concavity) and basic motor actions (grasping, touching, tapping). The dependences between actions, perception and effects are encoded in the directed edges within the Bayesian network. One shortcoming of this model is the dependency learning (i.e. the Bayesian network structure). Since in a Bayesian framework it is impractical to estimate the likelihoods of all possible dependency structures, Markov chain Monte Carlo (MCMC) sampling was used to approximate them. However, one practical problem with MCMC is that it can be quite inefficient (usually multiple chains are necessary); secondly, the approximation can be misleading when the training data is small in size, noisy and incomplete. By contrast, the dependency learning of our model is based on the category quantifications from homogeneity analysis, which is robust to noisy and missing data.

2. MODELING

In this section, two basic elements are explained for object-action relation modeling. First, we introduce a new data structure constructed from empirical object and action data (section 2.1). Secondly, section 2.2 presents an overview of the model structure (Figure 3), in which objects are represented in a multi-layered action-oriented space, and actions are likewise represented in an object-oriented space.

2.1 Data Structure

Since our objective is to learn the relations between objects and actions, the training data is constructed from two sources. One is the *object dataset*, in which basic properties (e.g. size, functionality, material) are labelled, and various low- and high-level geometric properties can be extracted by visual perception. The other source is the *action dataset*



(a)

O	Mesh	<Gripper	L_Geo		H_Geo		Func	Mate	Action log				
			3D	2D	rim	handle			Grasp_C	Roll	Cut	Chop	Grasp_E
1	file1	1			1	-1	1	1	1	-1	*	1	-1
2	file2	-1			-1	*	2	2	-1	*	1	1	*
3	file3	-1			1	1	2	5	*	1	*	1	1
4	file4	1			1	1	5	3	1	1	-1	*	1
5	file5	1			-1	-1	1	4	*	1		1	-1
6	file6	-1			1	1	4	6	1	-1	*	-1	1

Functionality	Container	Food	Cooker	Cutting tool	Eating tool
	1	2	3	4	5

Material	Plastic	Glass	Ceramic	Plant	Animal	Metal
	1	2	3	4	5	6

(b)

Figure 2: (a) Two examples of object-action profiles. (b) Collection of object-action profiles, * denoting missing data. Incompleteness (or sparseness) will always be a problem in training data.

that collects the effect of different actions applied on objects. In our study, the information from these two sources is merged into *object-action profiles*. Figure 2(a) presents two examples of object-action profiles. In the upper part, object shape is displayed. Some basic properties are labelled and geometric features are extracted. Because we are only concerned with the kitchen scenario, functionalities are limited to {container, food, cooker, cutting tool, eating tool}, and materials are limited to {plastic, glass, wood, plant, animal, metal}. For size, a binary indicator is used to check if it is smaller than the gripper's maximum range. In addition, low-level and high-level geometric features of objects can be detected or labelled (although we currently only use high-level geometric features such as rim, handle[1], because they are more informative of our actions than low-level features). In the lower part, the resulting effects of different actions on the object are recorded with binary values (+1 means successful and −1 otherwise). We consider some more-or-less common kitchen actions {grasping by closing fingers, rolling, cutting, grasping by expanding fingers, chopping}. It is worth noting that the strategies of feature labelling and action selection used in this paper are just one among many ways of describing the proposed model (section 2.2) and learning/reasoning procedure (section 3); they can be replaced by equivalent or more elaborate mechanisms. It should also be noted that in practice a very limited number of action experiments or simulations can be conducted on only a few objects, so incompleteness (or sparseness) of experimental data is a fact we have to deal with (Figure 2(b)).

2.2 Model Structure

In this paper, the object-action relations are modeled as shown in Figure 3. Actions and objects are represented in different spaces, that is, *action space* and *object space* respectively. The object space is composed of different layers that correspond to different actions. In each layer of the object space, the objects are linked pairwise (Figure 3), and the connection between a pair of objects is weighted proportionally to their similarity with respect to the corresponding action. The similarities between objects can be measured based on co-occurring properties or geometric features that can influence the outcome of the action. For instance, if object A (mug) and B (goblet) are both containers (therefore exhibit hollow structure), their similarities will be high in the "Grasp by expanding fingers" layer. However, their similarity would be low in the "roll" layer since A has a handle but B does not, and having a handle or not is a decisive factor for rolling.

In action space there is only one layer. Different actions are connected with each other, likewise with the connections weighted proportionally to their similarities. The similarities between actions can be interpreted as the similarities between their corresponding layers in object space.

[1]Such labels can be obtained by straightforward shape analysis systems.

3. MODEL LEARNING AND REASONING

With training data organized in the form of Figure 2(b), we straightforwardly apply homogeneity analysis [2, 6] to project all columns of Figure 2(b) to category quantifications and rows to object scores (section 3.1). However, the object scores computed by homogeneity analysis are the same for all actions, which does not fit our multi-layer object space (section 2.2). The underlying principle of our multi-layer object representations is that the dependencies between every action and object properties and geometric features are different; therefore, objects should be represented differently with respect to different actions. Meanwhile, the dependency and correlation relations between different basic properties, geometric features and actions are usually complicated. Two examples of such dependencies can be seen in Figure 4. It can be easily imagined that if a container is smaller than the gripper range in size, then it probably can

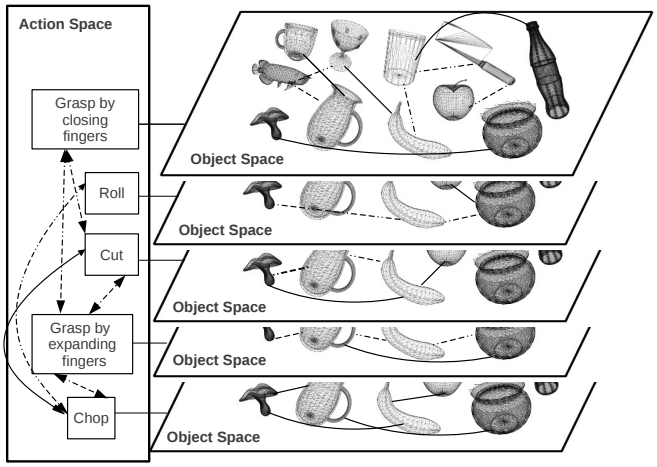


Figure 3: Object-action relational model. The object space is composed of action-specific layers, in which objects are interconnected (solid lines denote strong and dashed lines weak connections). There is only one layer in action space, and actions are connected in a similar way.

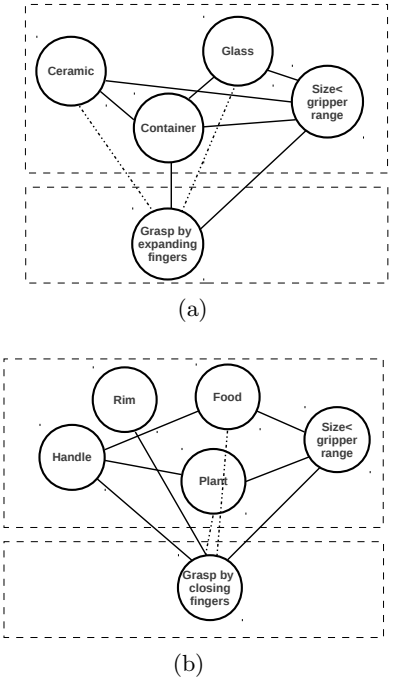


Figure 4: Two examples of dependencies between actions and objects' basic properties and geometry features: (a) grasp by expanding fingers; (b) grasp by closing fingers.

be graspable by expanding fingers, so there should be dependencies on "Container" and "<Gripper" for action "Grasp by expanding fingers"(Figure 4(a)). At the same time, containers smaller than the gripper are often made of ceramic or glass (e.g. bowls, mug, wineglass) in contrast to larger objects (e.g. plastic buckets or metal trash cans), so "Grasp by expanding fingers" might also be correlated with "Ceramic" and "Glass" (dashed lines). Similarly, usually an object is graspable if it is smaller than the gripper size or if it has a handle or rim, so it is reasonable to add dependencies between them (Figure 4(b)). Food items and plants are usually smaller than the gripper in a kitchen scenario, and they are unlikely to have handles. So extra dependencies on "Food" and "Plant" may be added as well. Instead of tediously reasoning about the dependencies for all actions, in section 3.2 a dependency checking mechanism is provided to remove unlikely or weak dependencies. The computed dependencies are also utilized to remap objects to different action layers with dependency weights.

3.1 Initial Learning with Homogeneity Analysis

Homogeneity analysis [2, 6] is a popular statistical tool for categorical multivariate analysis. Here we briefly review the procedure of homogeneity analysis with its application to object-action profile data. There are M object-action profiles in the dataset, each profile represented by a J-dimensional vector $O_i = [v_1, v_2, \ldots, v_J]^\top$ $(i = 1, \ldots, M)$, with each variable v_j denoting an attribute in the profile. Variable v_j takes on n_j categorical values (e.g., the action effect has binary values ± 1). By gathering the values of v_j over all M profiles in an $M \times n_j$ binary indicator matrix G_j, the whole set of indicator matrices can be gathered in a block matrix:

$$G = [G_1|G_2|\cdots|G_J] \tag{1}$$

The key feature of homogeneity analysis is that it simultaneously produces two projections to the same Euclidean space $\mathbb{R}^p$, one from J-dimensional profiles O_i, the other from the M-dimensional categorical attribute indicator vectors (columns of G). These projections are referred to as *object score* and *category quantification*, respectively [2, 6]. Suppose the collection of object scores is represented by an $M \times p$ matrix X, and category quantifications for variable v_j are represented by a $n_j \times p$ matrix Y_j. Then, the cost function of a projection can be formulated as:

$$f(X, Y_1, \cdots, Y_J) = \frac{1}{J}\sum_{j=1}^{J} \mathbf{tr}(X - G_j Y_j)^\top (X - G_j Y_j) \tag{2}$$

As emphasized above, in realistic cases the training dataset is usually sparse and incomplete, i.e., values of some v_j are missing. So for each G_j, we construct an $M \times M$ diagonal matrix S_j with diagonal values equal the sum of the rows of G_j, i.e., $S_j(i,i) = 0$ if the v_j value of O_i is missing. Then the corresponding cost function is

$$f(X, Y_1, \cdots, Y_J) = \frac{1}{J}\sum_{j=1}^{J} \mathbf{tr}(X - G_j Y_j)^\top S_j (X - G_j Y_j) \tag{3}$$

Usually two extra constraints are added to avoid trivial solution $(X = 0, Y_j = 0)$:

$$\frac{1}{M}\mathbf{1}_{M\times 1}^\top S_* X = \mathbf{0} \tag{4}$$

$$\frac{1}{M} X^\top S_* X = I \tag{5}$$

Here, $S_* = \sum_{j=1}^{J} S_j$. The first constraint (4) essentially nor-

malizes the projected object scores to be centered around the origin. The second restriction (5) standardizes all p dimensions of object score by rescaling the square length of each dimension to M. In addition, another effect of (5) is that the p columns of X are imposed to be orthogonal to each other.

To minimize the cost function (3) under these constraints (4, 5), usually the alternating least squares (ALS) algorithm [2, 6] is used. The basic idea of ALS is to iteratively optimize with respect to X or to $[Y_1, \cdots, Y_M]$ with the other held fixed. Assuming $X^{(0)}$ is provided arbitrarily at iteration $t = 0$, each iteration of ALS can be summarized as:

1. update Y_j:

$$Y_j^{(t)} = (G_j^\top S_j G_j)^{-1} G_j^\top X^{(t)}; \tag{6}$$

2. update X:

$$X^{(t+1)} = S_*^{-1} \sum_{j=1}^{J} G_j Y_j^t; \tag{7}$$

3. normalize X:

$$X^{(t+1)} = \textit{Gram-Schmidt}(X^{(t+1)}). \tag{8}$$

It can be seen (6) that category quantification of Y_j is computed as the centroid of the object scores that belong to it. Step 2 (7) updates object scores X by taking the average of the quantifications of the categories it belongs to. In step 3 (8) a *Gram-Schmidt* procedure is used to find the normalized and orthogonal basis of updated object scores from the previous step.

3.2 Dependency Learning

According to the description in the previous section, the objects and action effects can be projected into two spaces (object scores X and category quantifications Y_j of action variables v_j) by applying homogeneity analysis on the set of object-action profiles. Although this observation is close to how we model object-action relations (section 2.2), there still exist some obstacles that prevent us from directly putting them to practical use. First, by using homogeneity analysis, basic properties, geometric features and action effects are simultaneously projected to their corresponding category quantifications without modelling their interrelations explicitly. As we illustrated in Figure 4, the dependency between them is an important factor in our object-action relational model, so we must disentangle how each action depends on different basic properties and geometric features. Secondly, in our model the objects are represented at different layers corresponding to different actions, while the representations of objects with homogeneity analysis are unique object scores. Hence, it is also required to strategically decompose the object scores into different action layers.

To resolve these two problems, some extra steps can be developed to exploit more information from the object scores and category quantifications. First, the J variables $[v_1, v_2, \ldots, v_J]$ of each object O_i are divided into two groups, the object (variable) group V_o which covers basic properties and geometry features, and the action (variable) group V_a which contains action effects on the object O_i. We initially assume that each variable in action group $v_\beta^a \in V_a$ depends on all variables of the object group V_o. Then, for variable v_β^a, we find its corresponding positive and negative category quantifications $Y_{\beta,+}^a$ and $Y_{\beta,-}^a$, and compute the distances between them and all categories' quantifications in the object group as

$$d(Y_{\beta,+/-}^a, Y_{\omega,k}^o) = ||Y_{\beta,+/-}^a - Y_{\omega,k}^o||_2 \tag{9}$$

where $Y_{k,w}^o$ denotes the k-th category quantification of variable v_ω^o in the object group. We compute the maximum ratio between them as

$$\lambda_{\omega,k}^\beta = \max\left\{\frac{d(Y_{\beta,+}^a, Y_{\omega,k}^o)}{d(Y_{\beta,-}^a, Y_{\omega,k}^o)}, \frac{d(Y_{\beta,-}^a, Y_{\omega,k}^o)}{d(Y_{\beta,+}^a, Y_{\omega,k}^o)}\right\} \tag{10}$$

and eliminate the dependencies between action variable v_β^a and category quantifications in V_0 if

$$\frac{\lambda_{\omega,k}^\beta}{\sum_{\omega,k} \lambda_{\omega,k}^\beta} < \sigma \tag{11}$$

where $\sigma \in [0, 1]$ is a predefined threshold. The elimination criterion (11) is defined based on the concept that the object variables on which the action variable depends should have good discriminative abilities between its positive and negative categories.

Once the dependencies have been found, the second problem can be solved as well. Instead of computing object scores as the average of the all quantifications of the categories they belong to (7), the representations of objects in each action layer β are computed as the weighted average of quantifications of the (positive and negative) action categories and the category quantifications in V_o which the action is dependent on:

$$X_\beta = \hat{S}_{*,\omega,k}^{-1} \sum_{\omega,k \in \text{dependent}(\beta)} \pi_{\omega,k} \hat{G}_{\omega,k} \hat{Y}_{\omega,k} \tag{12}$$

where the $\hat{Y}_{\omega,k}$ are the category quantifications (out of n_ω) of variable v_ω^o on which action variable v_β^a depends. $\hat{G}_{w,k}$, $\hat{S}_*$ are the corresponding indicator matrix and diagonal matrix. $\pi_{\omega,k}$ denotes the normalized dependency weights which reflect how β depends on quantifications in $\hat{Y}_{\omega,k}$:

$$\pi_{\omega,k} = \frac{\lambda_{\omega,k}^\beta}{\sum_{\omega,k \in \text{dependent}(\beta)} \lambda_{\omega,k}^\beta} \tag{13}$$

Correspondingly, the centroid of object representations which belongs to positive and negative category in β action layer is:

$$\beta_c^{+/-} = (G_{\beta,+/-}^\top S_{\beta,+/-} G_{\beta,+/-})^{-1} G_{\beta,+/-}^\top X_\beta \tag{14}$$

where $G_{\beta,+/-}$ is the positive/negative-category column in G_β and $S_{\beta,E}$ is corresponding diagonal counting matrix.

The dependencies between action variables can be also similarly learned to find the correlation or anti-correlation between object effects. Since our model is dedicated to relations between objects and actions, action-action relations will be exploited in our future work.

