CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Electrical Engineering
Department of telecommunication engineering

Implementation of Multiple Instance Classification using Markov Networks

January 2020

Bakalant: Branislav Doubek

Supervisor: prof. Jan Kybic

Acknowledgement

I would like to thank my supervisor prof. Kybic for his patience and guidance. I would also like to express my deepest gratitude to my family, for their never-ending support.

Declaration of authorship

I declare that the presented work was developed independently with help of my supervisor prof. Kybic and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 7.1.2020

. . . . . .

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Doubek Branislav** Personal ID number: **459220**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Telecommunications Engineering**

Study program: **Electronics and Communications**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Implementation of Multiple Instance Learning using Markov Networks

Bachelor's thesis title in Czech:

Implementace metody "Multiple instance learning" pomocí Markovských sítí

Guidelines:

Implement learning and classification algorithms based on the article [1] that describes a multiple instance learning method using a graphical model. Test the method on standard datasets to verify the reported results. Apply the method further to classify cancer in the lymph node histology images from the Camelyon 16 challenge [2]. Try to improve the learning by formulating the optimization as a linear programming task.

Bibliography / sources:

[1] HAJIMIRSADEGHI, Hossein; MORI, Greg. Multi-Instance Classification by Max-Margin Training of Cardinality-Based Markov Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39.9: 1839-1852.
[2] LITJENS, Geert, et al. 1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset. GigaScience, 2018, 7.6: giy065.

Name and workplace of bachelor's thesis supervisor:

prof. Dr. Ing. Jan Kybic, Biomedical imaging algorithms, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **14.02.2019** Deadline for bachelor thesis submission: **07.01.2020**

Assignment valid until: **20.09.2020**

prof. Dr. Ing. Jan Kybic Supervisor's signature	Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Abstract
We present implementation of multiple instance learning (MIL) framework based on Markov networks with optimization methods of batch gradient descent, inference based SVM and propose new approach with the use of linear programming. In this paper we successfully validated our implementation on common multiple instance learning datasets and used it for classifying cancer metastases on sets of precalculated descriptors, which were generated from images of histological lymph node sections. Manual classification of individual images by pathologist is a lengthy process and using multiple instance learning to provide another diagnostic tool for classification could increase accuracy and speed of classification which is vital. The results measured on Camelyon dataset with batch gradient descent method were comparable to state-of-art methods.

Keywords: multiple instance learning, supervised learning, machine learning, Markov networks

Abstrakt
Tato práce pojednává implementaci multiple instance learning frameworku používající metody Markovských sítí s optimalizačními metodami batch gradient descent, inference based SVM a naší metodou linearního programování. Úspešně jsme ověřili naší implementaci na multiple instance learning datasetech, kterou jsme následně použili pro klasifikaci buněk rakoviny z předpočítaných deskriptorů, které byly vygenerovány z histologických fotek lymfatických uzlin. Manuální klasifikace individuálních fotek je prováděna patology, nicméně se jedná o zlouhavý proces a použití metody multiple instance learning, jakožto dalšího diagnostického nástroje pro klasifikaci rakoviny, který by zvýšil přesnost a rychlost klasificace je vitální. Výsledky, které byly naměřeny na Camelyon datasetu s optimalizační metodou batch gradient decent byly srovnatelné se state-of-art metodami.

Klíčová slova: multiple instance learning, supervised learning, strojové učení, Markovské sitě

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

MIL	Multiple Instance Learning
SVM	Support Vector Machine
MIMN	Multiple Instance Markov Networks
RMIMN	Ratio-constrained Multiple Instance Markov Networks
GMIMN	Generalized Multiple Instance Markov Networks

1 Introduction

In classical supervised learning the classifier is being trained on pre-labeled training instances. More training data usually means more accurate prediction, but the preparation of the dataset to train data on can be arduous, since someone beforehand needs to go through all the data and check, if all data points are correctly labeled to their corresponding classes, which can be a limiting factor.

Multiple instance learning is a variation to classical supervised learning, being able to learn on weakly supervised data by batching a group of unlabeled individual data points (i.e. descriptors) into sets called bags with known label.

Multiple instance learning has been introduced by Dietterich [1] who used Multiple instance learning approach for classification of aromatic molecules. Each molecule is called Musk (aromatic) if at least one of its molecular shapes binds to a specific receptor, otherwise it is non-Musk molecule. Other applications of multiple instance learning can be found in an stock prediction [2], where each positive bag contains 100 individual stocks (i.e. instances), which yielded highest return and negative bag contains 5 the worst performing stocks. Another application include video annotation [3], where each video is transformed into bag, in which instance denotes single frame of the video. Last but not least multiple instance applications can be found in region based image annotation [4], online object tracking [5], text categorization[6] and recognition of handwriting and speech[7].

1.1 Problem definition

Multiple instance learning methods deal with weakly supervised data, which consist of sets of instances, called *bags*. The label of a bag is known, where as labels of individual instances are not.

The aim of multiple instance learning is to infer patterns from training dataset and to train mapping function (i.e. classifier), which assigns correct class label $Y \in \{-1, 1\}$ to testing bag $\mathbf{X}$ containing some arbitrary count of precalculated descriptors $\mathbf{x}$.

The importance of multiple instance learning lies in ability to learn

from weakly supervised data, which is useful in some computer and medicine applications, where it is sometimes too expensive or borderline impossible to annotate dataset.

Detecting presence of cancer cells from histological images of breast lymph nodes is a delicate process in which groups of trained pathologists search through hyper resolution images for signs of cancer cells. This method is slow and inefficient due to time and training requirements needed to perform individual classification, which led to creation of Camelyon dataset[8] containing a multitude of weakly annotated histological images.

Considering the nature of the problem, in which even small counts of cancer cells can signify presence of cancer and inability to label individual pixels, we have decided to use multiple instance learning framework described [9], which is able to not only learn from individual data points, but also to learn on aggregations of cardinality (i.e. count) of positive instances in training bags.

Providing efficient and accurate framework, would be able to decrease the amount of time and resources spent on individual classification, which would lead to faster diagnosis and treatment.

1.2 Scope of thesis

The aim of this thesis is to implement multiple instance learning framework based on Markov networks, optimization methods of inference based SVM and batch gradient descent and newly proposed method, which uses linear programming. Last aim of our thesis is to validate implemented methods and explore performance of our implementation on Camelyon and multiple instance learning datasets.

1.3 Thesis structure

This thesis is organized as follows: In next section we provide introduction to supervised learning, and overview of categories of multiple instance learning methods. Section 3 is used for description of framework proposed in [9]. In section 4 we describe optimization methods of batch gradient descent, inference based SVM and propose new optimization method with the use of linear programming. Followed by

description of multiple instance learning datasets, which we used to validate and evaluate performance of our implementation in section 5. Last but not least we describe results we measured in section 6 and end with a conclusion.

2 Theoretical background

This section is used to briefly overview some techniques which are necessary for understanding multiple instance learning and our thesis. We start by briefly introducing supervised and various categories of multiple instance learning.

2.1 Supervised learning

Supervised learning is one of the types of machine learning. In supervised learning training dataset contains pairs $(x_i, y_i) \in R^d$ x $\{-1, 1\}$. Each pair consists of data point $\mathbf{x}_i$ with corresponding label y_i.
The task of supervised learning is to observe and infer patterns from training dataset and to find mapping function (i.e. classifier), which correctly assigns label y to data point $\mathbf{x}$ from testing dataset.

2.2 Multiple instance learning

In this subsection we review different categories of multiple instance learning and provide examples of state-of-art methods for each of them.

2.2.1 Multiple instance assumptions

Standard multiple instance assumption was introduced by Dietterich [1] and defines positive bag as set containing at least 1 positive instance and negative bag as set containing only negative instances. Dietterich's proposed method called APR uses axis-parallel hyper-rectangles to allocate area, containing at least 1 positive instance from each positive bag and no instances from negative bags.
While this assumption naturally describes problem of aromatic molecules, it is unable to model cardinality based aggregations in bag, which might be crucial in predicting correct bag label. Other methods, which use standard multiple instance assumption are Diverse density[10], mi-SVM[6] and MI-SVM[6]
Ratio-constrained assumption softens standard assumption by describing positive bag as set, in which ratio of positive instances to

all instances exceeds threshold value ρ. Methods, which used non-standard multiple instance assumption in their approach are AL-SVM[11], ALP-SVM[11]. Work proposed in [12], introduced general multiple instance assumption represented cardinality potential. The concept of generalized multiple instance assumption is also used in [9].

2.2.2 Instance space methods

Instance space methods aim to solve problem of multiple instance learning by training instance-based classifier on individual instances from bag and combining multiple instance learning assumptions to create bag-based classifier. One example of instance based methods are diverse density algorithm introduced in [2], which tries to solve multiple instance learning problem by finding point in space, which measures how many positive instances from positive bags are nearby and how far are all instances from negative bags. Other examples of instance based methods are modified SVM-like algorithms mi-SVM[6], MI-SVM[6] and MICA[13].

2.2.3 Bag space methods

Different way of solving multiple instance learning problem provides bag space approach, in which whole bag is considered an object, from which global representations or discriminative information is extracted[9].
The main approach is to calculate descriptors from entire bags, which are then used to train traditional instance-based classifier. Some examples of bag-space methods are MI-Kernel[14], MI-Graph[15] and EM-DD[10].

3 Method

This section is being used for description of already existing method of Markov networks proposed in [9].
We start by introducing formal notation and follow by describing framework based on Markov networks.

3.1 Notation

Similar notation is being used as in [9] for easier orientation in equations.
Let R^d denote instance space of our dataset. Training set consists of pairs $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n), \ldots, (\mathbf{X}_N, Y_N)\}$. Each pair is composed of set of instances called *bag* $\mathbf{X}_n = \{\tilde{\mathbf{x}}_{n1}, \ldots, \tilde{\mathbf{x}}_{ni}, \ldots, \tilde{\mathbf{x}}_{nm_n}\}$ (m_n denotes count of instances in n-th bag) and its corresponding *bag label*, $Y_n \in \{-1, 1\}$. Every instance $\tilde{\mathbf{x}}_{ni} \in X_n$ is represented by instance feature vectors $\mathbf{x}_{ni} \in R^d$
In the model of Markov networks every instance $\tilde{\mathbf{x}}_{ni}$ also contains hidden instance label $h_{ni} \in \{0, 1\}$. Collection of all hidden instance labels in *bag* $\mathbf{X}_n$ is denoted as $\mathbf{h}_n = (h_{n1}, \ldots, h_{ni}, \ldots, h_{nm_n})$.

3.2 Scoring function

We use scoring function, which has been introduced in [9] and depicts cost of classification for bag X, instance labels $\mathbf{h}$ and bag label Y. This is done by combining instance potential $\phi^I_{\mathbf{w}}$ and cardinality potential σ_c.
The scoring function is defined as:

$$f_w(\mathbf{X}, \mathbf{h}, Y) = \sum_{i=1}^{m} \phi^I_{\mathbf{w}}(\mathbf{x}_i, h_i) + \sigma_c(\mathbf{h}, Y) \tag{1}$$

where m denotes count of instances in $\mathbf{X}$.
Both of these potentials are represented as linear functions of weights. Instance potential is parameterized by vector of weights $\mathbf{w} \in R^d$ and intercept point b, while cardniality weights $\mathbf{w}^+_c \backslash \mathbf{w}^-_c$ parameterize cardinality potential.

3.3 Instance potential

Instance potential[9] defines compatibility between instance descriptors $\mathbf{x}_i$, instance label h_i, vector of weights $\mathbf{w}$ and intercept point b.

$$\phi_w(\mathbf{x}_i, h_i) = h_i(\mathbf{w}^T\mathbf{x}_i + b) \tag{2}$$

3.4 Cardinality potential

Cardinality potential defines compatibility between count of instance labels $\mathbf{h}$ and bag label Y[9].

$$\sigma_c(\mathbf{h}, Y) = \begin{cases} C^+_{\mathbf{w}}(m^+, m) & \textbf{if: } Y = 1 \\ C^-_{\mathbf{w}}(m^+, m) & \textbf{if: } Y = -1 \end{cases} \tag{3}$$

$$m^+ = \sum_{i=1}^{m} h_i$$

Where C_w is a function describing relations between *bag label* and count of positive and negative instances.

3.4.1 MIMN

Multiple Instance Markov Network uses standard MIL assumption. If at least 1 instance in a bag is labeled as positive, then the whole bag is considered positive. On the other hand the bag is negative only if all of its instances are negative.
This corresponds to:

$$\begin{aligned} C^+_w(m^+, m) &= \begin{cases} -\infty & \textbf{if: } m^+ = 0 \\ w^+_c & \textbf{if: } m^+ \neq 0 \end{cases} \\ C^-_w(m^+, m) &= \begin{cases} -\infty & \textbf{if: } m^+ \neq 0 \\ w^-_c & \textbf{if: } m^+ = 0 \end{cases} \end{aligned} \tag{4}$$

3.4.2 RMIMN

Ratio-constrained multiple instance Markov network[9] is a more robust version of previous definition, since not all MIL problems can

be efficiently described with MIMN definition of the bag, e.g. precalculated descriptors in negative bags can be contaminated by noise, which can be mistaken for positive pattern, which can lead to a lot false positives while testing. In RMIMN definition the ratio of positive to all instances inside of an bag has to be smaller than ρ if the bag is negative and bigger than ρ if bag is positive.

$$
\begin{aligned}
C^{+}_{\mathbf{w}}(m^{+}, m) &= \begin{cases} -\infty & \textbf{if: } 0 \leq \frac{m^{+}}{m} \leq \rho \\ w^{+}_{c} & \textbf{if: } \rho \leq \frac{m^{+}}{m} \leq 1 \end{cases} \\
C^{-}_{\mathbf{w}}(m^{+}, m) &= \begin{cases} -\infty & \textbf{if: } \rho \leq \frac{m^{+}}{m} \leq 1 \\ w^{-}_{c} & \textbf{if: } 0 \leq \frac{m^{+}}{m} \leq \rho \end{cases}
\end{aligned}
\tag{5}
$$

3.4.3 GMIMN

Generalized multiple instance Markov network[9] increases the number of cardinality parameters used to map number of positive instances inside the bag by creating κ positive and negative cardinality weights, which act as a bins in histogram for cardinality aggregations.
Using notation from [9]:

$$
\begin{aligned}
C^{+}_{\mathbf{w}}(m^{+}, m) &= \begin{cases} -\infty & \textbf{if: } m^{+} = 0 \\ \sum_{k=1}^{\kappa} w^{+}_{ck} \mathbb{1}\left(\frac{k-1}{\kappa} < \frac{m^{+}}{m} \leq \frac{k}{\kappa}\right) & \textbf{if: } m^{+} \in \{1, \ldots, m\} \end{cases} \\
C^{-}_{\mathbf{w}}(m^{+}, m) &= \begin{cases} -\infty & \textbf{if: } m - m^{+} = 0 \\ \sum_{k=1}^{\kappa} w^{-}_{ck} \mathbb{1}\left(\frac{k-1}{\kappa} \leq \frac{m^{+}}{m} < \frac{k}{\kappa}\right) & \textbf{if: } m^{+} \in \{0, \ldots, m-1\} \end{cases}
\end{aligned}
\tag{6}
$$

In this thesis we define positive and negative vector of cardinality weights as $\mathbf{w}^{+}_{c} = (w^{+}_{c1}, \ldots, w^{+}_{ck}, \ldots, w^{+}_{c\kappa})$ and $\mathbf{w}^{-}_{c} = (w^{-}_{c1}, \ldots, w^{-}_{ck}, \ldots, w^{-}_{c\kappa})$ for simplification purposes, since RMIMN and MIMN are special cases of this notation for $\kappa = 1$.

3.5 Inference

"The inference problem is to find the best set of instance labels $\boldsymbol{h}^{*}$ *given observed values from the bag* $\boldsymbol{X}$ *and the bag label* Y*."*[9]
The inference algorithm is used to find a vector of instance labels $\mathbf{h}^{*}$, which maximizes scoring function of the bag.

Inference algorithm with cardinality-based potentials have been introduced in [16],[12] and have been proven to be exact in maximizing score of cardinality and instance aggregations based models.
The mathematical definition of inference can be written as:

$$\mathbf{h}^*(\mathbf{X}, Y) = \underset{\mathbf{h}}{\operatorname{argmax}} \sum_{i=1}^{m} \phi^I_{\mathbf{w}}(\mathbf{x}_i, \mathbf{h}_i) + \sigma_c(\mathbf{h}, Y) \tag{7}$$

The pseudo-code for binary inference algorithm is shown below:

Algorithm 1 Inference algorithm for binary classification

procedure INFERENCE FOR BINARY CASE
Initialize $\mathbf{X}$, Y
Calculate instance potential $\phi_{\mathbf{w}}(x_i, 1)$ using (eq. 2) for every instance feature vector.
Set $\phi_{\mathbf{w}} = (\phi_{\mathbf{w}}(x_1, 1), \ldots, \phi_{\mathbf{w}}(x_m, 1))$
Set $\phi^{s(i)}_{\mathbf{w}}$ as sorted $\phi^I_{\mathbf{w}}$ in decreasing order
if Y=1 **then**
Find upper limit of summation $t \leq m$ for which $\sum_{i=1}^{m} \phi^{s(i)}_{\mathbf{w}} C^+_{\mathbf{w}}(i, m-i)$ is maximized using (Equation 7)
if Y=-1 **then**
Find upper limit of summation $t \leq m$ for which $\sum_{i=1}^{m} \phi^{s(i)}_{\mathbf{w}} + C^-_{\mathbf{w}}(i, m-i)$ is maximized using (Equation 7)
Set $h_{s(i)} = 1$ $(\forall s(i) \leq t)$ and $h_{s(i)} = 0$ $(\forall s(i) > t)$
Output: Instance labels $\mathbf{h}$

3.6 Learning

This subsection is being used to explain the learning procedure on training dataset $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n), \ldots, (\mathbf{X}_N, Y_N)\}$.