3.3 Reasoning

Given the object-action relational model learned with the procedure above, typical reasoning tasks are presented in Table 1. First, we discuss effect (E) prediction given object (O) and action (β). Assume O is an unseen object. Its representation in action layer β can be computed (12), and then

input	output	applications
object & action	effect	effect outcome prediction
action & effect	object	object selection
object & effect	action	action planing/recognition

Table 1: Typical applications of the object-action relation model.

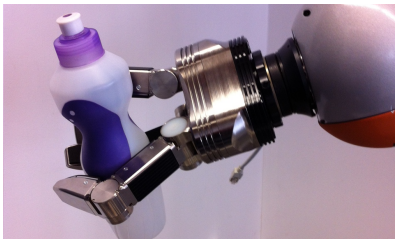


Figure 5: Robot hand used for action labelling

the binary effect classification can be easily done by majority voting of the k-nearest neighbouring objects of training set (or using any other suitable classifier).

Second, the model can perform object (O) selection out of a set of candidates $\mathbf{C}$ based on action (β) and effect ($E \in [-1, 1]$). Given the desired category E of action β, first object representations in candidate set $X_\beta^{(O \in \mathbf{C})}$ can be computed (12). Then the ratio of the distance between each $X_\beta^{(O)}$ and β_c^E to the distance between $X_\beta^{(O)}$ and β_c^{-E} (14) can be computed:

$$\phi_O = \frac{d(X_\beta^{(O)}, \beta_c^E)}{d(X_\beta^{(O)}, \beta_c^{-E})} \tag{15}$$

The optimal object $O^\dagger$ is the one with smallest ϕ_O. Alternatively, with the ratios of all objects in $\mathbf{C}$ computed, the object retrieval result can be ranked by their ratios in increasing order.

Finally, action selection or planning is also useful to find an optimal action among many that share similar semantic effects based on certain criteria. For example, both cutting and chopping are actions that break objects into smaller parts. However, they are executed with different tools (cleavers for chopping and knives for cutting) and with different strength. So if the task is to break an object O into parts with minimum strength from the higher-level planner, then one may want to perform a chopping action only if necessary. To this end, we compute the representation of O in cutting and chopping layers respectively and predict their corresponding effects, based on which the most energy-saving action will be selected.

4. EXPERIMENTS

4.1 Synthetic Database and Model Learning

To evaluate the proposed object-action relational model and learning method, we constructed a synthetic dataset of object-action profiles. We collected 140 kitchen objects (Figure 1) from the web [1] and annotated them as shown in Figure 2. The labeling and actions are set in the same way as described in section 2.1. Basic properties and high-level geometry features[2] of objects were labelled by a student volunteer. The effects of different actions applied on objects are labeled as well based on common sense[3]. The robot gripper is presented to the labeller (Figure 5) for the consideration of different actions.

First, the model is learned with full and noisy-free data. By applying homogeneity analysis as described in section 3.1, we obtain 3-dimensional category quantifications of 10 variables in object-action profiles (Figure 6). With extra maximum ratio computation (10) (Figure 7), the dependency between each action and objects' basic properties and geometric features are discovered (Table 2). Table 2 shows that "grasp by expanding fingers" and "grasp by closing fingers" exactly match our previous dependency analysis in Figure 4, i.e. the proposed model yields semantically reasonable object-action relations.

4.2 Reasoning Tasks

To quantitatively evaluate the proposed model, the following experiments test the model on two reasoning tasks, effect prediction and object selection[4]. In both experiments, the 140 object-action profiles are randomly divided into training set (100) and test set (40). In addition, as we already pointed out, in practice the empirical object-action data can be noisy and incomplete because of inaccuracy of perception systems and lack of real (or simulated) experiments. Therefore, to test the robustness of the model to noise and missing data, 10% noise are added and 20% entries are removed from the 100 training instances. The noise is generated by shifting the labels of variables with probability 0.1, and entries in Figure 2 are removed with probability 0.2.

Effect Prediction

According to the reasoning procedure described in section 3.3, 40 test objects are first projected to different representations at different action layers. Then the final effects of actions are decided by using a simple k-nearest-neighbour (KNN) classifier with the 100 representations of training objects. We use $k = 10$ for both full-data and missing-and-noisy-data conditions. We ran 50 trials in which different size-100 training (both full and missing-and-noisy) and size-40 test data sets are randomly generated. The average precision of correct effect classification of five actions are presented in Figure 8(a), from which it can be seen that the prediction results with both full training data and missing-and-noisy data are rather accurate, with the former slightly outperforming the latter (as is to be expected).

Object Selection

The object selection experiment is set up to test how accurate an object can be "recommended" to meet the effect of an action. The reasoning is based on the procedure in section 3.3, and the recommendation is ranked based on ratios (15). Similarly to the effect-prediction experiment, 50 trials with different training and test data are run, and the average results of 5 actions (positive and negative) are pre-

[2]We did not use low-level geometric features in our experiments.

[3]In future work, we plan to use simulated and ultimately physical robotic action.

[4]Since action selection applications usually require higher-level planners to handle constraints, robustness criteria etc., we did not consider them in our pilot experiments.

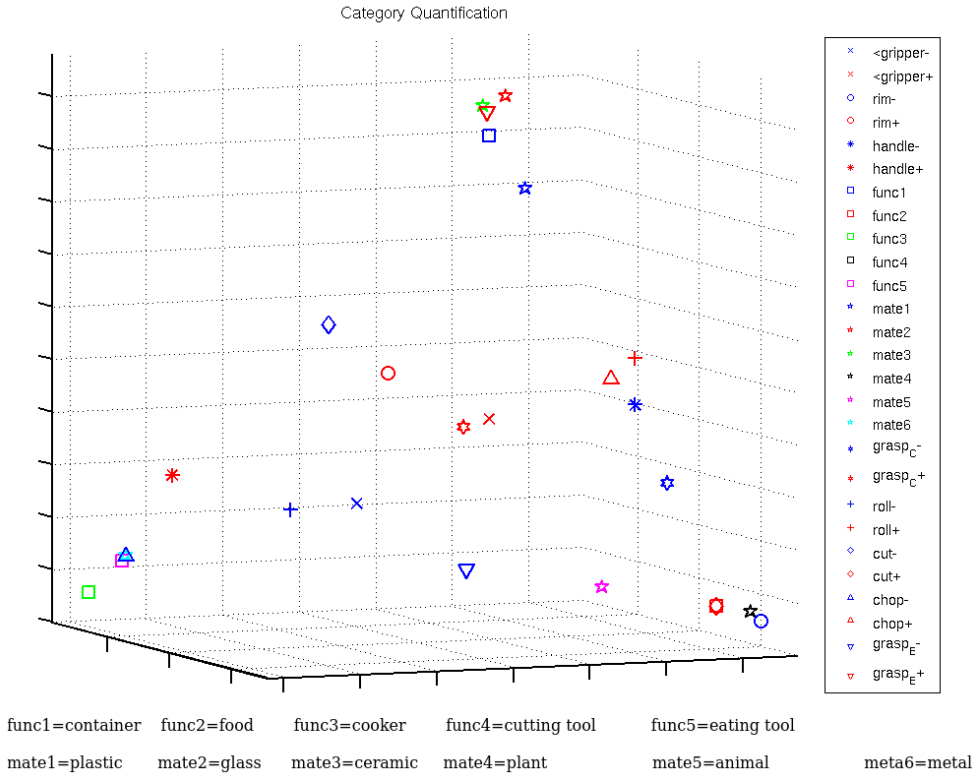
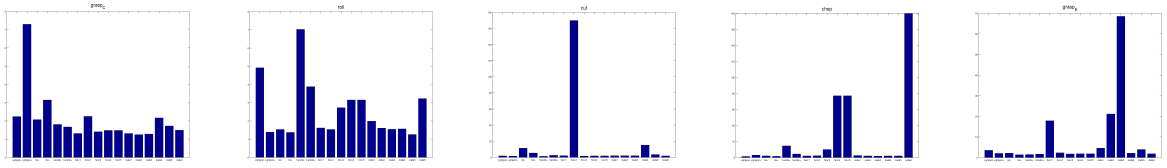


Figure 6: Category quantifications of variables in object-action profiles (best viewed in color).



(a) (b) (c) (d) (e)

Figure 7: Check the dependency of five actions ((a) grasp by closing fingers (b) roll (c) cut (d) chop (e) grasp by expanding fingers) on category quantifications of object variables (from left to right bars denotes the maximum ratios (10) of <gripper-, <gripper+, rim-,rim+, handle-, handle+, container, food, cooker, cutting tool, eating tool, plastic, glass, ceramic, plant, animal, metal).

V_a	**Depended category quantification of variable in V_o**
$grasp_C$	<gripper-, <gripper+, handle-, rim-, rim+,function=food, material=plant
roll	<gripper-, handle-, handle+, function=cooker,function=cutting tool, function=eating tool, material=metal
cut	function=food, material=plant
chop	function=cutting tool, function=eating tool, material=metal
$grasp_E$	function=container, material=glass, material=ceramic

Table 2: Dependency of five actions on category quantifications of object variables after elimination (11).

sented in Figure 8(b)-8(f) with precision-recall curves. It can be seen that except for the poor results on grasping by closing fingers, object retrieval of all other actions and effects are acceptable. The reason for poor performance in the negative case of grasping by closing fingers, according to our preliminary analysis, is that there are are too few instances of $grasp_C-$ in the training data; most objects in the kitchen are graspable. The results with missing-and-noisy training data are slightly inferior to those with full training data. Two obvious performance gaps appear in the negative case of chopping, and in the positive case of grasping by expanding fingers. In conclusion, both effect prediction and

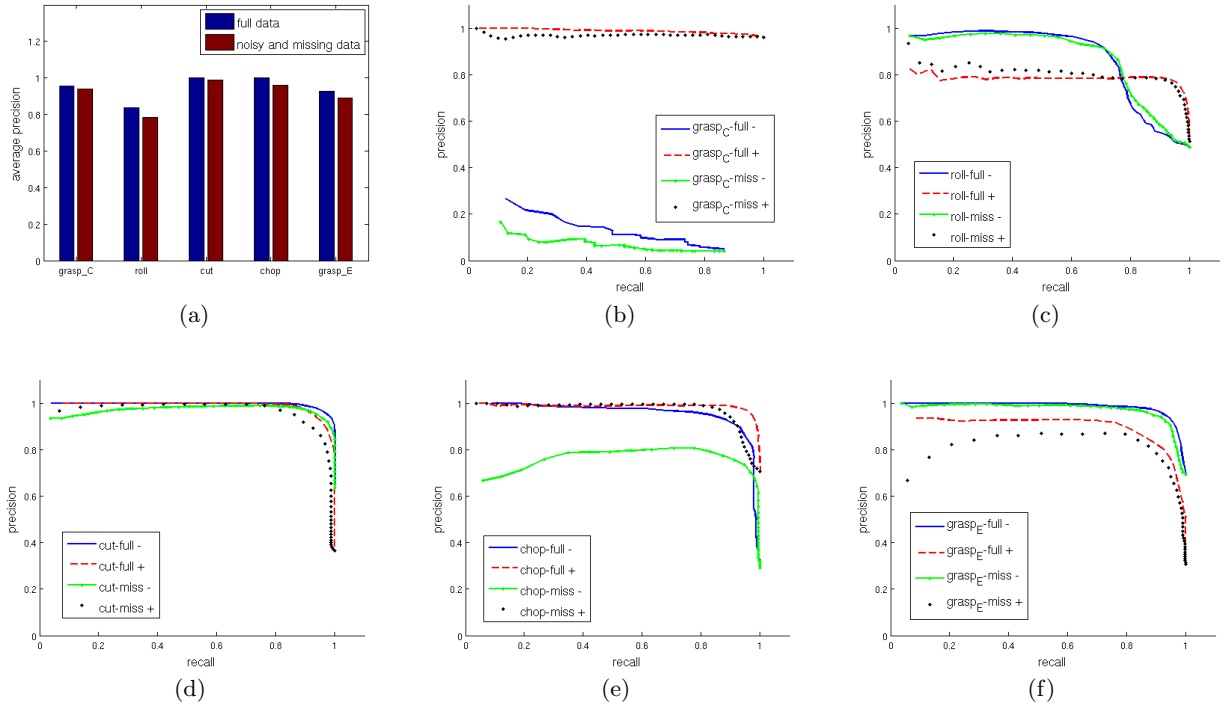


Figure 8: (a) The average precision of correct effect prediction of five actions; (b)-(f) the precision-recall curves of average object selection results in all positive and negative of five actions.

object selection experiments quantitatively demonstrate the promising capabilities of our object-action relational model by displaying its high accuracies and robustness to noisy and incomplete data.

5. CONCLUSION

We presented a novel computational model of object-action relations. Actions are represented in terms of their effects on objects, and objects are represented as well in an action-oriented manner. The model can be effectively learned with homogeneity analysis and extra discovery of dependencies between action and object variables. One strength of the proposed model is that it does not require complex, highly-combinatorial descriptions of objects and actions. The object representations with respect to different actions are computed with only a small number of the most decisive object variables. Actions are presented by their positive and negative action-effect category quantifications. Another merit of the model, according to experimental results, is that it is robust to noisy and missing data, which is an unavoidable problem in practice.

6. REFERENCES

[1] www-roc.inria.fr/gamma/download/.

[2] J. de Leeuw and P. Mair. Homogeneity Analysis in R: The Package homals. . Technical report, Department of Statistics, UCLA, 2007.

[3] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragić. Generalizing Grasps Across Partly Similar Objects. In *International Conference on Robotics and Automation*, pages 3791–3797. IEEE, 2012.

[4] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning Grasp Affordance Densities. *Paladyn Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[5] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[6] G. Michailidis and J. de Leeuw. The Gifi System of Descriptive Multivariate Analysis. *Statistical Science*, 13:307–336, 1998.

[7] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *IEEE International Conference on Robotics and Automation, ICRA 2012,*, pages 4373 –4378, May 2012.

[8] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *IEEE 8TH International Conference on Development and Learning*, China, 2009.

[9] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, Feb 2008.

[10] E. Oztop, N. Bradley, and M. Arbib. Infant grasp learning: a computational model. *Experimental Brain Research*, 158(4):480–503, 2004.

4.3 Multi-Label Learning with Kernel Generalized Homogeneity Analysis

X. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Multi-Label Learning with Kernel Generalized Homogeneity Analysis*, Unpublished, 2015.

Multi-Label Learning with Kernel Generalized Homogeneity Analysis

Hanchen Xiong Sandor Szedmak Justus Piater

{*hanchen.xiong,sandor.szedmak,justus.piater*}*@uibk.ac.at*

Institute of Computer Science, University of Innsbruck

Abstract

Canonical correlation analysis (CCA) and homogeneity analysis (HA) are two popular methods for analyzing multivariate data. Although they are applied on different data types – the former is used on two sets of variables while the latter operates on multivariate categorical variables – we reveal that they are actually closely related. Building on this relation, we generalize HA to handle continuous variables, which leads to a relaxed variant of multiple-set CCA. Furthermore, kernel functions are also utilized to enable generalized HA to learn nonlinear dependencies within data.

In present paper, we in particular investigate how kernel generalized HA (KGHA) can be applied to multi-label learning. We found that, for vector-valued functions, KGHA works as a learning method consisting of two advantageous components: low-rank output kernel learning and co-regularized multi-view learning. Low-rank output kernel learning coincides with lower-dimensional latent label space discovery, while co-regularized multi-view learning is related to multiple kernel learning for heterogeneous information fusion. Furthermore, a large-scale KGHA learning scheme is developed by employing a block-wise Nyström approximation. We evaluate KGHA on two multi-label classification applications, image annotation and protein function prediction. Our experimental results on several benchmark databases demonstrate that KGHA compares favorably to other state-of-the-art methods.