The optimization process (i.e. learning) is done by minimizing loss criterion with respect to classifier's weights $\mathbf{w}$, intercept point b and positive/negative cardinality weights $\mathbf{w}^+_c/\mathbf{w}^-_c$, while maximizing scoring function for correctly classified bags.
In the next few equations we demonstrate deriving final form of hinge-like loss criterion defined in [9]:
Let $\mathbf{h}^+_n$ be an output of inference algorithm run on bag $\mathbf{X}_n$ and its bag label Y_n and $\mathbf{h}^-_n$ be an output of inference algorithm run on opposite polarity of Y_n.
For simplification purposes we create variable:

$$\Delta h_{ni} = h^-_{ni} - h^+_{ni} \tag{8}$$

Which denotes difference of instance labels found by inference algorithm for different polarities of Y_n.
We also define variable ζ_n for each training bag X_n in dataset:

$$\zeta_n = \max \begin{cases} 0 \\ 1 + f_w(\mathbf{X}_n, \mathbf{h}_n, -Y_n) - f_w(\mathbf{X}_n, \mathbf{h}_n, Y_n) \end{cases} \tag{9}$$

which is equivalent to:

$$\zeta_n = \max \begin{cases} 0 \\ 1 + \sum_{i=1}^{m_n} \phi_{\mathbf{w}}(\mathbf{x}_{ni}, \Delta h_{ni}) + \sigma_C(\mathbf{h}_n^-, -Y_n) - \sigma_C(\mathbf{h}_n^+, Y_n) \end{cases} \tag{10}$$

where ζ_n denotes a slack variable, which measures difference between scoring functions for incorrectly labeled bag and correctly labeled bag. If $\sum_{n=1}^{N} \zeta_n$ is equal to 0, scoring function of each correctly labeled bag achieved higher cost then scoring function of incorrectly labeled bag. We then rewritte original loss function defined in [9] into:

$$\min_{\mathbf{w},b,\mathbf{w}_c^+,\mathbf{w}_c^-} J = \min_{\mathbf{w},b,\mathbf{w}_c^+,\mathbf{w}_c^-} \sum_{n=1}^{N} \zeta_n + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{11}$$

Where $\|w\|$ denotes ℓ_2 regularization norm, which forces optimization algorithms to iteratively decrease values of weights $\mathbf{w}$.

3.7 Classification

The process of assigning bag label Y to a bag $\mathbf{X}$ is called classification and is done by finding hidden instance labels $\mathbf{h}$ with inference algorithm for both classes and using them to calculate score function (using eq. 1) of bag $\mathbf{X}$ for corresponding Y. Bag label Y is determined by choosing label, for which score function achieved the highest value.

4 Optimization

This section describes various optimization algorithms we used to minimize loss function J defined in eq. 11.

4.1 inference based SVM

We implemented inference based SVM method described in [9] the performance of which should be comparable to mi-SVM[6]. Learning(i.e. optimizing) the classifier is done by altering between finding optimal instance labels $\mathbf{h}^*$ using inference algorithm across all training bags and calculating $\mathbf{w}^*, b^*$ on all pairs comprised of feature instance vector $\mathbf{x}$ and its corresponding instance label h^*. This is done until maximum number of iterations has been reached, or until collection of all instance labels $\mathbf{h}^*$ stops changing between iterations. During training of the classifier cardinality weights $\mathbf{w}_c^+/\mathbf{w}_c^-$ are set to 0 and not trained.

Algorithm 2 Inference based SVM

procedure INFERENCE BASED BINARY SVM
  Initialize $\mathbf{w}$ as random vector with size d
  Set $b \leftarrow 0$
  while *labels* have changed **and** *iteration* $<$ *max iterations* **do**
    for each $n \in$ N **do**
      Use inference algorithm (eq. 7) with $\mathbf{X}_n, Y_n$ to find $\mathbf{h}_n^*$
    Calculate SVM solution across all pairs $(\mathbf{X}_n, \mathbf{h}_n^*)$
    Let $\mathbf{w}^*, b^*$ be an output of SVM
    Let $\mathbf{w} \leftarrow \mathbf{w}^*$ and $b \leftarrow b^*$
Output: Optimized weights $\mathbf{w}$ and intercept point b

4.2 Batch gradient descent with momentum

Batch gradient descent is an iterative algorithm used for optimizing differentiable objective functions by iteratively optimizing classifier's variables . Minimization of loss function J is done by moving with a step size η in an opposite direction of the objective function J's gradient. We use momentum variation of gradient descent for faster minimization of loss function J.
For simplification purposes, we introduce vector $\mathbf{z}$, created by con-

catenating all classifier's variables as follows:

$$\underset{1\times(d+1+2\kappa)}{\mathbf{z}} = (\mathbf{w}, b, \mathbf{w}_c^+, \mathbf{w}_c^-) \tag{12}$$

Which we then use in batch gradient descent method with momentum

$$\begin{aligned} \mathbf{v}^{\tau+1} &= \alpha\mathbf{v}^{\tau} - \eta\nabla J(\mathbf{z}^{\tau}) \\ \mathbf{z}^{\tau+1} &= \mathbf{z}^{\tau} + \mathbf{v}^{\tau+1} \end{aligned} \tag{13}$$

Where $\mathbf{v}^{\tau+1}$ is a momentum vector calculated in τ iteration, $\mathbf{z}^{\tau+1}$ contains solution of τ iteration, η is a step size, α is momentum parameter, which we used a cross-validation parameter and ∇J denotes gradient of objective function:

$$\nabla J = \Big(\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d}, \frac{\partial J}{\partial b}, \frac{\partial J}{\partial \mathbf{w}_c^+}, \frac{\partial J}{\partial \mathbf{w}_c^-}\Big)$$

Partial derivations of the objective function J are listed below:

$$\frac{\partial J}{\partial w_j} = \lambda w_j + \sum_{n=1}^{N} \frac{\partial \zeta_n}{\partial w_j}$$

$$\frac{\partial \zeta_n}{\partial w_j} = \begin{cases} 0 & \textbf{if: } \zeta_n \leq 0 \\ \sum_{i=1}^{m_n} \Delta y_{ni} x_{nij} & \textbf{if: } \zeta_n > 0 \end{cases} \tag{14}$$

$$\frac{\partial \zeta_n}{\partial b} = \begin{cases} 0 & \textbf{if: } \zeta_n \leq 0 \\ \sum_{i=1}^{m_n} \Delta y_{ni} & \textbf{if: } \zeta_n > 0 \end{cases} \tag{15}$$

where x_{nij} denotes j-th element from i-th instance feature vector of bag $\mathbf{X}^n$ and m_n denotes count of instances in the same bag.

$$\frac{\partial \zeta_n}{\partial w_{ck}^+} = \begin{cases} 1 & \textbf{if: } Y^n = -1 \wedge \zeta_n > 0 \wedge \mathbb{1}\Big(\frac{k-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^-}{m_n} \leq \frac{k}{\kappa}\Big) \\ -1 & \textbf{if: } Y^n = 1 \wedge \zeta_n > 0 \wedge \mathbb{1}\Big(\frac{k-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^+}{m_n} \leq \frac{k}{\kappa}\Big) \\ 0 & \textbf{if: } \zeta_n \leq 0 \end{cases} \tag{16}$$

$$\frac{\partial \zeta_n}{\partial w_{ck}^-} = \begin{cases} -1 & \textbf{if: } Y^n = -1 \wedge \zeta_n > 0 \wedge \mathbb{1}\Big(\frac{k-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^+}{m_n} \leq \frac{k}{\kappa}\Big) \\ 1 & \textbf{if: } Y^n = 1 \wedge \zeta_n > 0 \wedge \mathbb{1}\Big(\frac{k-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^-}{m_n} \leq \frac{k}{\kappa}\Big) \\ 0 & \textbf{if: } \zeta_n \leq 0 \end{cases} \tag{17}$$

κ denotes size of vector $\mathbf{w}_c^+/\mathbf{w}_c^-$ (for RMIMN and MIMN definitions of cardinality potential $\kappa = 1$), h_{ni}^+, resp. h_{ni}^- denotes i-th instance

label in found with inference algorithm on bag $\mathbf{X}^n$ with label Y^n, resp. $-Y^n$.
Pseudo-code for BGD is described below.

Algorithm 3 Batch gradient descent

procedure BGD
Initialize *last loss* $\leftarrow \infty$
Initialize $\mathbf{w}$ as random vector with size d
Set $b \leftarrow 0$
Set $\mathbf{v}$ as zero vector with size d
Set $\mathbf{w}_c^- \leftarrow 0$
Set $\mathbf{w}_c^+ \leftarrow 0$
while iteration $\leq$ *max iterations* **and** *last loss* < 1 **do**
Initialize *total loss* $\leftarrow 0$
for Every n $\in$ N **do**
Calculate bag loss ζ_n with eq. 10
if $\zeta_n > 0$ **then**
total loss $\leftarrow$ *total loss* $+ \zeta_n$
Calculate partial derivatives of loss function J with respect to its weights $\mathbf{w}$ using eq. 14, intercept point b using eq. 15, positive cardinality weights $\mathbf{w}_c^+$ using eq. 16 and negative cardinality weights $\mathbf{w}_c^-$ using eq. 17
Concatenate all partial derivatives into vector $\mathbf{z}$ defined in eq. 12
Multiply each element of $\mathbf{w}$ by regularization parameter λ and add the result to corresponding element in $\mathbf{z}$
if *last loss* $\leq$ *total loss* **then**:
Calculate $\mathbf{v}$,$\mathbf{w}, b, w_c^+, w_c^-$ using eq. 13
Set *last loss* $\leftarrow$ *total loss*
else
Set $\eta \leftarrow \frac{\eta}{10}$
Output: $\mathbf{w}, b, \mathbf{w}_c^+, \mathbf{w}_c^-$

4.3 Linear programming

We propose new approach for optimizing eq. 11 with the use of linear programming. We change regularization norm from ℓ_2 to ℓ_1, which transforms original loss function into:

$$\begin{aligned} \min_{\mathbf{w},b,\mathbf{w}_c^+,\mathbf{w}_c^-,\zeta_1,\ldots,\zeta_n} \quad & \frac{\lambda}{2}\sum_{j=1}^{d}|\mathbf{w}_j| + \sum_{n=1}^{N}\zeta_n \\ \text{s.t.} \quad & \Delta h_{ni}(\mathbf{w}^T \cdot \mathbf{x}_{ni} + b) + \sigma_c^- - \sigma_c^+ - \zeta_n \leq 1 \\ & \zeta_n \geq 0 \\ & \forall n = 1, \ldots, N \quad \forall i = 1, \ldots, m_n \end{aligned} \tag{18}$$

Where Δh_{ni} denotes result of eq. 8, σ_c^-/σ_c^+ denotes cardinality potential defined in eq. 3.
We rewrite eq. 18 into matrix form:

$$\begin{aligned} \min_{\tilde{\mathbf{z}}} \quad & \mathbf{c}^T\tilde{\mathbf{z}} \\ \text{s.t.} \quad & \mathbf{G}\tilde{\mathbf{z}} \leq \mathbf{a} \end{aligned} \tag{19}$$

where:

$$\tilde{\mathbf{z}} = (\mathbf{w}, b, \mathbf{w}_c, \mathbf{w}_c, \zeta_1, \ldots, \zeta_N, \mathbf{t})$$

$\mathbf{t}$ is a variable used as a substitute for $|\mathbf{w}|$ in objective function.
Mathematically:

$$\begin{aligned} & t_j = \left|w_j\right| \\ \text{s.t.} \quad & t_j \geq w_j \\ & t_j \geq -w_j \\ & \forall j \leq d \end{aligned}$$

This substitution trick transforms originally non-linear objective function into linear form, which can be solved with linear programming methods.
We define following vector:

$$\underset{1\times(2d+1+2\kappa+n)}{\mathbf{c}^T} = \left(\underset{1\times(d+1+2\kappa)}{\mathbf{0}} \quad \underset{1\times(n+d)}{\mathbf{1}}\right) \tag{20}$$

where κ denotes length of cardinality weights $\mathbf{w}_c^+ \backslash \mathbf{w}_c^-$.
We then rewrite first constraint from eq. 18 into vector Q_n:

$$\underset{1\times(2+1+2\kappa)}{\mathbf{Q}_n} = \left(\textstyle\sum_i^{m_n} \Delta h_{ni} x_{ni1} \quad \ldots \quad \sum_i^{m_n} \Delta h_{ni} x_{nid} \quad \sum_i^{m_n} \Delta h_{ni} \quad \underset{1\times\kappa}{\mathbf{u}_n^+} \quad \underset{1\times\kappa}{\mathbf{u}_n^-}\right) \tag{21}$$

$$\underset{\kappa\times 1}{\left(\mathbf{u}_n^+\right)^T} = \begin{pmatrix} Y^n \mathbb{1}\left(0 < \frac{\sum_i^{m_n} h_{ni}^+}{m_n} \leq \frac{1}{\kappa}\right) \\ \vdots \\ Y^n \mathbb{1}\left(\frac{k-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^+}{m_n} \leq \frac{k}{\kappa}\right) \\ \vdots \\ Y^n \mathbb{1}\left(\frac{\kappa-1}{\kappa} < \frac{\sum_i^{m_n} h_{ni}^+}{m_n} \leq 1\right) \end{pmatrix}$$

$$\underset{\kappa\times 1}{\left(\mathbf{u}_n^-\right)^T} = \begin{pmatrix} -Y^n\mathbb{1}\left(0 \le \frac{\sum_i^{m_n} h_{ni}^-}{m_n} < \frac{1}{\kappa}\right) \\ \vdots \\ -Y^n\mathbb{1}\left(\frac{k-1}{\kappa} \le \frac{\sum_i^{m_n} h_{ni}^-}{m_n} < \frac{k}{\kappa}\right) \\ \vdots \\ -Y^n\mathbb{1}\left(\frac{\kappa-1}{\kappa} \le \frac{\sum_i^{m_n} h_{ni}^-}{m_n} < 1\right) \end{pmatrix}$$

For RMIMN and MIMN cardinality potentials(κ=1) vectors $\mathbf{u}^+$ and $\mathbf{u}^-$ will transform into scalar values Y_n and $-Y_n$.
We then compose constraint matrices $\mathbf{G}$ and $\mathbf{a}$:

$$\underset{(2n)\times(2d+1+2\kappa+n)}{\mathbf{G}} = \begin{pmatrix} \underset{n\times(d+1+2\kappa)}{\mathbf{Q}} & \underset{n\times n}{diag(\mathbf{-1})} & \underset{n\times d}{\mathbf{0}} \\ \underset{n\times(d+1+2\kappa)}{\mathbf{0}} & \underset{n\times n}{diag(\mathbf{-1})} & \underset{n\times d}{\mathbf{0}} \\ \underset{d\times d}{diag(\mathbf{1})} & \underset{d\times(1+2\kappa+n)}{\mathbf{0}} & \underset{d\times d}{diag(\mathbf{-1})} \\ \underset{d\times d}{diag(\mathbf{-1})} & \underset{d\times(1+2\kappa+n)}{\mathbf{0}} & \underset{d\times d}{diag(\mathbf{-1})} \end{pmatrix} \tag{22}$$

$$\underset{(2d+1+\kappa+n)\times n}{\mathbf{Q}^T} = (Q_1 \quad Q_2 \quad \dots \quad Q_n)$$

$$\underset{1\times(2n+2d)}{\mathbf{a}^T} = (\underset{1\times n}{\mathbf{-1}} \quad \underset{1\times(n+2d)}{\mathbf{0}}) \tag{23}$$

We formulate the pseudo-code for linear programming method as follows:

Algorithm 4 Linear programming optimization method

procedure LINEAR PROGRAMMING
Initialize $\mathbf{w}$, b, $\mathbf{w}_c^+$, $\mathbf{w}_c^+$ and learning rate η
Let *best loss* $\leftarrow \infty$
while *iteration* $<$ *max iterations* **and** *best loss* < 1 **do**
Let $\mathbf{z}$ denote classifier's variables
Let *total loss* $\leftarrow 0$
for each pair $(\mathbf{X}^n, Y^n)$ in training dataset **do**
Find instance labels $\mathbf{h}_n^+$, $\mathbf{h}_n^-$ using Equation 7
Calculate $\Delta\mathbf{h}_n$ with eq. 8
Append matrix $\mathbf{Q}_n$ calculated using eq. 21
Calculate $\mathbf{c}$, $\mathbf{G}$, $\mathbf{a}$ using eq. 20, eq. 22 and eq. 23
Calculate solution for z with matrices $\mathbf{c}$, $\mathbf{G}$, $\mathbf{h}$
Let $\tilde{\mathbf{z}}^{\tau+1}$ be an output of linear programming solver
Calculate *last loss* function using objective function in eq. 18
if *best loss* $\leq$ *last loss* **then**:
Set $\tilde{\mathbf{z}}_{best} \leftarrow \tilde{\mathbf{z}}^{\tau+1}$
Set *last loss* $\leftarrow$ *best loss*

Output: Vector of optimized classifier's variables $\tilde{\mathbf{z}}_{best}$

5 Datasets

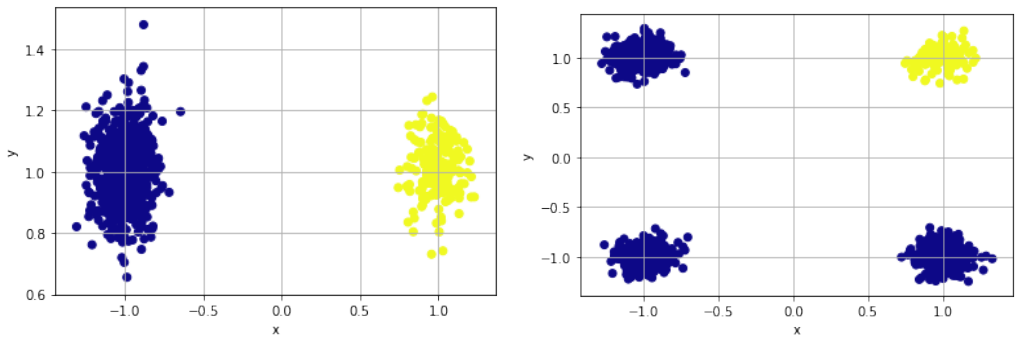
This section is used for description of training datasets, which we used for validating and evaluating performance of our framework's implementation and optimization algorithms.

5.1 Synthetic datasets

Synthetic datasets were generated for purposes of validation and visualization of inference and optimization algorithms.