1 Introduction

The study of embedding complex data into a lower-dimensional space is an important task in machine learning. Relevant methods include *principal component analysis* (PCA), *canonical correlation analysis* (CCA), *homogeneity analysis* (HA, also known as multiple correspondence analysis), to name just a few. These methods are generally known as *multivariate analysis* (MVA; Izenman 2008). Originally, MVA methods were proposed with linear projections to satisfy different objective functions, in supervised or unsupervised contexts. For instance, the objective of PCA is to maximize the variances of linear projections of data onto a small number of principal bases. CCA seeks two lower-dimensional coordinate frames in which two sets of variables (*e.g.* input and output) are maximally correlated. HA, by contrast, operates on multivariate categorical data, and outcomes are a set of linear projections which can map both data instances

and categorical values to a low-dimensional space such that their consistency is preserved as much as possible. Although these methods have been successfully employed in various application domains, detecting linear patterns within data is rather limited in the face of increasingly more complicated data. Therefore, some nonlinear embedding techniques, *e.g. kernelized versions* of the above-mentioned MVA methods or *nonlinear manifold learning* methods Ma & Fu (2011), are increasingly used in modern data analysis.

We start with introductions to CCA and HA (Section 2), in which their corresponding objective functions, constraints, solutions and properties are explained. Similar to CCA, in a supervised-learning context, HA can be employed by considering one set of variables as outputs and the remaining sets as input features from heterogeneous information sources. Then, in section 3, by reformulating CCA on J sets of variables with $J > 2$, we arrive at an objective function of a form identical to HA. Building on this relationship, we generalize homogeneity analysis to handle continuous variables, which leads to a relaxed variant of multiple-set CCA. Furthermore, in section 4, we add a trade-off parameter to fit supervised-learning scenarios and kernel functions to enable generalized HA for learning nonlinear patterns. We refer to this novel HA as kernel generalized HA (KGHA). Similarly to regular HA, KGHA is trained via alternating least squares (ALS), which is more efficient than the multiple pair-wise eigenvalue computation in multiple-set CCA.

In section 5 we study KGHA in the multi-label learning case. We show that when used for learning vector-valued functions (*e.g.* multi-label, multi-task learning), KGHA is an elegant combination of low-rank output kernel learning and co-regularized multi-view learning. Low-rank output kernel learning coincides with multi-label dimensionality reduction Ye et al. (2011), which enables learners to gain higher efficiency and accuracy Ji & Ye (2009) by exploiting more compact yet informative latent space. Also, co-regularized multi-view learning is related to multiple kernel learning (MKL; Bucak et al. 2014), in which heterogeneous information is encoded in an ensemble of kernels to match outputs. One feature worth noting is that, since multi-label is encoded in a lower-dimensional latent space, co-regularization in KGHA takes place in a subsapce of multi-view, which differs from conventional co-regularization Rosenberg & Bartlett (2007).

This paper makes four contributions. First, we reveal the close connections between HA and multiple-set CCA, which sheds light on new understanding and potential extensions of these two MVA techniques. Second, we propose a novel multi-label learning method, KGHA, which is composed of two advantageous components, low-rank output kernel learning and co-regularized multi-view learning. Third, we develop a large-scale learning scheme for KGHA by employing a block-wise Nyström method for approximating kernel matrices and conjugate gradient for solving ALS. Finally, according to our experimental results in image annotation and protein function prediction tasks, KGHA can improve performance on several benchmark databases.

1.1 Related Work

Our study can be connected to many other work in different respects. The following gives a short summary of recent advances of relevant research.

Variants of CCA. It has been shown that CCA is related to other MVA techniques, such as partial least square (PLS) Sun et al. (2009) and Fisher linear discriminative analysis Sun et al. (2011). More extensions of CCA for multi-label learning can be found in Hardoon et al. Hardoon et al. (2004) and Sun et al. Sun et al. (2011).

Multi-Label Prediction. Basically, multi-label learning has been studied with different "canonical" learning schemes , e.g. regression Hsu et al. (2009); Lin et al. (2014) and ranking Elisseeff & Weston (2002). Recently, structured output learning has also been leveraged Hariharan et al. (2010); Xiong et al. (2014) for this study. Our method belongs to the regression category. **Multi-Label Dimensionality Reduction.** Much effort has been put into learning a shared subspace for multi-label outputs (see a review by Ye et al. 2011). Other notable work includes projection via compressed sensing Hsu et al. (2009) and feature-aware label encoding Lin et al. (2014). **MKL**. MKL has been recruited as a framework for integrating multiple input features from heterogeneous information sources Wang et al. (2008). Especially in computer vision and bioinformatics applications Bucak et al. (2014); Mostafavi & Morris (2010), since various visual features and bio-related features are available, MKL plays an important role in manipulating geometric structures of data in multiple features to fit certain applications. **Co-regularization for Multi-View Learning.** Co-regularization has been well investigated in multi-view learning Rosenberg & Bartlett (2007); Sridharan & Kakade (2008), however, these studies focus on semi-supervised learning. A similar subspace co-regularization in supervised-learning circumstance was proposed in Guo & Xiao Guo & Xiao (2012), where nevertheless only two views are considered.

Two pieces of work closely related to KGHA are FaIE Lin et al. (2014) and MultiK-MHKS Wang et al. (2008) respectively. First, in FaIE, lower-dimensional projections of multi-label data are found by jointly optimizing the correlations between input features and projections and recoverability of projections back to the original output data. From a different perspective, KGHA can be formulated as an objective function rather similar to FaIE. KGHA goes beyond FaIE by considering multiple features from heterogeneous sources of information. Secondly, in MultiK-MHKS, an extra regularization is used to encourage consensus among predictions from multiple kernels, which is identical to our co-regularized multi-view learning. Our work differs from MultiK-MHKS in that we learn multiple kernels in a non-binary subspace, and thus use least-squares loss instead of misclassification loss. In this sense, KGHA can be considered a combination of FaIE and MultiK-MHKS.

2 Preliminaries

2.1 Canonical Correlation Analysis

Canonical correlation analysis (CCA) Hardoon et al. (2004) was developed to find the correlations between two sets of variables. The essence of CCA is to seek a pair of linear transformations, one for each set, such that the correlation of transformed variables is maximized. Assume that a data instance is composed of two set of variates, $[g_1^\top, g_2^\top]$, of which the dimensions are d_1 and d_2 respectively. A dataset $\mathcal{D}$ consisting of M such instances can be represented as a $M \times (d_1 + d_2)$ matrix of the form $\mathcal{D} = [G_1, G_2]$. By using two matrices $\mathbf{w}_1 \in \mathbb{R}^{d_1 \times p}$ and $\mathbf{w}_2 \in \mathbb{R}^{d_2 \times p}$ with $p < \min(d_1, d_2)$, we can project the data into a lower, p-dimensional space:

$$\hat{\mathcal{D}} = [G_1\mathbf{w}_1, G_2\mathbf{w}_2] \tag{1}$$

Assuming the original data are already centered, $\hat{\mathcal{D}}$ will be centered as well, and the covariance of $\hat{\mathcal{D}}$ is $\mathbf{w}_1 G_1 G_2 \mathbf{w}_2$. The objective of CCA is to select $\mathbf{w}_1$ and $\mathbf{w}_2$ to

maximize the correlation between $G_1\mathbf{w}_1$ and $G_2\mathbf{w}_2$:

$$\begin{aligned}\{\mathbf{w}_1^*, \mathbf{w}_2^*\} &= \underset{\mathbf{w}_1,\mathbf{w}_2}{\operatorname{argmax}} \frac{\mathbf{w}_1^\top G_1^\top G_2 \mathbf{w}_2}{||\mathbf{w}_1 G_1||||\mathbf{w}_2 G_2||} \\ &= \underset{\mathbf{w}_1,\mathbf{w}_2}{\operatorname{argmax}} \frac{\mathbf{w}_1^\top C_{12} \mathbf{w}_2}{\sqrt{\mathbf{w}_1^\top C_{11} \mathbf{w}_1 \mathbf{w}_2^\top C_{22} \mathbf{w}_2}}\end{aligned} \tag{2}$$

where C_{12}, C_{11}, C_{22} are blocks within the covariance matrices of $\mathcal{D}$:

$$\operatorname{cov}(\mathcal{D}) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{3}$$

(2) can be rewritten as

$$\begin{aligned}\{\mathbf{w}_1^*, \mathbf{w}_2^*\} &= \underset{\mathbf{w}_1,\mathbf{w}_2}{\operatorname{argmax}} \mathbf{w}_1^\top C_{12} \mathbf{w}_2 \\ s.t. \quad & \mathbf{w}_1^\top C_{11} \mathbf{w}_1 = 1,\ \mathbf{w}_2^\top C_{22} \mathbf{w}_2 = 1\end{aligned} \tag{4}$$

It has been shown Bie et al. (2005); Hardoon et al. (2004) that the solution to (4) can be obtained by solving following generalized eigenvalue problem:

$$\begin{pmatrix} \mathbf{0} & C_{12} \\ C_{21} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix} = \lambda \begin{pmatrix} C_{11} & \mathbf{0} \\ \mathbf{0} & C_{22} \end{pmatrix} \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix} \tag{5}$$

There can be many solutions for (5), which correspond to different eigenvectors. One important property of different solution pairs (*e.g.* when we have p solution pairs $\mathbf{W}_1 = [\mathbf{w}_1^1, \ldots, \mathbf{w}_1^p]$, $\mathbf{W}_2 = [\mathbf{w}_2^1, \ldots, \mathbf{w}_2^p]$) is that the projections onto different $\mathbf{w}_{i=1,2}^{k\in[1,p]}$ are uncorrelated to each other:

$$\begin{aligned}\forall i = 1, 2, \quad & \mathbf{W}_i^\top C_{ii} \mathbf{W}_i = I_p \\ \forall k \neq h, \quad & \mathbf{w}_1^{k\top} C_{12} \mathbf{w}_2^h = 0\end{aligned} \tag{6}$$

It was also shown that the solutions of (5) $\mathbf{w}_1, \mathbf{w}_2$ lie in the span of G_1 and G_2 respectively, *i.e.* $\mathbf{w}_1 = G_1^\top \boldsymbol{\alpha}_1$, $\mathbf{w}_2 = G_2^\top \boldsymbol{\alpha}_2$, $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathbb{R}^M$. By substituting the alternative form of $\mathbf{w}_1, \mathbf{w}_2$ into the primal form of CCA (5), we can write out the dual form of CCA Bie et al. (2005); Hardoon et al. (2004) as

$$\begin{pmatrix} \mathbf{0} & K_1 K_2 \\ K_2 K_1 & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{pmatrix} = \lambda \begin{pmatrix} K_1^2 & \mathbf{0} \\ \mathbf{0} & K_2^2 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{pmatrix} \tag{7}$$

where $K \in \mathbb{R}^{M\times M}$ is the *Gram matrix* of the data, *i.e.* $K_{i=1,2} = G_i G_i^\top$. Therefore, by comparing (5) and (7), we can see that when $M < d_1 + d_2$, the dual form can be used to accelerate computing. The value of the dual form is even more significant when the *kernel method* is used on the data and the Gram matrix is replaced with a kernel matrix:

$$\mathcal{K}(G_i^{(m)}, G_i^{(n)}) = \left\langle \phi_i(G_i^{(m)}), \phi_i(G_i^{(n)}) \right\rangle \tag{8}$$

where $\phi_i : \mathbb{R}^{d_i} \to \mathcal{H}$ is a feature map from original data space to a reproducing kernel Hilbert space (RKHS). Kernel methods are of great help in detecting nonlinear patterns within the data.

2.2 Homogeneity Analysis

Homogeneity analysis Michailidis & de Leeuw (1998) is a popular tool for analyzing and visualizing multivariate categorical data. Assume that there are M data instances

in a dataset $\mathcal{D} = \{O_m\}_{m=1}^{M}$, and each data instance is represented by a J-dimensional vector $O_m = [v_1, v_2, \ldots, v_J]^\top$ $(m = 1, \ldots, M)$. Variable v_j takes on n_j categorical values. Here we briefly review the procedure of homogeneity analysis with its application to this simple dataset. Since the data is represented in a categorical space, we need to convert them to a vector space. To this end, we list n_j categorical values of v_j over all M data instances into an $M \times n_j$ binary indicator matrix G_j. The set of indicator matrices can be gathered in a block matrix

$$G = [G_1|G_2|\cdots|G_J]. \tag{9}$$

The key feature of homogeneity analysis is that it simultaneously produces two projections into the same Euclidean space $\mathbb{R}^p$, one from J-dimensional data instances O_i, the other from the M-dimensional categorical attribute indicator vectors (columns of G). These projections are referred to as *object scores* and *category quantifications*, respectively Michailidis & de Leeuw (1998). In addition, these two projections are intended to preserve the consistency among data instances and attribute values as closely as possible to the data in the original categorical space:

- data instances that exhibit similar attribute values are located closely together;
- data instances are close to their attribute category values.

Suppose that the collection of data instances is represented by an $M \times p$ matrix X, and category quantifications for variable v_j are represented by a $n_j \times p$ matrix Y_j. Then, the cost function of projections can be formulated as:

$$f(X, Y_1, \ldots, Y_J) = \frac{1}{J} \sum_{j=1}^{J} ||X - G_j Y_j||_F^2 \tag{10}$$

where $||\cdot||_F$ denotes the Frobenius norm. Two extra constraints are added to avoid the trivial solution ($X = \mathbf{0}, \forall j \in [1, J] \quad Y_j = \mathbf{0}$):

$$\mathbf{1}_{M\times 1}^\top X = \mathbf{0} \tag{11}$$

$$X^\top X = I_p \tag{12}$$

The first constraint (11) essentially normalizes the projected object scores to be centered around the origin. The second restriction (12) standardizes all p dimensions of the object score by rescaling the square length of each dimension to M. In addition, another effect of (12) is that the p columns of X are imposed to be orthogonal to each other.

To minimize the cost function (10) under these constraints (11, 12), usually the alternating least squares (ALS) algorithm Michailidis & de Leeuw (1998) is used. The basic idea of ALS is to iteratively optimize with respect to X or to $[Y_1, \ldots, Y_M]$ with the other held fixed. Assuming $X^{(0)}$ is provided arbitrarily at iteration $t = 0$, each iteration of ALS can be summarized as:

1. $\forall j \in [1, J]$,update Y_j:

$$Y_j^{(t)} = (G_j^\top G_j)^{-1} G_j^\top X^{(t)} \tag{13}$$

2. update X:

$$X^{(t+1)} = J^{-1} \sum_{j=1}^{J} G_j Y_j^{(t)} \tag{14}$$

3. normalize X:

$$X^{(t+1)} = \textit{Gram-Schmidt}(X^{(t+1)}) \tag{15}$$

It can be seen (13) that the category quantification of Y_j is computed as the centroid of the object scores that belong to it. Step 2 (14) updates object scores X by taking the average of the quantifications of the categories it belongs to. In step 3 (15) a *Gram-Schmidt* procedure is used to find the normalized and orthogonal basis of updated object scores from the previous step. In this way, the object scores will be located close to the category quantifications they fall in, and category quantifications will be close to the object scores belonging in them.