5.1.1 Gaussian noise dataset

We generated 2 versions of Gaussian noise synthetic dataset. First simpler version of noise dataset was created from data points generated as 2 two-dimensional noises, with y-axis mean in both Gaussian noises being positive and x-axis mean having opposite polarity. Second, more complex version was created using data points generated from 4 two-dimensional Gaussian noises (this dataset has been introduced in [17]), with each Gaussian noise having its means situated in different corner of centered square with length of its side equal to 2 (Scatter plot of signals can be seen in Figure 1).
All data points with positive mean in both axes were set to be positive and rest as negative. Synthetic multiple instance learning datasets were created by batching individual data points into bags by randomly sampling our artificial dataset. As in noisy image dataset we created MIMN and RMIMN versions for both complexities of our dataset. RMIMN version of noise dataset defines positive bag as a set containing at least 50% positive data points. Both datasets consists of 100 positive and 100 negative bags, with 10 instances in each bag.

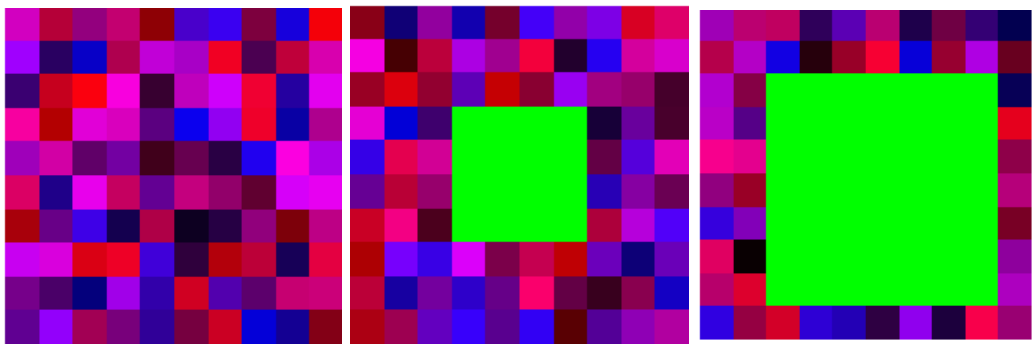


(a) Simple version (b) Complex version

Figure 1: Gaussian noise dataset (yellow denotes positive instances, blue denotes negative instances)

5.1.2 Noisy image dataset

We generated a 200 of a noisy 10x10 images(bags), with green primary color set to 0 for each pixel(instance feature vector $\mathbf{x}$ denotes RGB values of pixel and hidden instance label h denotes label of individual pixel). Positive bag is created by drawing green square in the middle of image. We experiment with both versions of noisy image dataset using MIMN definition(If at least 1 pixel is green, whole image is labeled positive) and RMIMN definition, in which ratio of green pixels in image must be over 50% for bag to be labeled positive.



(a) negative image (b) negative image (c) positive image

Figure 2: Visualization of (a)MIMN and (b)RMIMN(ρ=0.49) version of negative image. Example of positive image can be seen in (c)

5.2 Image datasets

In this subsection we describe 3 feature sets created by[6], which were used to evaluate binary multiple instance learning classifier in im-

age annotation task. The images were taken from Corel dataset and transformed into segments of descriptors describing color, texture and shape of an image. Generated features sets are respectively called Fox, Elephant and Tiger. Image sets contain 100 positive and 100 negative bags and are highly dimensional (instance is described as 230 dimensional instance feature vector).

5.3 Musk datasets

As in almost any multiple instance learning approach Musk1 and Musk2 are benchmark datasets used to evaluate accuracy of the classifier. Every bag contains 6 instances on average in Musk1 and 60 in Musk2, where every instance is described by a 166 dimensional vector. Molecule is labeled as musk if at least 1 of low energy conformations is labeled as positive and non-musk if all conformations are labeled as negative. Further details of Musk1 and Musk2 datasets can be found in [1].

5.4 Camelyon dataset

We provide a brief description of Camelyon dataset introduced in [8] and from which we received generated descriptors, that we used for evaluating performance of our optimization algorithms. Camelyon dataset has been created for detection of cancer metastases from histological images. In our version of the dataset, every histological image(bag) has been sliced into on average 5695 patches(i.e instances) and from each patch neural network generated instance consisting of 32 descriptors.

6 Results

The purpose of this sections is to discuss results received from validation of inference algorithm as well as evaluation of various optimization algorithms on binary multiple instance learning datasets dataset.

6.1 Hyper parameter tuning and testing procedure

K-fold cross validation was used to estimate optimal values for hyper parameters used in our classifier. This was done by running k-fold cross validation across table of possible hyper parameter values and choosing parameter for which classifier achieved highest accuracy on, while other hyper parameters are constant.
Before measuring final performance of our classifier, we used k-fold cross validation on training dataset for finding optimal set of hyper parameters. We were sequentially optimizing parameters in following order: learning rate η, for batch gradient descent momentum parameter α, regularization parameter λ, resp.C and parameters ρ, resp.κ for RMIMN, resp. GMIMN cardinality potentials. After finding optimal set of hyper parameters, we trained classifier with tuned hyper parameters on entire training dataset and measured accuracy on testing dataset. We repeated this process 5 times for different shuffled common multiple instance learning datasets with 75\ 25 training to testing split and calculated overall accuracy and uncertainty. The results from k-fold cross validation can be found in Appendix C.

6.2 Inference validation

Validation process of inference algorithm was done by training instance-based classifier(linear SVM from [18]) on individual data points and their labels on all versions of synthetic datasets, followed by reusing newly found weights $\mathbf{w}$ and intercept point b in our implementation for predicting bag label using inference algorithm on unseen bags. This process was repeated for RMIMN and MIMN versions of synthetic dataset. In RMIMN versions of synthetic datasets we used RMIMN cardinality potential with ρ value set to optimal value. Results can be seen in Table 1.

Synthetic dataset	Accuracy of instance-based classifier[%]
complex MIMN Noise	1.000 ± 0.0
complex RMIMN Noise	1.000 ± 0.0
simple MIMN Noise	1.000 ± 0.0
simple RMIMN Noise	1.000 ± 0.0
MIMN Image	1.000 ± 0.0
RMIMN Image	1.000 ± 0.0

Table 1: Inference validation results on MIMN version of synthetic datasets

From results we can see, that with prior knowledge of count based aggregations in training dataset, inference algorithm is able to find correct patterns in training dataset. We visualize instance labels assigned with inference algorithm in Appendix B.

6.3 Validation of optimization methods

We validated implemented optimization algorithms defined in section 4 for learning on synthetic datasets. The results of validation can be seen in Table 2, Table 3 and Table 4.

Optimization method	simple Noise MIMN	simple Noise RMIMN
batch gradient descent	1.000 ± 0.0	1.000 ± 0.0
inference based SVM	1.000 ± 0.0	1.000 ± 0.0
linear programming	1.000 ± 0.0	1.000 ± 0.0

Table 2: Comparison of accuracies measured with different optimization methods on simple Noise dataset

Optimization method	complex Noise MIMN	Complex noise RMIMN
batch gradient descent	1.000 ± 0.0	1.000 ± 0.0
inference based SVM	1.000 ± 0.0	1.000 ± 0.0
linear programming	1.000 ± 0.0	1.000 ± 0.0

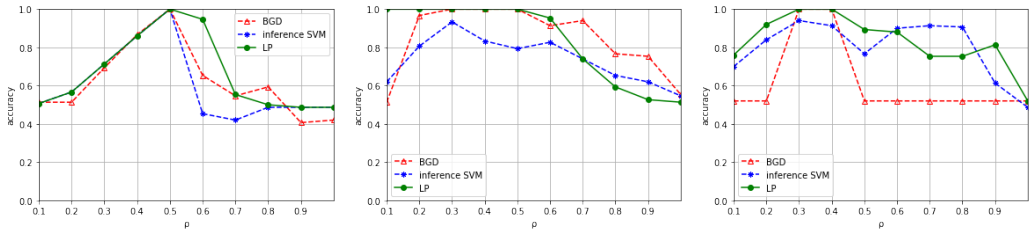
Table 3: Comparison of accuracies measured with different optimization methods on complex Noise dataset

Optimization method	Image MIMN	Image RMIMN
batch gradient descent	1.000 ± 0.0	1.000 ± 0.0
inference based SVM	1.000 ± 0.0	1.000 ± 0.0
linear programming	1.000 ± 0.0	1.000 ± 0.0

Table 4: Comparison of accuracies measured with different optimization methods on Image dataset

6.4 Estimating cardinality hyper parameters

Synthetic datasets and k-fold cross validation were used to estimate ratio of positive instance inside a bag. We did this by fixing all hyper parameters of our classifier, with the exception of ρ for RMIMN potential. We measured overall accuracy, which classifier achieved during k-fold cross validation for particular $\rho \backslash \kappa$ value and at the end we chose hyper parameter value for which classifier scored highest overall accuracy. We visualized accuracy of all optimization algorithms, which we used in our implementation for multiple ρ values on RMIMN version of synthetic datasets. The plots of accuracy measured on synthetic datasets for ρ values can be seen in Figure 3. Visualization of hyper parameters estimation for RMIMN and GMIMN potentials done on multiple instance learning datasets can be found in Appendix C.