3 Linking CCA and HA

Based on the previous section, we can see that CCA and HA are used in two different data types: the former operates on two sets of variables while the latter is used on multivariate categorical variables. We reveal that they are closely related when CCA is generalized to multiple sets of variables. Suppose we want to find 2 sets of p linear projections $\mathbf{W}_1 = [\mathbf{w}_1^1, \ldots, \mathbf{w}_1^p]$, $\mathbf{W}_2 = [\mathbf{w}_2^1, \ldots, \mathbf{w}_2^p]$ for regular CCA. We can rewrite (2) as

$$\begin{aligned} &\{\mathbf{W}_1^*, \mathbf{W}_2^*\} = \underset{\mathbf{W}_1, \mathbf{W}_2}{\operatorname{argmin}} ||G_1\mathbf{W}_1 - G_2\mathbf{W}_2||_F^2 \\ &s.t. \quad \forall i \in \{1,2\}, \forall k, h \in [1,p], k \neq t \\ &\quad \mathbf{W}_i^\top C_{ii} \mathbf{W}_i = I_p, \quad \mathbf{w}_1^{k\top} C_{12} \mathbf{w}_2^h = 0 \end{aligned} \tag{16}$$

When we have $J > 2$ sets of variables, (16) will be:

$$\begin{aligned} &\{\mathbf{W}_1^*, \ldots, \mathbf{W}_J^*\} = \underset{\mathbf{W}_1, \ldots, \mathbf{W}_J}{\operatorname{argmin}} \sum_{i=1,j=1}^{J} ||G_i\mathbf{W}_i - G_j\mathbf{W}_j||_F^2 \\ &s.t. \quad \forall i, j \in [1,J], \forall k, h \in [1,p], k \neq h \\ &\mathbf{W}_i^\top C_{ii} \mathbf{W}_i = I_p, \quad \mathbf{w}_i^{k\top} C_{ij} \mathbf{w}_j^h = 0 \end{aligned} \tag{17}$$

Lemma 3.1 *The objective function in (17) is equivalent to*

$$\min_{X, \mathbf{W}_1, \ldots, \mathbf{W}_J} \frac{1}{J} \sum_{j=1}^{J} ||X - G_j\mathbf{W}_j||_F^2 \tag{18}$$

Proof For simplicity, we only consider one data instance $\mathcal{D} = [g_1^\top, \ldots, g_J^\top]$. $\forall i, j \in [1,J], i \neq j$, we denote $\mathbf{W}_i^\top g_i$ and $\mathbf{W}_j^\top g_j$ as v_i and v_j respectively, $v_i, v_j \in \mathbb{R}^p$. Then the objective function in (17) is

$$\begin{aligned} & \sum_{i=1,j=1}^{J} ||v_i - v_j||^2 \\ = & \sum_{i=j,j=1}^{J} \sum_{k=1}^{p} \left(v_{ik}^2 + v_{jk}^2 - 2v_{ik}v_{jk}\right) \\ = & \sum_{k=1}^{p} \left(\sum_{i=1,j=1}^{J} v_{ik}^2 + \sum_{i=1,j=1}^{J} v_{jk}^2 - \sum_{i=1,j=1}^{J} 2v_{ik}v_{jk} \right) \\ = & \sum_{k=1}^{p} \left(J\sum_{i=1}^{J} v_{ik}^2 + J\sum_{j=1}^{J} v_{jk}^2 - 2\sum_{i=1}^{J} v_{ik} \sum_{j=1}^{J} v_{jk} \right). \end{aligned} \tag{19}$$

In addition, by denoting $\mathcal{M}_1^k = \frac{1}{J}\sum_{j=1}^{J} v_{jk}$, $\mathcal{M}_2^k = \frac{1}{J}\sum_{j=1}^{J} v_{jk}^2$, (19) is equal to

$$J^2 \sum_{k=1}^{p} \left(\mathcal{M}_2^k - (\mathcal{M}_1^k)^2\right). \tag{20}$$

Since $\left(\mathcal{M}_2^k - (\mathcal{M}_1^k)^2\right)$ is the variance of the kth component in $\{v_i\}_{i=1}^{J}$, this is further equal to

$$J^2 \sum_{k=1}^{p}\sum_{j=1}^{J}(v_{ik} - \mathcal{M}_1^k)^2 = J^2 \sum_{j=1}^{J} ||v_i - \mathbf{M}_1||^2 \tag{21}$$

where $\mathbf{M}_1 = [\mathcal{M}_1^1, \ldots, \mathcal{M}_1^p]^\top$. (21) can be phrased as a rescaled optimization problem

$$\min_X \frac{1}{J}\sum_{j=1}^{J} ||v_i - X||^2 \tag{22}$$

with optimal solution $X = \mathbf{M}_1 = \frac{1}{J}\sum_{j=1}^{J} v_j$. When M data instances are considered, it is straightforward to extend (22) to

$$\min_X \frac{1}{J}\sum_{j=1}^{J} ||G_i\mathbf{W}_i - X||_F^2 \tag{23}$$

which completes the proof of the lemma. ∎

Comparing (10) and (23), we can see that multiple-set CCA has the same objective function as HA (by replacing Y_i with $\mathbf{W}_i$), yet with different constraints; see (11), (12) and (17). In the following, we will show some connections between constraints in multiple-set CCA Ω_{mCCA} and constraints in HA Ω_{HA}.

First, since in $\Omega_{mCCA}, \forall j \in [1, J], \mathbf{1}_{M\times 1}^\top G_j\mathbf{W}_j = 0, \mathbf{1}_{M\times 1}^\top X = \frac{1}{J}\sum_{j=1}^{J}\mathbf{1}_{M\times 1}^\top G_j\mathbf{W}_j = 0$, which coincides with the first constraint in Ω_{HA} (11). Secondly, in $\Omega_{mCCA}, \forall i, j \in [1, J], \forall k, h \in [1, p], k \neq h, \mathbf{W}_i^\top C_{ii}\mathbf{W}_i = I_p, \quad \mathbf{w}_i^{k\top} C_{ij}\mathbf{w}_j^h = 0$. Therefore, $X^\top X = \frac{1}{J^2}\left(\sum_{j=1}^{J}\mathbf{W}_j^\top C_{jj}\mathbf{W}_j + 2\sum_{i\neq j}\mathbf{W}_i^\top C_{ij}\mathbf{W}_j\right)$. We can see that when the correlation of projected data in every pair (i, j) are ideally maximized to 1, $X^\top X = I_p$, which is a rescaled version of the second constraint in Ω_{HA} (11). However, satisfying Ω_{HA} cannot ensure satisfaction of any constraint in Ω_{mCCA}. Therefore, roughly speaking, we can consider Ω_{mCCA} as a sufficient but not necessary condition for Ω_{HA}, or in other words, Ω_{HA} is a relaxed version of Ω_{mCCA}.

4 Kernel Generalized Homogeneity Analysis

Based on the analysis above, we can generalize HA as a relaxed variant of multiple-set CCA by replacing binary indicator matrices of J types of features. One strength we gain by using HA is that normalization constraints on J individual projections are eliminated. Therefore, by using ALS for training, multiple pair-wise eigenvalue computations can be avoided. In a supervised-learning context, we can assume that the Jth set of variables are outputs (denoted by $T = [t^{(1)}, t^{(2)}, \ldots, t^{(M)}]^\top \in \mathbb{R}^{M\times d_J}$) and the remaining $J-1$ sets of variables represent $J-1$ input features from heterogeneous

information sources. Then (10) is rewritten as

$$\begin{aligned} & f(X, \mathbf{W}, \ldots, \mathbf{W}_{J-1}, \mathbf{P}) \\ = \quad & \frac{1}{J}\left(\sum_{j=1}^{J-1} \underbrace{||X - G_j\mathbf{W}_j||_F^2}_{\rho_j} + \underbrace{||X - T\mathbf{P}||_F^2}_{\pi}\right) \end{aligned} \tag{24}$$

where $\mathbf{P}$ is the projection associated with outputs T[1]. Interestingly, ρ_j and π in (24) are identical to the predictability and recoverability of X respectively, which are two concepts recently introduced in FaIE Lin et al. (2014). More concretely, predictability is measured by how much input features are correlated with lower-dimensional representations of multi-label outputs, while recoverability refers to how successfully the compact representations can be decoded back to binary vectors. It is worth noting that only one ρ_j was used in FaIE. Following the philosophy of Lin et al. (2014), we also introduce a trade-off parameter λ to balance $\sum_{j=1}^{J} \rho_j$ and π. After rescaling we can further rewrite (24) as:

$$f = \lambda \sum_{j=1}^{J-1} ||X - G_j\mathbf{W}_j||_F^2 + ||X - T\mathbf{P}||_F^2 \tag{25}$$

Similarly to kernel CCA and kernel FaIE, we can add a kernel function (8), for each feature, on a pair of data points, $\mathcal{K}_j(G_j^{(m)}, G_j^{(n)}), j \in [1, J], m, n \in [1, M]$. We refer to this novel learning method as kernel generalized HA (KGHA). Since updates of Y_j in (13) solve a multivariate linear regression (MLR), by replacing it with a dual form of kernel multivariate ridge regression (KMRR), we can develop a dual learning algorithm for KGHA by changing the first two steps in ALS to

1. $\forall j \in [1, J]$, update the dual matrix $\boldsymbol{\alpha}_j \in \mathbb{R}^{M \times p}$:

$$\boldsymbol{\alpha}_j^{(t)} = (K_j + c_j I_M)^{-1} X \tag{26}$$

2. update X:

$$X^{(t+1)} = \frac{1}{\lambda(J-1)+1}\left(\sum_{j=1}^{J-1} \lambda K_j \boldsymbol{\alpha}_j + K_J \boldsymbol{\alpha}_J\right) \tag{27}$$

where c_j is a ridge parameter for each feature . K_j denotes the kernel matrix of the data within the kth feature or the Gram matrix if no kernel function is applied.

5 Multi-Label Learning with KGHA

We now investigate the application of KGHA on multi-label learning, in which KGHA works as a learning framework with low-rank output kernel learning and subspace co-regularized multi-view learning. For the kernel on the jth feature ($j \in [1, J-1]$), the original data are mapped to a RKHS $\phi_j(G_j^{(m)}) \in \mathcal{H}_j, m \in [1, M]$. We define a linear kernel on multi-label outputs as did Hariharan et al. (2010) and Dinuzzo et al. (2011):

$$\mathcal{K}_T(t^{(m)}, t^{(n)}) = \left\langle \phi_T(t^{(m)}), \phi_T(t^{(n)}) \right\rangle = \left\langle \mathbf{Q}^\top t^{(m)}, \mathbf{Q}^\top t^{(n)} \right\rangle \tag{28}$$

[1]From now on, we refer to the same thing by using G_J or T, and similarly, $\mathbf{P}$ and $\mathbf{W}_J$ are equivalent.

where $t^{(m)}, t^{(n)} \in \mathbb{B}^{d_J} = \mathcal{T}$, $\mathbf{Q} \in \mathbb{R}^{d_J \times d_J}$ captures the pairwise dependencies between elements in $t^{(m)}$. Using a pairwise formulation as in (17), the objective function of KGHA is

$$\begin{aligned} &\sum_{j=1}^{J-1} \underbrace{||\phi_j(G_j)\mathbf{W}_j - T\mathbf{Q}\mathbf{P})||_F^2}_{\mathcal{A}_j} \\ &+ \lambda \sum_{i,j=1:i\neq j}^{J-1} \underbrace{||\phi_i(G_i)\mathbf{W}_i - \phi_j(G_j)\mathbf{W}_j||_F^2}_{\mathcal{B}_{ij}} \end{aligned} \tag{29}$$

where $\phi_j(G_j) = [\phi_j(G_j)^{(1)}, \ldots, \phi_j(G_j)^{(M)}]^\top$, $\sum_{j=1}^{J-1} \mathcal{A}_j$ corresponds to low-rank output kernel learning with $J-1$ features, while $\sum_{i,j=1:i\neq j}^{J-1} \mathcal{B}_{ij}$ corresponds to co-regularization for multi-view learning.

5.1 Low-Rank Output Kernel Learning

Let $\tilde{\mathbf{Q}} = \mathbf{Q}\mathbf{P}$ be a low-rank feature map $\tilde{\phi}_T$ for $\mathcal{T}$. Then, $\mathcal{K}_T(t^{(m)}, t^{(n)}) = t^{(m)\top}\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^\top t^{(n)} = t^{(m)\top}\mathbf{L}t^{(n)}$. In each $\mathcal{A}_j$, with a feature map defined on both input and output, a function to be learned is defined as $f_j(G_j^{(m)}, t^{(m)}) = \mathbf{W}_j^\top \left(\phi_j(G_j^{(m)}) \otimes \tilde{\phi}_T(t^{(m)})\right)$, where $\otimes$ denotes tensor product. Therefore, within the framework of regularization in reproducing kernel Hilbert spaces (RKHS) of vector-valued functions Micchelli & Pontil (2005), the unique kernel $\mathbf{H}_j$ associated with the RHKS of $\tilde{\phi}_T(\mathcal{T})$-valued function is

$$\begin{aligned} \mathbf{H}_j &= \left\langle \phi(G_j^{(m)}) \otimes \tilde{\phi}_T(t^{(m)}), \phi(G_j^{(n)}) \otimes \tilde{\phi}_T(t^{(n)}) \right\rangle \\ &= \left\langle \phi(G_j^{(m)}), \phi(G_j^{(n)}) \right\rangle \left\langle \tilde{\phi}_T(t^{(m)}), \tilde{\phi}_T(t^{(n)} \right\rangle \\ &= \left\langle t^{(m)}, \mathcal{K}_j^{m,n}\mathbf{L}t^{(n)} \right\rangle, \end{aligned}$$

where $\mathcal{K}_j^{m,n}$ is the kernel value $\mathcal{K}_j(G_j^{(m)}, \phi(G_j^{(n)})$. $\mathcal{K}_j^{m,n}\mathbf{L}$ defines an operator-valued, positive semidefinite $\mathcal{T}$-kernel: $\mathbb{R}^{d_j} \times \mathbb{R}^{d_j} \to \mathbb{R}^{d_J \times d_J}$. Because of the decomposability $\mathbf{H}_j = \mathcal{K}_j \cdot \mathbf{L}$ Dinuzzo et al. (2011), $\mathbf{L}$ corresponds to a low-rank output kernel Dinuzzo & Fukumizu (2011). Since $\tilde{\mathbf{Q}}$ itself specifies a linear dimensionality reduction, a plain Gram matrix is used for T in (26) to learn $\tilde{\mathbf{Q}}$. Low-rank output kernel learning, to some extent, is equivalent to multi-label dimensionality reduction (see a review by Ye et al.2011), whose target is to find a lower-dimensional latent space for multi-label space so as to capture inter-label dependencies as well as to remove nuisance noise.

5.2 Co-regularized Multi-view Learning

Co-regularization has been popularly employed in multi-view learning Farquhar et al. (2005); Brefeld et al. (2006); Rosenberg & Bartlett (2007). Essentially, co-regularization works as an extra model-complexity controller by penalizing functions which tend to generate big disagreements among multiple views (see pairwise $\mathcal{B}_{ij}$ in (29)). In particular, an improved generalization bound of using co-regularization was presented by Rosenberg & Bartlett (2007) in terms of Rademacher complexities. While most co-regularization is for semi-supervised learning, quite similar to our work, a subspace co-regularised multi-view learning paradigm was proposed by Guo & Xiao (2012) for supervised learning. A similar regularization is also used in MultiK-MHKS Wang et al. (2008) for multiple kernel learning, which strategically integrates heterogeneous information with an ensemble of kernels. Since MultiK-MHKS works on the original binary

output space, the squared misclassification loss is used. However, a least-squares loss is used in KGHA as a regression on lower-dimensional representations of multi-label outputs.

5.3 Prediction

With training data $\mathcal{D} = [G_1, G_2, \ldots, G_{J-1}; T]$, we can obtain $J-1$ dual matrices $\{\boldsymbol{\alpha}\}_{j=1}^{J-1}$ and one linear output decoding matrix $\tilde{\mathbf{Q}} = T^\top \boldsymbol{\alpha}_J$. Then, given a test inputs $\mathcal{D}^{test} = [\dot{G}_1^\top, \dot{G}_2^\top, \ldots, \dot{G}_{J-1}^\top]$, the predicted lower-dimensional representation is

$$\dot{X} = \frac{1}{J-1} \sum_{j=1}^{J-1} \mathcal{K}_j(\dot{G}_j, G_j)\boldsymbol{\alpha}_j \tag{30}$$

where $\mathcal{K}_j(G_j, \dot{G}_j)$ is a cross kernel matrix. Then the score values of labels are computed as

$$\dot{T} = \dot{X}\tilde{\mathbf{Q}}^\top(\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^\top)^\dagger \tag{31}$$

where † denotes the Moore-Penrose pseudoinverse. Finally, labels can be predicted by retrieving top$-l$ (l is the desired number of labels) ranked score values.

6 Large-Scale KGHA Learning

The computation in each ALS iteration is dominated by the matrix inversion in (26); thus the time complexity of training KGHA is $O(JM^3)$. On the other hand, the space complexity is $O(JM^2)$. In modern machine learning tasks, it is not uncommon to come across databases with large numbers (*e.g.* millions) of training instances. Like other kernel-based methods, learning on such large-scale databases is challenging since the storage and computation of large kernel matrices will go beyond the memory of normal PCs. To enable KGHA for large-scale learning, in our experiments we used low-rank approximations of kernel matrices with Memory Efficient Kernel Approximation (MEKA, Si et al.2014). MEKA is essentially a block-wise Nyström algorithm by first clustering instances to obtain dense diagonal kernel blocks, and then obtaining rank-k (k is small) approximations of all diagonal blocks with Nyström algorithm and also off-diagonal blocks with regression. For the kernel matrix of the j-th feature,

$$\tilde{K}_j \approx W_j L_j W_j^\top \tag{32}$$

where $W_j = \oplus_{s=1}^{S} W_j^{(s)}$ (*i.e.* the direct sum of $W_j^{(s)}$ in S blocks, $W_j \in \mathbb{R}^{Sk \times Sk}$) and $L_j \in \mathbb{R}^{Sk \times Sk}$ consists of S^2 block-linking matrices. MEKA was reported to outperform other low-rank approximations by exploiting the block structure in kernel matrices Si et al. (2014). By using MEKA, the space complexity of training KGHA decreases to $O(J(Mk + (ck)^2))$. In addition, to avoid the matrix inverse in (26), we employed *conjugate gradient* (CG) to solve the linear equation for each feature:

$$(K_j + c_j I_M)\boldsymbol{\alpha}_j = X \tag{33}$$

and consequently, time complexity of solving ALS is reduced to $O(JMk)$.

Dataset	#labels	#training instances	#test instances	#average labels
Corel5k	260	4500	500	3.3965
Espgame	268	18689	2081	4.6859
Iaprtc12	291	17665	1962	5.7187

Table 1: Statistics of three image-annotation benchmark datasets.

Feature	Dim	Source	Descriptor	Location
DenseHueV3H1	300	texture	Hue	dense
DenseSiftV3H1	3000	texture	Sift	dense
Gist	512	-	Holistic	-
HarrisHueV3H1	300	texture	Hue	Harris
HarrisSiftV3H1	3000	texture	Sift	Harris
HsvV3H1	5184	color	HSV	-
LabV3H1	5184	color	LAB	-
RgbV3H1	5184	color	RGB	-

Table 2: A summary of 8 heterogeneous visual features.

	Corel5K			Espgame			Iaprtc12		
p/d_J	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
0.2	26.1	30.7	28.2	30.1	18.8	23.1	35.9	24.7	29.3
0.4	30.8	35.1	32.8	33.7	24.4	28.3	38.2	25.2	30.4
0.5	**33.7**	**42.5**	**37.6**	**37.8**	**27.3**	**31.7**	40.1	**29.7**	**34.1**
0.6	29.1	38.6	33.2	33.1	26.8	29.6	40.7	26.4	32.0
0.8	28.5	38.1	32.6	31.9	25.5	28.3	**41.3**	26.5	32.2

Table 3: Performance of KGHA with different p values.

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp	**33.0**	42.0	**37.0**	39.0	**27.0**	**32.0**	45.0	**34.0**	**39.0**
FastTag	32.0	**43.0**	**37.0**	**46.0**	22.0	30.0	**47.0**	26.0	34.0
r-MLR	27.7	29.3	28.5	24.3	19.3	21.5	34.8	19.5	25.0
r-KMLR	31.7	35.1	33.3	24.8	26.6	25.7	36.6	20.1	26.0
L-HA	30.0	27.5	28.7	25.1	22.4	23.7	38.1	22.5	28.3
KGHA−d	33.1	38.1	37.5	32.7	27.2	29.7	**43.8**	21.3	28.7
KGHA−r	31.3	32.6	32.0	28.8	18.6	22.6	31.2	22.6	26.2
KGHA	**33.7**	**42.5**	**37.6**	**37.8**	**27.3**	**31.7**	40.1	**29.7**	**34.1**

Table 4: Comparison between KGHA and other related methods on three image-annotation benchmark databases. The results in the upper panel were reported by Chen et al. (2013).

7 Experiments

To evaluate the proposed KGHA for multi-label learning, we test it on image annotation and protein function prediction tasks.

7.1 Image Annotation

7.1.1 Data and Evaluation

In this experiment, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies Guillaumin et al. (2009); Makadia et al. (2010); Chen et al. (2013) with performance evaluations reported therein. Therefore, we can easily compare our method with others. Statistics of the three benchmark datasets are summarized in Table 1. Readers are referred to Makadia et al. (2010) for more details of the three datasets. We worked with 8 visual features extracted by Guillaumin et al. (2009). They include one Gist descriptor, three global color histograms and four histograms of local bag-of-words texture features[2]. The descriptions of 8 features are summarized in Table 2. Readers are referred to Guillaumin et al. (2009) for more details on extracting these features. Our large-scale learning scheme (section 6) is applied on Espgame and Iaprtc12 since these contain large numbers of instances.

Following Chen et al. (2013), 5 labels with top prediction score values were annotated to each image. We evaluated annotation performance using *precision* (P), *recall* (R), and the *F1* measure (F). For each tag, the precision is computed as the ratio of the number of images assigned the tag correctly over the total number of images predicted to have the tag, while the recall is the number of images assigned the tag correctly divided by the number of images that truly have the tag. Then precision and recall are averaged across all tags. Finally, the F1 measure is calculated as $F = 2\frac{P \times R}{P+R}$.

7.1.2 Results and Comparison

First, KGHA was implemented and tested on three databases. A Gaussian kernel $\mathcal{K}_j^{\text{Gauss}} = \exp(-||G_j^{(m)} - G_j^{(n)}||_2^2/2\sigma_j^2)$ was used on all 8 visual features, with σ_j set to the average value of $||G_j^{(m)} - G_j^{(n)}||_2, m, n \in [1, M]$. The reduced dimension p is set to different 5 values ($p = \{0.2, 0.4, 0.5, 0.6, 0.8\} \times d_J$). Hyperparameters $(\lambda, \{c_j\}_{j=1}^{J-1})$ were selected by grid search with 4-fold cross validation from $\{10^{-5}, \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Here for simplicity we use a common ridge parameter c for all J features, and we found it almost does not affect performance. Experimental results are presented in Table 3. It can be seen that the best performance was achieved with $p/d_J = 0.5$. on all three datasets. To verify the significance of low-rank output kernel learning and co-regularization, we also implemented another five simplified methods for comparison: (1) multivariate linear ridge regression (r-MLR); (2) multivariate kernel ridge regression (r-KMLR); (3) linear HA (L-HA); (4) KGHA with $p = d_J$ (KGHA$-$d, no dimensionality reduction); (5) KGHA with extremely small λ (KGHA$-$r, no manifold regularization). To ensure fairness, the same Gaussian kernel construction and appropriate hyperparameter searching are used in all methods. The results of all six methods are presented in the lower panel of Table 4. We see that KGHA generally outperforms other methods, which empirically proves the importance of low-rank output kernel learning and co-regularization. In addition, the upper panel of Table 4 lists the results of some notable methods that were recently developed or surveyed Guillaumin et al. (2009); Makadia et al. (2010); Chen et al. (2013). KGHA demonstrates promising capabilities by comparing favorably to the state-of-the-art.

[2]In the original dataset, two versions of color features and texture features are available, with and without spatial layout; here we use only those with layout.

Dataset	information sources for kernels	#labels	#instances	#ave. labels
Human	domain composition complexes protein-DNA/RNA-interaction p-transcriptional modification tissue expression protein interaction OMIM disease Su-tissue expression	254	3704	3.7268
Yeast	Pfam domain structure co-participation PPI network gene interaction network cell cycle&gene expression	13	3588	1.5279

Table 5: The statistics and heterogeneous features of two datasets.

7.2 Protein Function Prediction

7.2.1 Data and Evaluation

Two datasets were used in the second experiment. The first one is Human from Mostafavi & Morris (2010) and the second one is Yeast from Tsuda et al. (2005). Following the same preprocessing on Human dataset by Yu et al. (2013), we filtered out some protein instances with too general or too narrow functions. For both datasets, multiple kernels (similarity networks) were already pre-computed. The statistics and heterogeneous features of two datasets are listed in Table 5.

For better comparison with the state-of-the-art results in bioinformatics community, we evaluated performances with $1-$RankingLoss Yu et al. (2012, 2013). *RankingLoss* is computed as the average fraction of label pairs which are incorrectly ordered. To align with Macro F1, $1-$RankingLoss was used instead. The score values of all labels in (31) were used for ordering. In the Human experiment, 60% of the instances were used for training and 40% for testing. On the other hand, 80% of instances were used for training and 20% for testing in Yeast experiment. In both cases, 5 independent trials were run with different random subsets, based on which the average and standard deviation of $1-$RankingLoss were computed and reported.

7.2.2 Results and Comparison

Similarly to our image-annotation experiment, KGHA with five different p values were tried on Human and Yeast databases. Also, the same hyper-parameter tuning was conducted. Results are presented in Table 6, from where we can see that optimal p is $0.5 \times d_J$ in the Human experiment and $0.6 \times d_J$ in Yeast. Three simplified methods (r-MLR and L-HA were not used since only kernel values are provided in original datasets) were also tried on two datasets with results listed in the lower panel of Table 7. KGAH is again superior to all others, which suggests its robustness and stability. In addition, four state-of-the-art results on these two datasets (see the upper panel of Table 7) were reported in Yu et al. Yu et al. (2012, 2013). It can be seen that KGHA yielded rather comparable performance.

p/d_J	Human	Yeast
0.2	70.03 ($\pm$0.69)	71.58 ($\pm$0.63)
0.4	78.63 ($\pm$0.53)	76.00 ($\pm$0.89)
0.5	**82.74** ($\pm$0.56)	81.13 ($\pm$0.75)
0.6	77.14 ($\pm$0.42)	80.06 ($\pm$0.83)
0.8	76.31 ($\pm$0.61)	**83.97** ($\pm$0.80)

Table 6: Evaluation of KGHA using $1-$RankingLoss (%) with different p values.

Method	Human	Yeast
SW	78.49 ($\pm$0.58)	**84.35** ($\pm$0.73)
GRF-MK	81.11 ($\pm$0.51)	80.93 ($\pm$0.99)
PfunBG-MK	81.60 ($\pm$0.50)	80.50 ($\pm$0.87)
TMEC	**83.40** ($\pm$0.46)	80.79 ($\pm$1.01)
r-KMLR	72.10 ($\pm$0.65)	73.11 ($\pm$0.94)
KGHA$-$d	78.69 ($\pm$0.57)	75.25 ($\pm$0.92)
KGHA$-$r	77.15 ($\pm$0.78)	72.69 ($\pm$1.24)
KGHA	**81.74** ($\pm$0.56)	**83.97** ($\pm$0.80)

Table 7: Comparison between KGHA and other related methods using $1-$RankingLoss (%) on three image-annotation benchmark databases. The results in the upper panel were reported by Yu et al. (2013) and Yu et al. (2012).

8 Conclusion

A novel multi-label learning framework, kernel generalized homogeneity analysis (KGHA), was proposed. Starting from the connections between regular HA and multiple-set CCA, we revealed that HA can be generalized as a relaxed variant of multiple-set CCA to handle multiple heterogeneous features. By using kernel functions, we showed that KGHA, in multi-label learning, works as a method consisting of low-rank output kernel learning and co-regularized multi-view learning. We also presented some interesting links between low-rank output kernel learning and multi-label dimensionality reduction, co-regularization and multiple kernel learning, respectively. Promising results are achieved by using KGHA in our experiments on image annotation and protein function prediction.

References

Bie, Tijl De, Cristianini, Nello, and Rosipal, Roman. Eigenproblems in Pattern Recognition. In *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*, pp. 129–170. Springer, 2005.

Brefeld, Ulf, Gärtner, Thomas, Scheffer, Tobias, and Wrobel, Stefan. Efficient co–regularised least squares regression. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Bucak, Serhat Selcuk, Jin, Rong, and Jain, Anil K. Multiple kernel learning for visual object recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1354–1369, 2014.

Chen, Minmin, Zheng, Alice, and Weinberger, Kilian Q. Fast image tagging. In *ICML*, 2013.

Dinuzzo, Francesco and Fukumizu, Kenji. Learning low-rank output kernels. In *ACML*, 2011.

Dinuzzo, Francesco, Ong, Cheng S., Gehler, Peter V., and Pillonetto, Gianluigi. Learning Output Kernels with Block Coordinate Descent. In *ICML*, 2011.

Elisseeff, André and Weston, Jason. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

Farquhar, Jason D. R., Hardoon, David R., Meng, Hongying, Taylor, John S., and Szedmák, Sándor. Two view learning: SVM-2K, theory and practice. In *NIPS*, 2005.

Guillaumin, Matthieu, Mensink, Thomas, Verbeek, Jakob, and Schmid, Cordelia. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

Guo, Yuhong and Xiao, Min. Cross language text classification via subspace co-regularized multi-view learning . In *ICML*, 2012.

Hardoon, David R., Szedmak, Sandor, and Shawe-Taylor, John. Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation*, 16(12):2639–2664, 2004.

Hariharan, Bharath, Zelnik-Manor, Lihi, Vishwanathan, S. V. N., and Varma, Manik. Large scale max-margin multi-label classification with priors. In *ICML*, 2010.

Hsu, Daniel, Kakade, Sham, Langford, John, and Zhang, Tong. Multi-label prediction via compressed sensing. In *NIPS*, 2009.

Izenman, Alan J. *Modern Multivariate Statistical Techniques : Regression, Classification, and Manifold Learning*. Springer New York, 2008.

Ji, Shuiwang and Ye, Jieping. Linear dimensionality reduction for multi-label classification. In *International Jont Conference on Artifical Intelligence*, IJCAI'09, 2009.

Lin, Zijia, Ding, Guiguang, Hu, Mingqing, and Wang, Jianmin. Multi-label classification via feature-aware implicit label space encoding. In *ICML-14*, 2014.

Ma, Yunqian and Fu, Yun. *Manifold Learning Theory and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011. ISBN 1439871094, 9781439871096.

Makadia, Ameesh, Pavlovic, Vladimir, and Kumar, Sanjiv. Baselines for image annotation. *International Journal of Computer Vision*, 90:88–105, 2010.

Micchelli, Charles A. and Pontil, Massimiliano. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

Michailidis, George and de Leeuw, Jan. The Gifi System Of Descriptive Multivariate Analysis. *STATISTICAL SCIENCE*, 13:307–336, 1998.

Mostafavi, Sara and Morris, Quaid. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics*, 26(14):1759–1765, 2010.

Rosenberg, David S. and Bartlett, Peter L. The rademacher complexity of co-regularized kernel classes. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*. Journal of Machine Learning Research - Proceedings Track, 2007.

Si, Si, jui Hsieh, Cho, and Dhillon, Inderjit. Memory efficient kernel approximation. In Jebara, Tony and Xing, Eric P. (eds.), *ICML-14*, 2014.

Sridharan, Karthik and Kakade, Sham M. An information theoretic framework for multi-view learning. In *21st Annual Conference on Learning Theory - COLT*, 2008.

Sun, Liang, Ji, Shuiwang, Yu, Shipeng, and Ye, Jieping. On the equivalence between canonical correlation analysis and orthonormalized partial least squares. In *IJCAI*, 2009.

Sun, Liang, Ji, Shuiwang, and Ye, Jieping. Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1), January 2011.

Tsuda, Koji, Shin, Hyunjung, and Schölkopf, Bernhard. Fast protein classification with multiple networks. *Bioinformatics*, 21(2):59–65, January 2005.

Wang, Zhe, Chen, Songcan, and Sun, Tingkai. Multik-mhks: A novel multiple kernel learning algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2), February 2008.

Xiong, Hanchen, Szedmák, Sándor, and Piater, Justus H. Joint SVM for accurate and fast image tagging. In *European Symposium on Artificial Neural Networks, ESANN*, 2014.

Ye, Jieping, Ji, Shuiwang, and Sun, Liang. *Multi-Label Dimensionality Reduction*. Chapman & Hall/CRC, 2011.

Yu, Guoxian, Domeniconi, Carlotta, Rangwala, Huzefa, Zhang, Guoji, and Yu, Zhiwen. Transductive multi-label ensemble classification for protein function prediction. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

Yu, Guoxian, Rangwala, Huzefa, Domeniconi, Carlotta, Zhang, Guoji, and Yu, Zhiwen. Protein function prediction using multi-label ensemble classification. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 10(4), July 2013.

Chapter 5

Optimization on Structures

"For the things we have to learn before we can do them, we learn by doing them"

Aristotle

5.1 Optimzation on Graphs

5.2 Optimization on Matrix Manifolds

5.2.1 3D Point Cloud Registration with Convex Optimization on SO(3) Manifold

IX. **Hanchen Xiong**, Sandor Szedmak, Justus Piater *Efficient, General Point Cloud Registration With Kernel Feature Maps*, In Proceedings of 10th International Conference on Computer and Robot Vision (CRV13), pp 83-90, 2013, IEEE.