(a) simple Noise dataset (b) complex Noise dataset (c) image dataset

Figure 3: Results of cross validation on multiple values of hyper parameter ρ measured on synthetic datasets with different optimization algorithms

6.5 Inference based SVM

With the use of liblinear's[18] implementation of linear SVM we successfully evaluated performance of mi-SVM-like optimization algorithm. Measured results of our implementation are slightly worse to similar methods and can be found in Table 5.

SVM Method	Musk1	Musk2	Tiger	Elephant	Fox
MIMN	0.826 ± 0.081	0.646 ± 0.131	0.744 ± 0.053	0.757 ± 0.051	0.564 ± 0.045
RMIMN	0.739 ± 0.075	0.738 ± 0.050	0.760 ± 0.040	0.824 ± 0.0357	0.564 ± 0.079
mi-SVM[6]	0.874	0.836	0.784	0.822	0.582
MI-SVM[6]	0.779	0.843	0.840	0.814	0.578

Table 5: Comparison of mi-SVM and inference based SVM on MIL datasets

6.6 Batch gradient descent

We implemented batch gradient descent method and evaluated its performance on multiple instance binary datasets and Camelyon dataset. We validated ability of batch gradient descent to train on weakly supervised data and to estimate value of cardinality parameters ρ and κ. We compare performance of batch gradient descent to state-of-art classification methods in Table 7.

6.7 Linear programming

We were able to implement linear programming method by using already implemented linear programming solver in lpsolvers library implemented with cvxopt[19] package. We ran cross-validation on common binary multiple instance learning datasets for maximum of 20 iterations per validation run. Results generated with linear programming were comparable to other state of art method, while accuracy achieved using GMIMN potential was worse than RMIMN across all datasets. We were not able to measure performance of linear programming approach on Camelyon dataset, due to inability of linear programming solver to find feasible solutions during training.

6.8 Camelyon results

We used predefined training set to estimate hyper parameters of our optimization methods with 5-fold cross validation. We then ran 5-fold cross validation on entire dataset and averaged results of those runs, which can be seen in Table 6. We had to set maximum number of iterations to 100 for both optimization methods, due to size of dataset, in which calculating 1 iteration for our optimization algorithms took around 1 hour. From results we can see, that MIMN

version of batch gradient descent outperformed all other optimization approaches. Resulting score achieved by batch gradient descent with MIMN cardinality potential is comparable to state-of-art methods.

Optimization method	Accuracy
inference based SVM MIMN	0.768 ± 0.102
inference based SVM RMIMN	0.645 ± 0.079
BGD MIMN	**0.899 ± 0.032**
BGD RMIMN	0.7143 ± 0.018

Table 6: Comparison of results measured on Camelyon datasets

6.9 Summary of results

Summary of results we measured on common binary multiple instance learning datasets and comparison to other state-of-art methods can be seen in Table 7.
We can see that the best results of batch gradient descent and linear programming are comparable to state-of-art methods, while RMIMN cardinality potential outperforms the MIMN and GMIMN cardinality potentials by a small margin, which might be due to low amount of values of κ we tested on GMIMN potential and smaller amount of cross-validation runs, which resulted in higher standard deviation of our tests.

Method	Fox	Tiger	Elephant	Musk1	Musk2
Inference based SVM MIMN	0.504 ± 0.047	0.74 ± 0.092	0.736 ± 0.065	0.692 ± 0.097	0.636 ± 0.131
Inference based SVM RMIMN	0.547 ± 0.054	0.708 ± 0.064	0.816 ± 0.038	0.660 ± 0.180	0.576 ± 0.081
BGD MIMN	0.580 ± 0.081	0.836 ± 0.029	0.828 ± 0.041	0.817 ± 0.047	0.7 ± 0.139
BGD RMIMN	0.576 ± 0.043	0.844 ± 0.03	0.856 ± 0.053	0.860 ± 0.036	0.707 ± 0.058
BGD GMIMN	0.548 ± 0.062	0.756 ± 0.043	0.712 ± 0.076	0.739 ± 0.03	0.739 ± 0.03
LP MIMN	0.576 ± 0.043	0.796 ± 0.035	0.840 ± 0.05	0.739 ± 0.092	0.692 ± 0.076
LP RMIMN	0.616 ± 0.079	0.764 ± 0.058	0.856 ± 0.053	0.834 ± 0.083	0.746 ± 0.132
LP GMIMN	0.548 ± 0.062	0.72 ± 0.05	0.736 ± 0.035	0.773 ± 0.089	0.669 ± 0.069
mi-SVM[6]	0.582	0.784	0.822	0.874	0.836
MI-SVM[6]	0.590	0.840	0.810	0.779	0.843
EM-DD[10]	0.561	0.721	0.783	0.848	0.849
AW-SVM[11]	0.640	0.830	0.820	0.860	0.840
AL-SVM[11]	0.630	0.780	0.790	0.860	0.830
ALP-SVM[11]	**0.660**	**0.860**	0.835	0.863	0.862
MICA[13]	0.587	0.826	0.805	0.844	**0.905**
MI-Kernel[14]	0.603	0.842	0.843	0.880	0.893
MI-Graph[15]	0.612	0.819	0.851	**0.900**	0.900
mi-Graph[15]	0.616	**0.860**	**0.868**	0.889	0.903

Table 7: Averaged accuracies of various state-of-art methods and comparison to performance achieved with our optimization algorithms

7 Conclusion

In this thesis we implemented framework of Markov networks introduced in [9]. We successfully validated optimization methods of batch gradient descent, inference based SVM and new method using linear programming on multiple instance learning datasets. Performance of our optimization algorithms was satisfactory, but still not comparable to state-of-art multiple instance learning methods. We measured accuracy of batch gradient descent and inference based SVM method on Camelyon dataset, in which MIMN definition of cardinality potential significantly outperformed non-standard multiple instance learning assumptions. This can be a consequence of extremely high number of instances inside a bag (ρ value 0.1 would mean that at least 569 instances on average have to be labeled as positive) , but further testing would be needed. From results on Camelyon dataset, we saw that performance of BGD using MIMN cardinality potential rivals 90% accuracy which is close to performance achieved by state-of-art optimization algorithms on histological images.

7.1 Future work

At the end of our thesis we propose few ideas for future work. First would be to find more robust linear programming solver and evaluate linear programming method on Camelyon dataset. Second would be to implement different kernels and non-convex optimization algorithms, so we can compare performance of our implementation to original paper. Last but not least would be to implement stochastic gradient descent method to reduce problem of lengthy iterations.

Appendices

A Python framework

In this section we provide tables of possible arguments with descriptions and general guide on how to set-up and run our framework.

A.1 Setting up

We provide link for our **github**[1] account, where anyone can download framework described in [9], which we reproduced in python3. We also provide quick set up guide on our github page.

A.2 Running framework

After following installations steps described on github page, we can run our program with command:

```
$> python3 main.py <split> <kernel> <dataset> <potential>
```

[1] https://github.com/branislav-doubek/multiple-instance-learning

A.3 Parameteres and keywords

We present tables of parameters and keywords used in our implementation of [9]'s framework in python3

Split type	**Description**
train	Trains the classifier
test	Loads trained classifier and evaluates its performance
cv	Runs cross-validation mode
run	Trains the classifier and tests its performance
validate	Runs 1 fold of cross-validation
Kernel type	**Description**
bgd	Batch gradient descent
svm	Mi-SVM defined in [6]
lp	Linear programming
Dataset	**Description**
noise	gaussian noise dataset
image	noise image dataset
fox	
tiger	
elephant	
musk1	
musk2	
Cardinality potential	**Description**
MIMN	defined in Equation 4
RMIMN	defined in Equation 5
GMIMN	defined in Equation 6
Parameters	**Description**
-ro	used in rmimn potential as a constraint
-c	sets parameter C
-iterations	sets maximum number of permitted iterations
-k	used in gmimn potential as a constraint
-rs	random seed
-v	visualize
-cv	cross-validate on hyperparameter
-lr	learning rate
`-valid_iter`	Selects fold used as a testing dataset in validation
-norm	regularization norm [1, 2] (Only for bgd)

Table 8: List of framework keywords and parameters

A.4 Cross-validation mode

If we use 'cv' split type, we run k-fold cross validation across array of possible values of tunable hyper parameters on training dataset. We publish list of tunable parameters in table below.