Efficient, General Point Cloud Registration With Kernel Feature Maps

Hanchen Xiong, Sandor Szedmark, Justus Piater
Institute of Computer Science, University of Innsbruck
Technikerstr.21a A-6020, Innsbruck, Austria
Email: {hanchen.xiong, sandor.szedmark, justus.piater}@uibk.ac.at

***Abstract*—This paper proposes a novel and efficient point cloud registration algorithm based on the kernel-induced feature map. Point clouds are mapped to a high-dimensional (Hilbert) feature space, where they are modeled with Gaussian distributions. A rigid transformation is first computed in feature space by elegantly computing and aligning a small number of eigenvectors with kernel PCA (KPCA) and is then projected back to 3D space by minimizing a consistency error. $SE(3)$ on-manifold optimization is employed to search for the optimal rotation and translation. This is very efficient; once the object-specific eigenvectors have been computed, registration is performed in linear time. Because of the generality of KPCA and $SE(N)$ on-manifold method, the proposed algorithm can be easily extended to registration in any number of dimensions (although we only focus on 3D case). The experimental results show that the proposed algorithm is comparably accurate but much faster than state-of-the-art methods in various challenging registration tasks.**

***Keywords*-kernel method; point cloud registration; SE(3) on-manifold optimization**

I. INTRODUCTION

3D free-form shape registration is an important problem in many fields and has sparked a large volume of related research literature. During the past two decades, 3D point clouds have become an increasingly more important and popular data structure to represent 3D shapes. Especially in contemporary robotics, 3D point cloud registration is an essential component of autonomous systems to assist in the perception of 3D objects and environments.

Until now, most existing 3D point cloud registration algorithms decompose the registration problem into two parts, *correspondence assignment* and *alignment*, because they argue that computing either of two steps will facilitate the other. A popular method is Iterative Closest Point (ICP) [1], which undoubtedly has been most widely used due to its simplicity in implementation. First, a pseudo correspondence is established by finding the nearest neighbor of each point. Then, the optimal rotation and translation are computed so as to minimize the average of Euclidean distances between all pairs of corresponding points. These two steps are iterated until converge. Obviously, the closest-distance criterion for correspondence is too weak, and therefore ICP can easily fail in practice when the displacement between two point clouds or outlier rate is relatively large. To enhance the accuracy of the correspondence, many improved versions of ICP were proposed by incorporating color, normal vectors, curvature, or strategically ignoring some unlikely correspondences [2]. However, despite various improvements, the *hard* assignment strategy employed by ICPs causes problems that require manual assistance in practical applications. To overcome this limitation, SoftAssign [3] and EM-ICP [4] were proposed by establishing one-to-many *soft* correspondences. Both methods assume that one point may correspond to all points in the other cloud with different likelihoods by constructing a correspondence matrix. To iteratively update this matrix, deterministic annealing is used in SoftAssign while an Expectation-Maximization (EM)-style method is employed in EM-ICP. Meanwhile, recently a Gaussian-mixture (GM) method [5] was developed to avoid iteratively computing the correspondences and alignment. GM probabilistically and globally models 3D point clouds as Gaussian mixtures in $\mathbb{R}^3$, and the optimal alignment between point clouds is computed by minimizing the discrepancy (L2 distance [5]) between their corresponding distributions.

For the task of aligning 3D point clouds $\mathbf{M}_1 = \{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ with $\mathbf{M}_2 = \{\mathbf{x}_i^{(2)}\}_{i=1}^{l_2}$, all methods described above can be interpreted as optimizing a common objective function:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{i=1}^{l_1}\sum_{j=1}^{l_2} \left(\mathbf{R}\mathbf{x}_i^{(1)} + \mathbf{b} - \mathbf{x}_j^{(2)}\right)^2 w_{i,j} \quad (1)$$

where $w_{i,j}$ denotes the correspondence between every pair of $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$. In ICP $w_{i,j} \in \{0,1\}$, and (1) is solved by iteratively updating $w_{i,j}$ in a winner-take-all manner under a shortest-distance criterion and solving a least-squares problem with respect to $\mathbf{R}$ and $\mathbf{b}$. In EM-ICP, $w_{i,j}$ is interpreted as the probability of the correspondence, so a one-way constraint ($w_{i,j} \in [0,1]$, $\sum_j^{l_2} w_{i,j} = 1$) is implicitly imposed. In SoftAssign, a stricter two-way constraint ($w_{i,j} \in [0,1]$, $\sum_j^{l_2} w_{i,j} = 1$, $\sum_i^{l_1} w_{i,j} = 1$) is introduced to enforce globally consistent point correspondences. Although GM does not model the correspondences explicitly, they can likewise be understood as an instance of (1) with Euclidean distance replaced by Mahalanobis distance, and an uniform prior of $w_{i,j} = \frac{1}{l_1 l_2}$ for each pair of i, j. In conclusion, so far 3D point cloud registration can be achieved either by explicitly modeling correspondences and laboriously updating them (EM-ICP and SoftAssign), or by making some fragile correspondence assumptions to sim-

plify the optimization procedure (ICP and GM). In addition, all these methods share the same computational complexity of $O(n^2)$ [1], which will be a heavy computational burden if the number of points n is relatively large. Therefore, a registration solution that can both express realistic priors over point correspondence matches and can be computed in a simpler (possibly non-iterative) and cheaper (possibly non-quadratic time) way is highly desirable. The method proposed in this paper fulfills both demands. Instead of doing point-wise correspondence search and computing in 3D space, our method first maps all points to a higher-dimensional (reproducing kernel Hilbert) feature space using kernel methods. The optimal transformation in feature space is then found by aligning Gaussians that approximate the two mapped point clouds. To project back to the 3D space, an objective function is constructed based on the fact that the transformed mapped points should be consistent with mapped transformed points. Finally, an $SE(3)$ on-manifold optimization scheme is exploited to provide an elegant and efficient gradient-type algorithm for registration.

Compared to previous registration methods, the strength of our method can be summarized in four points: (1) Although our algorithm was not developed on the basis of point correspondences, the form of its objective function (section III-A) suggests that correspondences are implicitly derived and to large degree it is consistent with the correct matches; (2) The experimental results (section IV) show that our method can work accurately and robustly in various challenging cases (large motion, outlier points); (3) Our method is much more efficient than other state-of-the-art methods, actually the computation complexity is $O(n \log n)$; (4) By using Kernel PCA and $SE(3)$ on-manifold optimization, the algorithm can be used as general point cloud registration framework with high flexibility and extensibility to any dimension.

II. RIGID TRANSFORMATION IN HILBERT SPACE

Intuitively, a straightforward way to align point clouds without point-wise correspondences is to first probabilistically fit each point cloud to a single Gaussian distribution in $\mathbb{R}^3$ and then align their means (translation) and covariances (rotation). However, the modeling ability of one single Gaussian in 3D space is too limited to capture the 3D point distribution of real-world objects, i.e. the mean and covariance in $\mathbb{R}^3$ are very sensitive to outliers. Inspired by kernel methods developed for set-format data [6], a 3D point clouds can be implicitly mapped to a much higher-dimensional Hilbert feature space, where a single Gaussian can fit well (Fig. 1) and hence yields higher tolerance to 3D disturbance in the original point cloud (e.g. outliers or non-rigid transformation). In addition, by applying kernel PCA, the eigenvectors of covariances can be efficiently computed and aligned without explicit computation in feature space.

[1] the complexity of ICP is $O(n \log n)$ if K-d trees are used for searching for the nearest neighbour

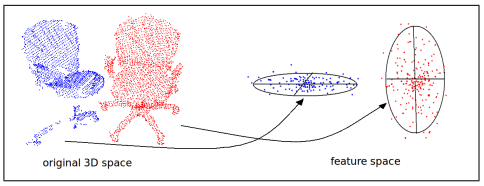


Figure 1. Mapping point clouds from 3D space to an infinite-dimensional Hilbert space, where a single Gaussian is sufficient to model distributions of complex shape.

A. Probabilistic modeling in Hilbert space

Inspired by kernel methods that have been widely used in machine learning, in order to map all points in a point cloud $\mathbf{M} = \{\mathbf{x}_i\}_{i=1}^{l}$ to a higher, possibly infinite-dimensional feature space, we can define a kernel function on 3D points $K(\mathbf{x}_i, \mathbf{x}_j)$. Then, a feature map can be implicitly induced by satisfying

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{2}$$

where ϕ is the corresponding feature map: $\mathbb{R}^3 \to \mathcal{H}$, and $\mathcal{H}$ is referred to as the reproducing Hilbert feature space. Since the structure of $\mathbf{M}$ can be far too complicated in $\mathbb{R}^3$, to ensure that one single Gaussian is capable of modeling the distribution of $\{\phi(\mathbf{x}_i)\}_{i=1}^{l}$ in $\mathcal{H}$, in this paper we select the kernel function as the radial basis function (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \tag{3}$$

because the induced feature map is a scaled Gaussian probability density function (PDF),

$$\phi(\mathbf{x}_i) = f(\xi|\mathbf{x}_i) = \exp \frac{-\|\xi - \mathbf{x}_i\|^2}{2\sigma^2}, \tag{4}$$

i.e., $\phi(\cdot)$ corresponds to an infinite-dimensional feature map.

With all points mapped into feature space, a Gaussian (mean and covariance) in $\mathcal{H}$ can be easily fitted by using maximum likelihood estimation (MLE):

$$\boldsymbol{\mu}_{\mathcal{H}} = \frac{1}{l} \sum_{i=1}^{l} \phi(\mathbf{x}_i) = \frac{1}{l} \phi(\mathbf{M})^{\top} \mathbf{1}_l \tag{5}$$

$$\boldsymbol{\Sigma}_{\mathcal{H}} = \frac{1}{l} \sum_{i=1}^{l} \left(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\mathcal{H}}\right) \left(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\mathcal{H}}\right)^{\top} \tag{6}$$

where $\phi(\mathbf{M})^{\top} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_l)]$ and $\mathbf{1}_l$ is an l-dimensional vector with all elements equal to 1.

B. Kernel PCA

To achieve the alignment between two covariances, their eigenvectors should be computed first. However, this computation is non-trivial in feature space. Kernel principal component analysis (KPCA) [7] is a technique developed

to compute eigenvectors in feature space without explicit computation in $\mathcal{H}$. Here we briefly review the procedure of KPCA with its application to 3D point clouds.

Assuming all points are already centered in feature space, the covariance $\mathbf{\Sigma}_{\mathcal{H}}$ of the Gaussian can be estimated as

$$\mathbf{\Sigma}_{\mathcal{H}} = \frac{1}{l}\sum_{i=1}^{l}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \tag{7}$$

which is a symmetric bilinear form on $\mathcal{H}$. Analogous to the symmetric covariance matrices in the finite-dimensional case, its nonzero eigenvalue λ_k and corresponding eigenvector $\mathbf{u}_k$ should satisfy

$$\lambda_k \mathbf{u}_k = \mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k. \tag{8}$$

By substituting (7) into (8), we have

$$\mathbf{u}_k = \frac{1}{\lambda_k}\mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k = \sum_{i=1}^{l}\alpha_i^k\phi(\mathbf{x}_i) \tag{9}$$

where $\alpha_i^k = \frac{\phi(\mathbf{x}_i)^\top\mathbf{u}_k}{\lambda_k l}$. Therefore, all eigenvectors $\mathbf{u}_k$ with $\lambda_k \neq 0$ must lie in the span of $\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_l)$, and (9) is referred to as the dual form of $\mathbf{u}_k$. By left-multiplying $\sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top$ on both sides of equation (8), we have

$$\begin{aligned} & \sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top\lambda_k\mathbf{u}_k &=& \sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top\mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k \\ \Leftrightarrow\quad & \lambda_k\sum_{i,j=1}^{l}\alpha_i^k K(\mathbf{x}_i,\mathbf{x}_j) &=& \frac{1}{l}\sum_{i,j=1}^{l}\alpha_i^k K(\mathbf{x}_i,\mathbf{x}_j)^2 \\ \Leftrightarrow\quad & l\lambda_k\boldsymbol{\alpha}^k &=& \mathbf{K}\boldsymbol{\alpha}^k \end{aligned} \tag{10}$$

where $\mathbf{K}$ is an $l \times l$ kernel matrix with $\mathbf{K}_{ij} = K(x_i, x_j)$, $\boldsymbol{\alpha}^k = (\alpha_1^k, \alpha_2^k, \ldots, \alpha_l^k)^\top$. It can be seen that $\{\boldsymbol{\alpha}^k, \eta^k = l\lambda_k\}$ is actually an eigenvalue-eigenvector pair of matrix $\mathbf{K}$. Therefore, by using dual forms of eigenvectors, the eigenvector decomposition of $\mathbf{\Sigma}_{\mathcal{H}}$ can be transformed to the decomposition of the finite matrix $\mathbf{K}$. In addition, because all $\mathbf{u}_k$ should be unit vectors:

$$1 = \mathbf{u}_k^\top\mathbf{u}_k = \langle\boldsymbol{\alpha}^k, \mathbf{K}\boldsymbol{\alpha}^k\rangle = \eta^k\boldsymbol{\alpha}^{k\top}\boldsymbol{\alpha}^k \tag{11}$$

the $\boldsymbol{\alpha}^k$ should be normalized as:

$$\boldsymbol{\alpha}^k \leftarrow \frac{\boldsymbol{\alpha}^k}{\sqrt{\eta^k}} \tag{12}$$

However, though the point cloud can be easily centered in 3D space, it does not mean it is also centered in feature space. By replacing $\phi(\mathbf{x}_i)$ with $\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \boldsymbol{\mu}$, the corresponding kernel matrix $\tilde{\mathbf{K}}$ is

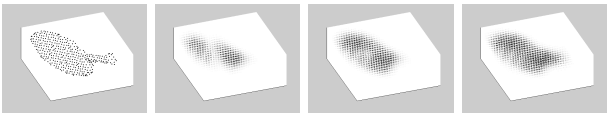
$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{l}\mathbf{E}\mathbf{K} - \frac{1}{l}\mathbf{K}\mathbf{E} + \frac{1}{l^2}\mathbf{E}\mathbf{K}\mathbf{E} \tag{13}$$

where $\mathbf{E}$ denotes an $l \times l$ matrix with all entries equal to 1. After similar eigenvector decomposition (10) and normalization (12) steps, we obtain eigenvectors

$$\tilde{\mathbf{u}}_k = \sum_{i=1}^{l}\tilde{\alpha}_i^k\left(\phi(\mathbf{x}_i) - \boldsymbol{\mu}\right) = \phi(\mathbf{M})^\top\underbrace{\left(\mathbf{I}_l - \frac{1}{l}\mathbf{E}\right)}_{\mathbf{I}^{\mathbf{E}}}\tilde{\boldsymbol{\alpha}}^k \tag{14}$$

where $\mathbf{I}_l$ is an $l \times l$ identity matrix and $\tilde{\boldsymbol{\alpha}}$ is the kth eigenvector of the matrix $\tilde{\mathbf{K}}$.

As analyzed in [6], it is misleading and wasteful to use full covariances, so only a small number of eigenvectors are sufficient to capture the structural property of the covariance $\mathbf{\Sigma}_{\mathcal{H}}$. Fig. 2 displays an example of a table tennis racket point cloud. Its Gaussian distribution in the feature space $\mathcal{H}$ can be well reconstructed using only 3 of its principal components associated with top largest eigenvalues.



(a) (b) (c) (d)

Figure 2. (a) A point cloud of table tennis racket; (b–d) reconstruction using the first 1–3 principal components. For each point in the bounding-box volume, the darkness is proportional to the density of the Gaussian in the feature space $\mathcal{H}$.

C. Alignment of Gaussians

Assume the task is to align a point cloud $\mathbf{M}_1 = \{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ with $\mathbf{M}_2 = \{\mathbf{x}_j^{(2)}\}_{j=1}^{l_2}$, instead of computing the optimal alignment in 3D space directly, we can alternatively first align them in feature space, and then project them back to $\mathbb{R}^3$ (section III). With the modeling procedure above applied on $\mathbf{M}_1$ and $\mathbf{M}_2$, the alignment of two point clouds in feature space corresponds to aligning two Gaussians. In this paper, we assume $\mathbf{D}$ eigenvectors are computed for the covariance of each point cloud: $\tilde{\mathbf{U}}_1 = \left[\tilde{\mathbf{u}}_1^1, \ldots, \tilde{\mathbf{u}}_1^k, \ldots, \tilde{\mathbf{u}}_1^{\mathbf{D}}\right]$, $\tilde{\mathbf{U}}_2 = \left[\tilde{\mathbf{u}}_2^1, \ldots, \tilde{\mathbf{u}}_2^k, \ldots, \tilde{\mathbf{u}}_2^{\mathbf{D}}\right]$. Therefore, the rotation in feature space $\mathbf{R}_{\mathcal{H}}$ is estimated by simultaneously aligning $\mathbf{D}$ pairs of corresponding eigenvectors: $\tilde{\mathbf{U}}_2 = \mathbf{R}_{\mathcal{H}}\tilde{\mathbf{U}}_1$. Because different eigenvectors of each point cloud are orthogonal to each other, based on the computation result in (14), it is easy to derive:

$$\begin{aligned} \mathbf{R}_{\mathcal{H}} &= \tilde{\mathbf{U}}_2\tilde{\mathbf{U}}_1^\top \\ &= \phi(\mathbf{M}_2)^\top\underbrace{\mathbf{I}_2^{\mathbf{E}}\left(\sum_{k=1}^{\mathbf{D}}\tilde{\boldsymbol{\alpha}}_2^k\tilde{\boldsymbol{\alpha}}_1^{k\top}\right)\mathbf{I}_1^{\mathbf{E}}}_{\mathbf{\Theta}_{\boldsymbol{\alpha}}}\phi(\mathbf{M}_1) \end{aligned} \tag{15}$$

Since the rotation (15) can be applied only if $\mathbf{M}_1$ has already been centered in feature space, to fully align two Gaussians, the translation in feature space $\mathbf{b}_{\mathcal{H}}$ obviously should equal the mean of the Gaussian that models $\mathbf{M}_2$ in feature space:

$$\mathbf{b}_{\mathcal{H}} = \boldsymbol{\mu}_{\mathcal{H}}^{(2)} = \frac{1}{l_2}\phi(\mathbf{M}_2)^\top\mathbf{1}_{l_2} \tag{16}$$

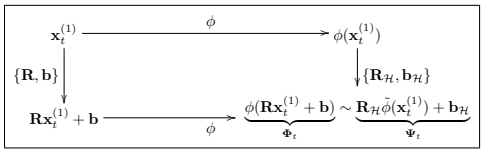


Figure 3. The consistency error is defined as the discrepancy between $\phi(\mathbf{R}x_t + \mathbf{b})$ and $\mathbf{R}_\mathcal{H}\tilde{\phi}(x_t) + \mathbf{b}_\mathcal{H}$.

Now we can align two Gaussians in feature space with $\{\mathbf{R}_\mathcal{H}, \mathbf{b}_\mathcal{H}\}$ computed in (15) and (16). However, due to the infinite-dimensional feature map defined in (4), there still exist two obstacles to be overcome: First, (15) and (16) cannot be computed in an analytic form; secondly, there is no trivial way to map $\{\mathbf{R}_\mathcal{H}, \mathbf{b}_\mathcal{H}\}$ back to 3D space. Fortunately, by designing a consistency error (section III-A), these two issues can surprisingly be solved simultaneously in a very smooth and elegant manner.

III. CARTESIAN POINT CLOUD ALIGNMENT

In this section we will project the rotation and translation in the feature space $\mathcal{H}$ back to 3D space by minimizing a specifically-designed consistency error (Fig. 3). It turns out that the final objective function can be constructed and solved without explicit computation in feature space. In addition, further connections with other registration methods will be exposed by discovering the hidden commonality among their objective functions. To enhance the generality of the proposed algorithm, an $SE(3)$ on-manifold optimization scheme is employed to search for the optimal transformation.

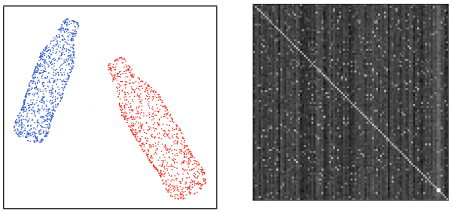
A. Consistency Error

Instead of tediously finding the inverse function $\phi^{-1}(\cdot)$ corresponding to the definition in (4) and applying it to $\{\mathbf{R}_\mathcal{H}, \mathbf{b}_\mathcal{H}\}$ to map them back to 3D space $\{\mathbf{R}, \mathbf{b}\}$, here we define a novel consistency error between $\phi(\mathbf{R}x_t + \mathbf{b})$ and $\mathbf{R}_\mathcal{H}\left(\phi(x_t) - \boldsymbol{\mu}_1\right) + \mathbf{b}_\mathcal{H}$ based on the fact that mapping after transformation should be consistent with transformation after mapping (Fig. 3). Therefore, we can find the optimal rotation and translation in 3D space by minimizing the average consistency error:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \frac{1}{l_1}\sum_{t=1}^{l_1} \|\boldsymbol{\Psi}_t - \boldsymbol{\Phi}_t\|^2 \tag{17}$$

Because $\|\boldsymbol{\Phi}(\boldsymbol{x})\|^2$ is the integration over the square of a Gaussian, which preserves constant under any translation $\mathbf{b}$ and rotation $\mathbf{R}$, and $\boldsymbol{\Psi}_t$ is fixed, by substituting (15) and (16) into (17), we have

$$\begin{aligned}\{\mathbf{R}^*, \mathbf{b}^*\} &= \arg\max_{\mathbf{R},\mathbf{b}} \frac{1}{l_1}\sum_{t=1}^{l_1} \boldsymbol{\Psi}_t^\top \boldsymbol{\Phi}_t \\ &= \arg\max_{\mathbf{R},\mathbf{b}} \underbrace{\frac{1}{l_1}\sum_{t=1}^{l_1} K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)^\top \boldsymbol{\rho}_t}_{\mathbf{O}}\end{aligned} \tag{18}$$



(a) (b)

Figure 4. (a) Two identical point clouds with exactly the same point permutation. (b) Visualization of ρ_{ti} computed for all pairs of points.

$$\boldsymbol{\rho}_t = \boldsymbol{\Theta}_{\boldsymbol{\alpha}}\left(K(\mathbf{x}_t^{(1)}, \mathbf{M}_1) - \frac{1}{l_1}\mathbf{K}_1\mathbf{1}_{l_1}\right) + \frac{1}{l_2}\mathbf{1}_{l_2} \tag{19}$$

where $K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)$ is an l_2-dimensional vector with $K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)_i = K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{x}_i^{(2)})$, and $K(\mathbf{x}_t^{(1)}, \mathbf{M}_1)$ is an l_1- dimensional vector with $K(\mathbf{x}_t^{(1)}, \mathbf{M}_1)_j = K(\mathbf{x}_t^{(1)}, \mathbf{x}_j^{(1)})$. It can be seen that by employing the kernel trick (2), we can elegantly avoid computation in the feature space $\mathcal{H}$ in both (18) and (19).

B. Relation to Other Approaches

As analyzed in section I, most existing registration methods can be unified to a general objective function (1) with different correspondence assumptions or iterative update strategies. The objective function (18) can be easily extended as follows:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{t=1}^{l_1}\sum_{i=1}^{l_2} -K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{x}_i^{(2)})\rho_{t,i} \tag{20}$$

By considering $-K(\cdot,\cdot)$ as an exponential distance and replacing $\rho_{t,i}$ with $w_{t,i}$, it turns out that surprisingly our method (20) is also a special case of (1), although we arrive there from a completely different starting point. This suggests that $\rho_{t,i}$ somehow implicitly encodes the correspondence likelihood between $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_i^{(2)}$ as well. To verify this argument experimentally, in Fig 4(a) there are two identical point clouds with exactly the same point permutation. We compute ρ_{ti} for all pairs of points in Fig 4(b). It can be seen that Fig 4(b) shows a very evident diagonal pattern with uniformly distributed noise, which is the reflection of our prior knowledge. However, different from most of other approaches, we do not model $w_{t,i}$ explicitly or update them iteratively. Instead, $\rho_{t,i}$ is derived from eigenvector alignment in feature space and only need to be computed once.

There is another way our method is related to Gaussian mixtures. By relaxing the non-negative coefficient constraint in the definition of Gaussian mixtures, each eigenvector in (14) can be considered as a pseudo Gaussian mixture with $\phi(\cdot)$ defined as in (4). In this way, instead of aligning two Gaussian mixtures in 3D space, what our method is actually doing is to simultaneously align $\mathbf{D}$ pairs of Gaussian

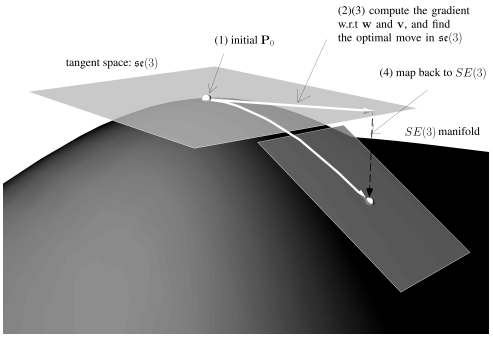


Figure 5. The $SE(3)$ manifold and its optimization scheme: (1) start from a rotation matrix $\mathbf{P}_0$; (2) use equation (26) as the local parametrization of the manifold at point $\mathbf{P}_0$, and compute the gradient with respect to $\{\mathbf{w}, \mathbf{v}\}$; (3) compute the best move in $\mathfrak{se}(3)$ by mapping the update of $\{\mathbf{w}, \mathbf{v}\}$; (4) map back to $SE(3)$: $\mathbf{P}_1 \leftarrow \exp(\Lambda)\mathbf{P}_0$; (5) repeat step (2)(3)(4) until convergence

mixtures in feature space, and then implicitly maps back to original 3D space.

C. SE(3) on-manifold optimization

When solving the optimization problem (18), the orthogonality constraint of the rotation matrix $\mathbf{R}$ must be taken into account: $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$. This constraint is a common obstacle in various rotation-related optimization problems, which drove many researchers to alternatives such as unit quaternions [5] or dual quaternion number [8]. However, although all these methods can work satisfactorily for 3D rotation, they are difficult to be extend to higher dimensions. Although we only focus 3D point cloud registration here, to make our algorithm more general, here we employ an $SE(3)$ on-manifold optimization scheme [9], which can be easily adapted to rotation matrices in any dimension. Another virtue of $SE(3)$ on-manifold optimization is that combined with gradient computing, an elegant optimizer can be developed based on its associated Lie algebra to circumvent the orthogonality constraint.

Each rotation and translation in 3D space $\{\mathbf{R}, \mathbf{b}\}$ can be jointly treated as a Euclidean transformation $\mathbf{P}$ in $\mathbb{R}^3$ by using homogeneous coordinates. From now on, $\mathbf{x}$ is used to denote homogeneous coordinate, and $\overline{\mathbf{x}}$ for the original one:

$$\mathbf{x}^\top = [\overline{\mathbf{x}}^\top, 1] \tag{21}$$

and correspondingly,

$$\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{b} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{22}$$

One specific $\mathbf{P}$ is actually an element of Lie group $SE(3)$ (Special Euclidean Group), which is a smooth manifold embedded in $\mathbb{R}^3$. Intuitively, the $SE(3)$ manifold can be considered as a topological space wherein all points are 4×4 Euclidean transformation matrices, and at each point, there exists a tangent space Λ, which happens to be its associated Lie algebra $\mathfrak{se}(3)$. The mathematical connection between $SE(3)$ and $\mathfrak{se}(3)$ is

$$\mathfrak{se}(3) \to SE(3) : \mathbf{P} = \exp(\Lambda) \tag{23}$$

where $\exp(\cdot)$ denotes the exponential map. The tangent space $\mathfrak{se}(3)$ can be considered as a linearization of the $SE(3)$ manifold within the infinitesimally small vicinity of certain point $\mathbf{P}_0$, so inversely, the exponential map works as a 'de-linearization'. All concepts described above are illustrated in Fig. 5. The Lie algebra $\mathfrak{se}(3)$ is a collection of matrices of the form

$$\Lambda = \begin{bmatrix} J(\mathbf{w}) & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \tag{24}$$

where $J(\mathbf{w})$ is an skew-symmetric matrix, which can be constructed from a 3D vector $\mathbf{w}$ with a skew operator $J(\cdot)$:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \to J(\mathbf{w}) = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \tag{25}$$

and $\mathbf{v}$ is an usual 3D vector. Therefore, Therefore, when $\exp(\Lambda) \to \mathbf{I}_3$ (e.g. computing gradient), we can establish a straightforward map from $\mathbb{R}^6$ to the local neighboring region of $\mathbf{P}_0$ on manifold as

$$\mathbf{P} = \exp\left(\begin{bmatrix} J(\mathbf{w}) & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix}\right) \cdot \mathbf{P}_0 \tag{26}$$

Last but not least, combination with a gradient-type method yields an final optimization procedure for $SE(3)$ parameters. By using (26), we can see that when computing the gradient with respect to $\mathbf{w}$ and $\mathbf{v}$, the orthogonality constraint will be avoided, and therefore, the constrained optimization problem in $SE(3)$ (18) is naturally and smoothly transformed to a much simpler, unconstrained problem in $\mathbb{R}^6$. Meanwhile, different from conventional gradient methods, instead of computing gradient and updating within the same space, in $SE(3)$ on-manifold optimization, after every update of $\{\mathbf{w}, \mathbf{v}\}$, it need to be mapped back to $SE(3)$, and subsequently the gradient is computed with the local parametrization of the corresponding neighboring region. The whole procedure of $SE(3)$ on-manifold optimization scheme is illustrated in Fig. 5.

D. Reduction of Computational Complexity

If we reexamine the objective function $\mathbf{O}$ (18), an interesting property can be leveraged to significantly reduce the computation complexity:

$$\begin{aligned} & \langle \mathbf{\Phi}_t, \mathbf{\Psi}_t \rangle \\ = \; & \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}})^\top \left(\sum_{k=1}^{\mathbf{D}} \tilde{\mathbf{u}}_2^k \tilde{\mathbf{u}}_1^{k\top} \left(\overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \right) + \boldsymbol{\mu}_2 \right) \\ = \; & \sum_{k=1}^{\mathbf{D}} \langle \tilde{\boldsymbol{u}}_2^k, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \langle \tilde{\boldsymbol{u}}_1^k, \overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \rangle + \langle \boldsymbol{\mu}_2, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \\ = \; & \sum_{k=1}^{\mathbf{D}} \langle \tilde{\boldsymbol{u}}_2^k, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \langle \tilde{\boldsymbol{u}}_2^k, \mathbf{R}_{\mathcal{H}} \overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \rangle + \langle \boldsymbol{\mu}_2, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \end{aligned} \tag{27}$$

where we can see that $\overline{\phi(\mathbf{P}\mathbf{x}_t^{(1)})}$ and $\overline{\mathbf{R}_{\mathcal{H}}\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1}$ are projected onto $\mathbf{D}$ eigenvectors $\underline{\left\{\tilde{\boldsymbol{u}}_2^k\right\}_{k=1}^{\mathbf{D}}}$ respectively, and an additional projection of $\phi(\mathbf{P}\mathbf{x}_t^{(1)})$ onto $\boldsymbol{\mu}_2$. Therefore, the computation of the objective function is actually done in a space spanned by $\mathbf{D}$ eigenvectors $\left\{\tilde{\boldsymbol{u}}_2^k\right\}_{k=1}^{\mathbf{D}}$ and one $\boldsymbol{\mu}_2$, which is a subspace of $\mathcal{H}$. We denote this subspace by $\mathcal{S}$. The feature map of each point $\phi(\mathbf{x}_i)$ can be projected onto $\mathcal{S}$ in the following way:

$$\mathcal{S}\left(\phi(\mathbf{x}_i)\right) = \sum_{k}^{\mathbf{D}+\mathbf{1}} \beta_{i,k}\mathbf{r}_k \tag{28}$$

where $\mathbf{r}_k$ are referred to as $\mathbf{D}+1$ reference vectors in $\mathcal{S}$, and $\beta_{i,k}$ are the corresponding coefficients used to express $\mathcal{S}\left(\phi(\mathbf{x}_i)\right)$. In other words, in $\mathcal{S}$, only $\mathbf{D}+1$ reference vectors, of which $\mathbf{D}$ should be linearly independent, can represent any $\mathcal{S}(\phi(\mathbf{x}_i))$. Therefore, to ensure that $\boldsymbol{\Phi}_t$ and $\boldsymbol{\Psi}_t$ are consistent with each other for all points $\mathbf{x}_t^{(1)}$ in $\mathbf{M}_1$, we only need to align $\mathbf{D}+1$ predefined reference vectors. In practice, we can randomly select $\mathbf{D}+1$ $\mathcal{S}(\phi(\mathbf{x}_i))$ because they are very likely to be linear independent in $\mathcal{S}$. Thus, the objective function can be simplified to