Tunable parameters	**List of values**	**Description**
η	{1e-5, 1e-4, 1e-3}	learning rate
α	{0.3, 0.5, 0.7}	momentum parameter
C	{0.001, 0.01, 0.1, 1, 10, 1000}	regularisation parameter
κ	{3,5,7,10}	parameter used in GMIMN potential
ρ	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}	parameter used in RMIMN potential

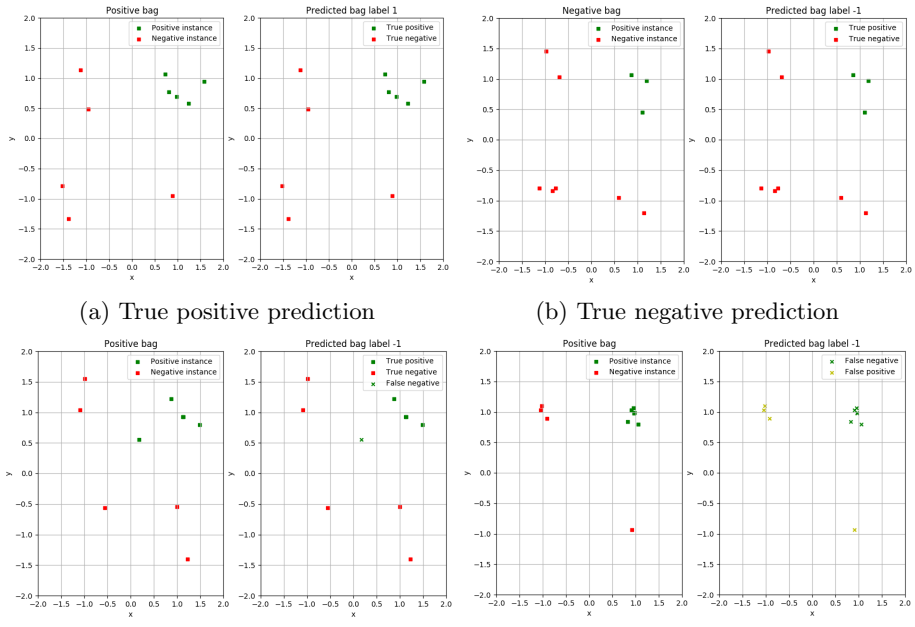
Table 9: List of cross-validation parameters

B Inference algorithm visualization

This appendix section is used for visualizing results of inference algorithm trained with batch gradient descent, inference based SVM and linear programming optimization algorithms on complex version of Noise dataset and RMIMN and MIMN definition of image dataset.

B.1 Noise dataset

We visualized few exemplary predictions in Figure 4 made on complex Noise dataset along with minimization of loss function in Figure 5 and Figure 6 during training to compare performances and valiity of inference based SVM, batch gradient descent and linear programming.



(a) True positive prediction

(b) True negative prediction

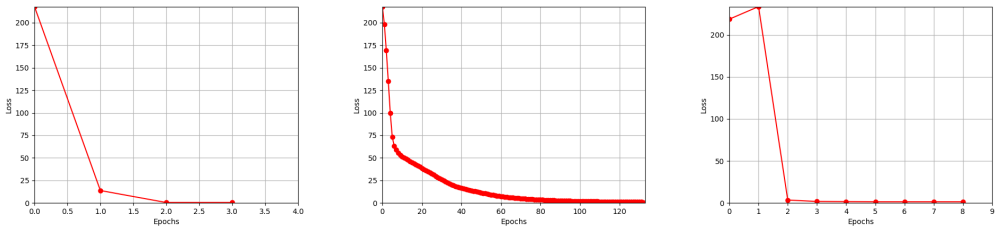
(c) False negative prediction

(d) False positive prediction

Figure 4: Visualization of predicted bag labels using inference algorithm on Noise dataset

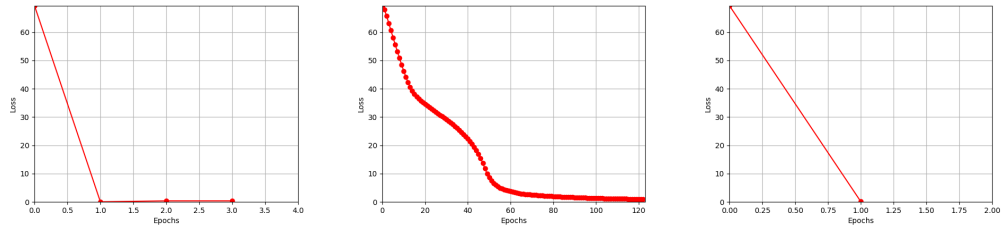
B.2 Image dataset

We visualized few examples of classification done on RMIMN and MIMN versions of synthetic datasets, which should provide some context on how the inference algorithm assigns instance labels to instance feature vectors. Resulting plots provide a proof, that we were able to



(a) inference based SVM (b) BGD (c) linear programming

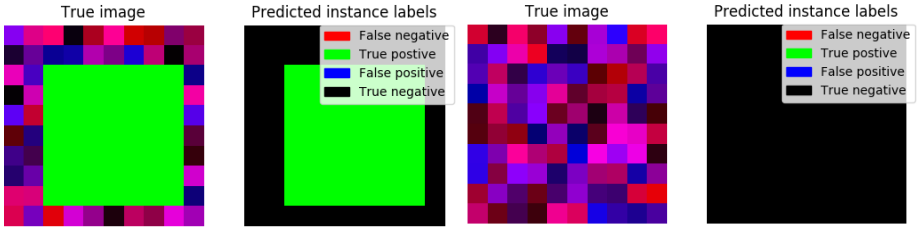
Figure 5: Visualization of minimization of loss function J for different optimization algorithms on MIMN version of complex Noise dataset



(a) inference based SVM (b) BGD (c) linear programming

Figure 6: Visualization of minimization of loss function J for different optimization algorithms on RMIMN version of complex Noise dataset

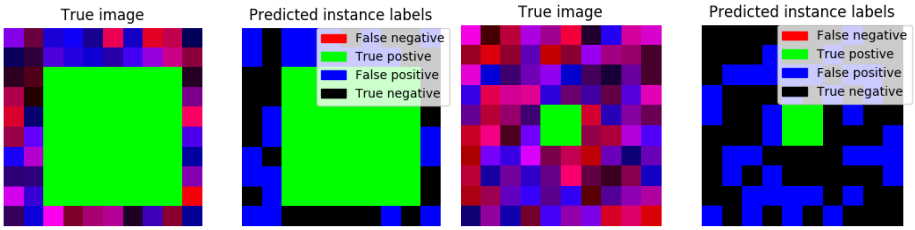
successfully find optimal solution for MIMN and RMIMN versions of Image dataset.



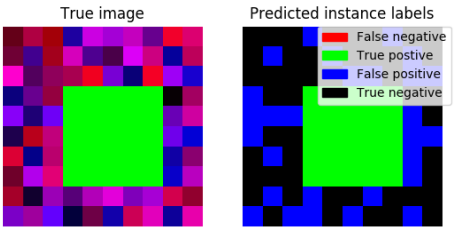
(a) True positive prediction (b) True negative prediction

Figure 7: Visualization of predicted bag labels using inference algorithm on MIMN definition of Image dataset

From Figure 8 example (c) we can see, that inference algorithm was able to find all positive instances inside a bag, but due to ratio of positive instances inside negative bag, which is close to ρ denoting classifier's RMIMN cardinality potential threshold. Classifier misclassifies bag as a result.
In Figure 9 and Figure 10 we visualized minimization of loss function over training epochs.

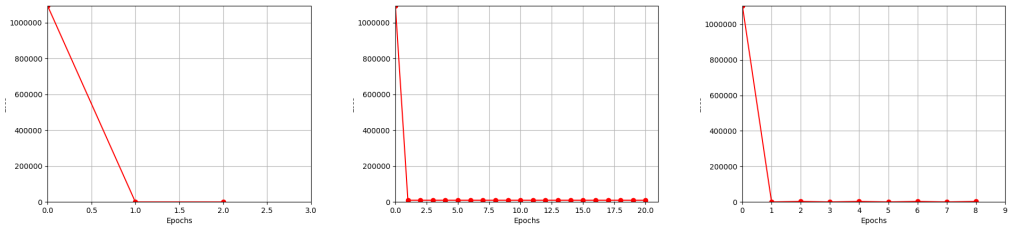


(a) True positive prediction (b) True negative prediction



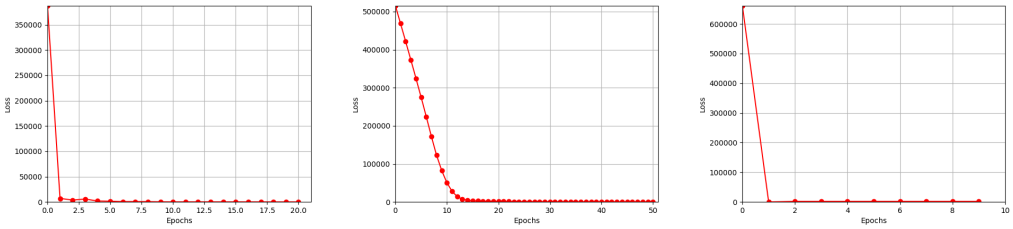
(c) False positive prediction

Figure 8: Visualization of predicted bag labels using inference algorithm on RMIMN definition of Image dataset



(a) inference based SVM (b) BGD (c) linear programming

Figure 9: Visualization of minimization of loss function J for different optimization algorithms on MIMN version of Image dataset



(a) inference based SVM (b) BGD (c) linear programming

Figure 10: Visualization of minimization of loss function J for different optimization algorithms on RMIMN version of Image dataset

C Results of cross validation

In this section we present results of k-fold cross validation on multiple instance learning datasets with various optimization algorithms. We visualize dependency on cardinality parameters ρ and κ for RMIMN and GMIMN potentials and present parameters, which we used in final evaluation along with averaged accuracy we measured on 5 differently seeded runs.