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\max_{\mathbf{R},\mathbf{b}} \frac{1}{\mathbf{D}+1} \sum_{t=1}^{\mathbf{D}+1} K(\mathbf{R}\mathbf{x}_{\mathbf{S}_t}^{(1)} + \mathbf{b}, M_2)^\top \boldsymbol{\rho}_t \tag{29}$$

where $\mathbf{S}$ denotes the randomly selected subset of $\mathbf{M}_1$. To practically apply $SE(3)$ on-manifold optimization on the objective function $\mathbf{O}$ (29), we compute the derivatives of (29) w.r.t. $\mathbf{w}$ and $\mathbf{v}$ as follows:

$$\frac{d\mathbf{O}}{d\left[\mathbf{w}^\top, \mathbf{v}^\top\right]^\top} = \frac{1}{\mathbf{D}+1} \sum_{t=1}^{\mathbf{D}+1} \left(\frac{d\mathbf{O}}{d\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}} \frac{d\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{d\left[\mathbf{w}^\top, \mathbf{v}^\top\right]^\top} \right) \tag{30}$$

where

$$\frac{d\mathbf{O}}{d\mathbf{P}\mathbf{x}_t^{(1)}} = \sum_{j=1}^{l_2} \rho_{\mathbf{S}_t,j} K\left(\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}, \mathbf{x}_j^{(2)}\right) \frac{1}{\sigma^2} \left(\mathbf{x}_j^{(2)} - \mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}\right)^\top \tag{31}$$

$$\frac{\partial \mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial \mathbf{w}} = \frac{\partial \exp(\Lambda)\mathbf{P}_0\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial \mathbf{w}} = \left[\overline{J(\mathbf{P}_0\mathbf{x}_{\mathbf{S}_t}^{(1)})}, \mathbf{0}_{3\times 1}\right]^\top \tag{32}$$

$$\frac{\partial \mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial \mathbf{v}} = \frac{\partial \exp(\Lambda)\mathbf{P}_0\mathbf{x}_t^{(1)}}{\partial \mathbf{v}} = \left[\mathbf{I}_3, \mathbf{0}_{3\times 1}\right]^\top \tag{33}$$

IV. EXPERIMENTS

A. Implementation details

Since the computed eigenvectors (14) are of no direction, there could be $2^{\mathbf{D}}$ possible alignments for $\mathbf{D}$ eigenvectors in feature space. Fortunately, according to experiment, we found that $\mathbf{D} = 3$ is actually enough to make sufficiently good alignment. Therefore, one has to compute all 8 possible alignments in feature space and project them back to $\mathbb{R}^3$, then the final optimal one is picked by checking the accumulated distances between every pair of corresponding points in two clouds, and we use shortest distance as the correspondence here. An outline of the proposed algorithm is given in Algorithm 1. In practice, to speed up the convergence of the algorithm, some sophisticated stepsize tricks can also be added. In addition, we also find that in Algorithm 1 computing eigenvectors (line 2) is the most time-consuming part, so in our implementation, fast-PCA [10] is employed to accelerate the computation.

Algorithm 1 3D Point Cloud Registration

Input: $\mathbf{M}_1 = \{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ and $\mathbf{M}_2 = \{\mathbf{x}_j^{(2)}\}_{j=1}^{l_2}$, $\mathbf{x} \in \mathbb{R}^3$
Output: the optimal motion estimation $\mathbf{P}^*$ which can align $\mathbf{M}_1$ with $\mathbf{M}_2$

1: construct two matrices $\tilde{\mathbf{K}}_1$ and $\tilde{\mathbf{K}}_2$ (13)
2: compute eigenvalue-eigenvector pairs for $\tilde{\mathbf{K}}_1$ and $\tilde{\mathbf{K}}_2$: $\{\boldsymbol{\alpha}_{m,k}, \eta_{m,k}\}$ $m = 1, 2$
3: normalize eigenvectors (12)
4: select $\mathbf{D} = 3$ eigenvectors with largest eigenvalues for both $\mathbf{M}_1$ and $\mathbf{M}_2$
5: randomly select a subset of $\mathbf{N} \geq \mathbf{D}+1$ size from $\mathbf{M}_1$
6: set initial $\mathbf{P}_0$ randomly
7: compute $\boldsymbol{\Theta}_{\boldsymbol{\alpha}}$ (15) with the subset
8: **while** 1 **do**
9: compute the gradient $\nabla_{\mathbf{w}}$ and $\nabla_{\mathbf{w}}$ with current $\mathbf{P}_n$ (30–33)
10: **if** both $\nabla_{\mathbf{w}}$ and $\nabla_{\mathbf{v}}$ are small enough **then**
11: **return** $\mathbf{P}_n$
12: **end if**
13: map the update of $\mathbf{w}$ and $\mathbf{v}$ back to $SE(3)$ (26)
14: set $n \leftarrow n+1$
15: **end while**
16: repeat line 7–15 $2^{\mathbf{D}}$ times with different sign combinations of eigenvectors, and select the final optimal $\mathbf{P}^*$ which yields the minimal accumulated distances between every pair of closest points in $\mathbf{P}_n\mathbf{M}_1$ and $\mathbf{M}_2$

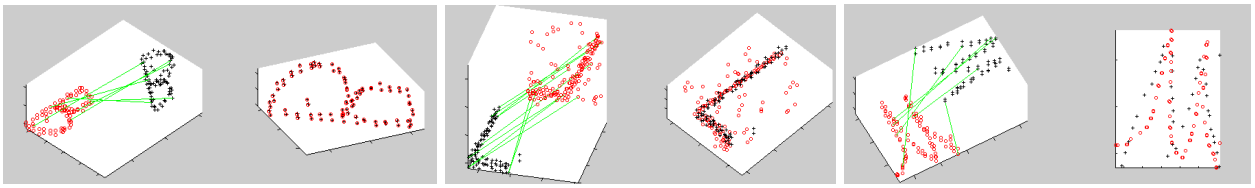
B. Qualitative Evaluation

For the sake of visualization, we first test our algorithm on some toy point clouds to see how it work qualitatively. In Figure 6, some test examples on handwritten letters are displayed. It can be seen that in rather challenging circumstances, i.e. (1) the motion between two point clouds is arbitrarily large (Figure 6(a)), (2) a large portion of outliers are added (Figure 6(b)), (3) nonrigid transformation is applied (Figure 6(c)), the proposed algorithm can still discover roughly correct corresponding points [2] between two point clouds (green lines in Figure 6) and make qualitatively acceptable alignment.

C. Quantitative Evaluation

To obtain a more precise and convincing evaluation of the the proposed algorithm, KIT database [11] is used for

[2] the correspondence for point $\mathbf{x}_t^{(1)}$ is determined by finding the index $j^* = \arg\max_{j\in[1,l_2]} \rho_{tj}$



(a) (b) (c)

Figure 6. Test of the proposed algorithm in typical challenging circumstances for registration: (a) large motion; (b) outliers; (c) nonrigid transformation

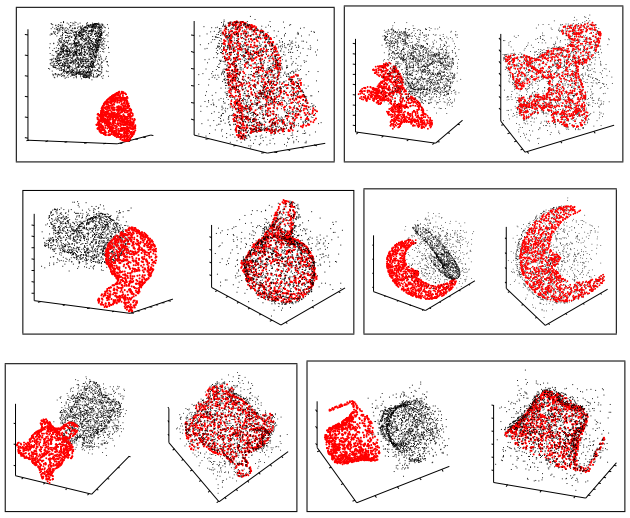
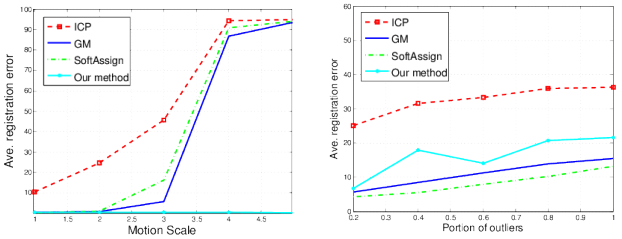


Figure 7. Some test results on KIT 3D object database



(a) (b)

Figure 8. Test of four registration algorithm on (a) different scales of motions; (b) different portion of outliers added.

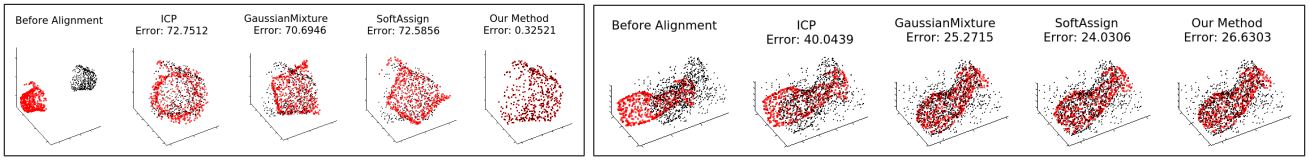
more intensive test (some test results can be seen in Figure 7). In addition, for quantitative comparison, ICP method, Gausssian Mixture(GM) and SoftAssign [3] are implemented as well. To ensure fairness, the same $SE(3)$ on-manifold optimization strategy is employed for their corresponding objective functions. Since the objects in KIT database is in triangulated mesh format, point clouds are generated by first sampling a triangle with the probability proportional to its area and then uniformly sampling a point from the selected triangle.

First, we test the robustness of four algorithms on different scales of motions. In our experiment, for motion scale i, the rotation angles of yaw, pitch and roll are $i \times [30°, 5°, 5°]$, and translations are $i \times [S_x, S_y, S_z]$, where $[S_x, S_y, S_z]$ are standard deviations of point clouds in three axes. Different motions are applied to the point cloud of each object (points cloud is sampled with size 1000) to generate a target point cloud to align with. Since we know the correspondence between the original and target point clouds, the error for each registration is computed as the average distance between every pair of corresponding points in two point clouds. The test result is plotted in Figure 8(a). We can see that four algorithms can work similarly well for small motions (scale $i = 1$). However, as i increases, the accuracy of ICP, GM and SoftAssign decrease, although GM and SoftAssign are much more robust than ICP for intermediate motions (scale $i = [2, 3]$). Our method, by contrast, performs consistently well for all scales of motions. A test example is displayed in Figure 9(a), from which we can see that the instability of ICP, GM and SoftAssign stems from the fact that they are likely to to stuck into local optimum when the motion is large (although the local optimum can be avoided by setting up many intial poses, it would take more time to guarantee that the global optimum is found).

Secondly, we test the robustness of four algorithms by adding different portion (the percentage of point cloud size) of outliers which are randomly sampled within the space around objects. The generated outliers are concatenated into the original point clouds, so the correspondence is still available and the registration error is computed in the same way as in the motion experiment. To avoid the effect of large motion, a relatively small motion (motion scale $i = 1$) is applied to all point clouds. The test result is plotted in Figure 8(b). We can see that SoftAssign is most stable for the case in which outliers are presented, GM and our method are slightly worse, and ICP is very sensitive to outlier even when the portion is small. A test example is displayed in Figure 9(b), from which we can see that except ICP, the result of other three algorithms are acceptable.

Last but never least, efficiency is a significant strength of our method, which enables it can be used for real

[3] the comparison in [12] has reported that SoftAssign and EM-ICP perform similarly, so we are not going to include EM-ICP in our experiment



(a) (b)

Figure 9. Comparison of four registration algorithms: (a) motion scale $i = 5$; (b) outlier portion$= 0.8$.

	100	200	500	1000	2000
Our method	1.172	1.489	2.162	5.126	21.165
ICP [1]	0.012	0.023	0.051	0.154	0.469
GaussianMixtures [5]	1.859	3.998	15.245	43.570	172.4
SoftAssign [3]	2.059	4.801	83.925	592.1	3812

Table I
AVERAGE EXECUTION TIME (SECONDS)

time applications. As we can see in Algorithm 1, after computing eigenvectors for kernel matrices, the complexity of computing optimal motion is linear to the size of points n. Since the complexity of fast PCA is $O(n \log n)$ [10], the overall complexity of Algorithm 1 is $O(n \log n)$. To compare the efficiency, all four algorithms are implemented in Matlab and run on the same hardware platform (usual i7 intel core laptop). Point clouds of all objects are generated with 5 different sizes (100, 200, 500, 1000, 2000), on which four algorithms are tested respectively. For each point cloud, a randomly generated motion is applied and random portion of outliers are added (to get an approximate average). Note that in this experiment we are only concerned about the running time, so the algorithms will stop when they converge even if the registration is bad. The average execution time (in seconds) of four algorithms on five set of point clouds are presented in Table I. We can see that the complexity of our algorithm is the same as ICP $O(n \log n)$, and it is much faster than SoftAssign and Gausssian Mixtures(GM) with complexity $O(n^2)$ (however, SoftAssign is usually more expensive than GM because it needs to iteratively update correspondence matrix).

CONCLUSION

We introduced a novel point cloud registration algorithm based on kernel-induced feature maps, kernel PCA and $SE(3)$ on-manifold optimization. The framework is theoretically elegant, and exhibits robustness and accuracy in fairly challenging circumstances. It is quite general and flexible to be extended to different dimensions and intra-category instances alignment . Remarkably, it outperforms most other methods in terms of efficiency.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

REFERENCES

[1] P. J. Besl and H. D. Mckay, "A Method for Registration of 3-D Shapes," *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[2] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *3DIM*, 2001, pp. 145–152.

[3] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness, "New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence," *Pattern Recognition*, vol. 31, pp. 957–964, 1997.

[4] S. Granger and X. Pennec, "Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration," in *European Conference on Computer Vision (ECCV 2002), volume 2353 of LNCS*. Springer, 2002, pp. 418–432.

[5] B. Jian and B. C. Vemuri, "Robust Point Set Registration Using Gaussian Mixture Models," *PAMI*, vol. 33, no. 8, pp. 1633–1645, 2011.

[6] R. Kondor and T. Jebara, "A Kernel between Sets of Vectors," in *ICML*, 2003.

[7] B. Schlkopf, A. Smola, and K.-R. Mller, "Nonlinear component analysis as a kernel eigenvalue problem," 1996.

[8] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[9] C. J. Taylor and D. J. Kriegman, "Minimization on the Lie Group SO(3) and Related Manifolds," Yale University, Tech. Rep., 1994.

[10] A. Sharma and K. K. Paliwal, "Fast Principal Component Analysis using Fixed-point Algorithm," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1151–1155, 2007.

[11] A. Kasper, Z. Xue, and R. Dillmann, "The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, May 2012.

[12] Y. Liu, "Automatic Registration of Overlapping 3D Point Clouds using Closest Points," *Image and Vision Computing*, vol. 24, no. 7, pp. 762–781, 2006.

Chapter 6

Conclusion

"The true face of Lushan is lost to my sight, for it is right in this mountain that I reside."

Su Shi

---

nterdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Bibliography

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004.

Sanjiv Kumar and Martial Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *In NIPS*, 2003a.

Sanjiv Kumar and Martial Hebert. Man-made structure detection in natural images using a causal multiscale random field. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 119–126, 2003b.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *NIPS*, 2003.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing Free-energy Approximations and Generalized Belief Propagation Algorithms. *IEEE Trans. Inf. Theor.*, 51(7):2282–2312, July 2005.