C.1 Tiger dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-3	0.1	x	x	0.836 ± 0.029
SVM	x	MIMN	x	1	x	x	0.74 ± 0.092
LP	x	MIMN	x	10	x	x	0.796 ± 0.035
BGD	0.7	RMIMN	1e-3	0.1	0.1	x	**0.844 ± 0.03**
SVM	x	RMIMN	x	1	0.8	x	0.708 ± 0.064
LP	x	RMIMN	x	10	0.4	x	0.764 ± 0.058
BGD	0.7	GMIMN	1e-3	0.1	x	5	0.756 ± 0.043
LP	x	GMIMN	x	10	x	5	0.72 ± 0.05

Table 10: Results of cross-validation on Tiger dataset

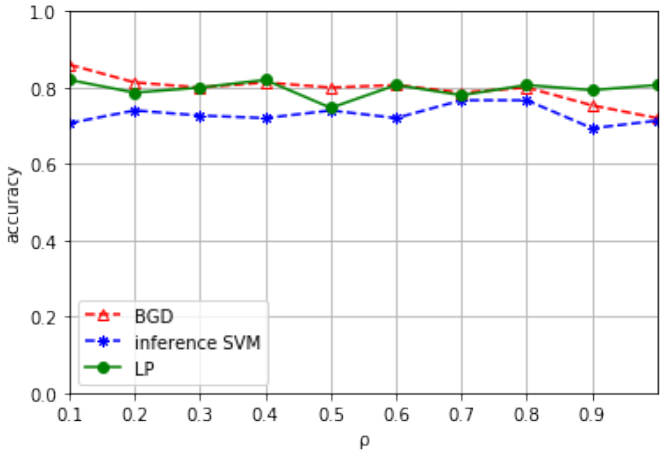


Figure 11: Comparison of accuracy based on RMIMN potential parameters on Tiger dataset

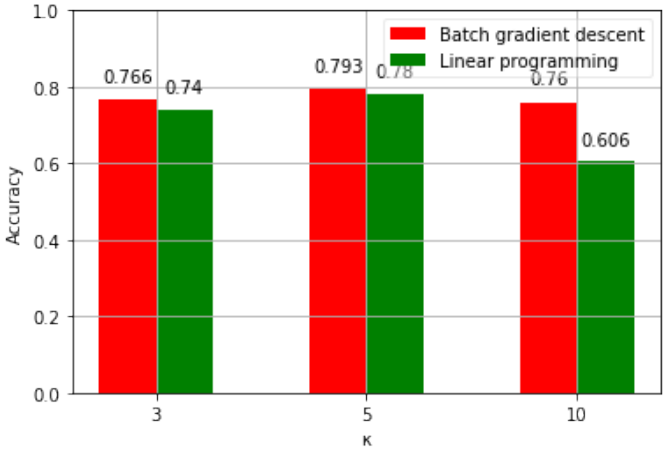


Figure 12: Comparison of accuracy based on GMIMN potential parameters on Tiger dataset

C.2 Fox dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-3	1	x	x	0.580 ± 0.081
SVM	x	MIMN	x	0.01	x	x	0.504 ± 0.047
LP	x	MIMN	x	1000	x	x	0.576 ± 0.049
BGD	0.7	RMIMN	1e-3	1	1.0	x	0.576 ± 0.043
SVM	x	RMIMN	x	0.01	1.0	x	0.547 ± 0.054
LP	x	RMIMN	x	1000	0.8	x	**0.616 ± 0.079**
BGD	0.7	GMIMN	1e-3	1	x	10	0.548 ± 0.062
LP	x	GMIMN	x	1000	x	3	0.548 ± 0.062

Table 11: Results of cross-validation on Fox dataset

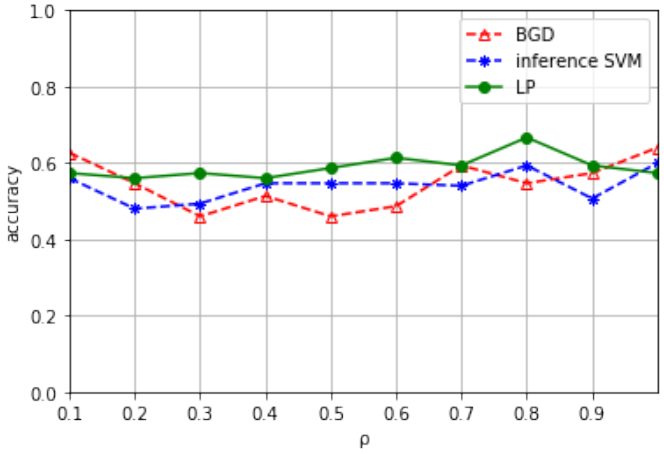


Figure 13: Comparison of accuracy based on RMIMN potential parameters on Fox dataset

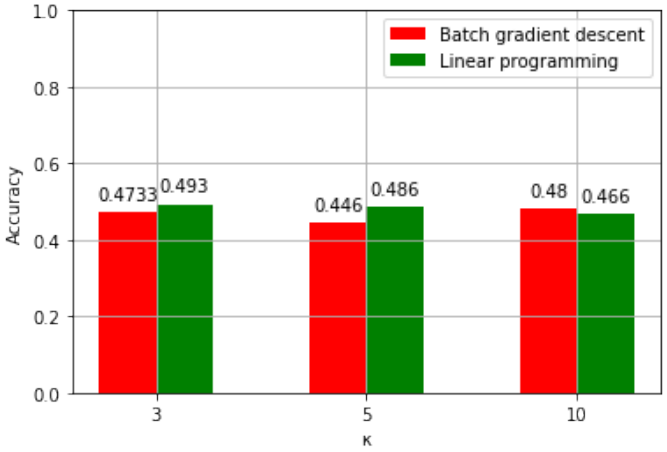


Figure 14: Comparison of accuracy based on GMIMN potential parameters on Fox dataset

C.3 Elephant dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-3	0.1	x	x	0.828 ± 0.041
SVM	x	MIMN	x	1	x	x	0.736 ± 0.065
LP	x	MIMN	x	1	x	x	0.840 ± 0.050
BGD	0.7	RMIMN	1e-3	0.1	1.0	x	**0.856 ± 0.053**
SVM	x	RMIMN	x	1	0.9	x	0.816 ± 0.038
LP	x	RMIMN	x	1	1.0	x	**0.856 ± 0.053**
BGD	0.7	GMIMN	1e-4	0.1	x	3	0.712 ± 0.076
LP	x	GMIMN	x	1	x	3	0.736 ± 0.035

Table 12: Results of cross-validation on Elephant dataset

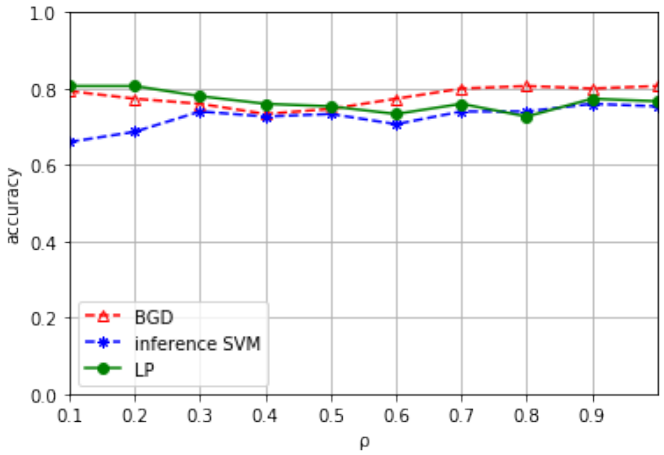


Figure 15: Comparison of accuracy based on RMIMN potential parameters on Elephant dataset

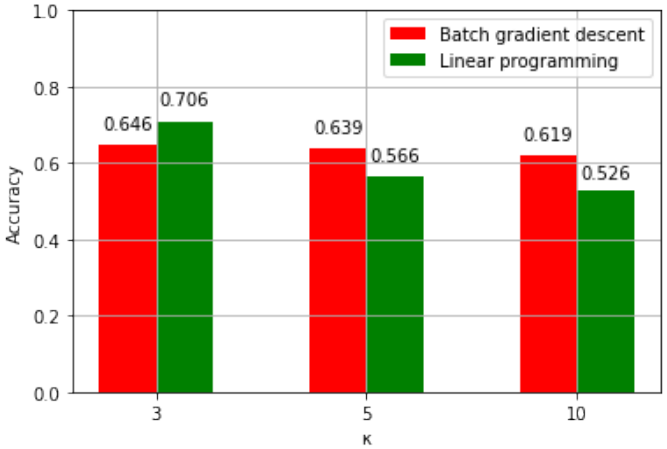


Figure 16: Comparison of accuracy based on GMIMN potential parameters on Elephant dataset

C.4 Musk1 dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-3	10	x	x	0.817 ± 0.047
SVM	x	MIMN	x	0.1	x	x	0.692 ± 0.097
LP	x	MIMN	x	10	x	x	0.739 ± 0.092
BGD	0.7	RMIMN	1e-3	10	0.2	x	**0.860 ± 0.036**
SVM	x	RMIMN	x	0.1	0.6	x	0.660 ± 0.180
LP	x	RMIMN	x	10	0.7	x	0.834 ± 0.083
BGD	0.7	GMIMN	1e-3	10	x	3	0.739 ± 0.03
LP	x	GMIMN	x	10	x	3	0.773 ± 0.089

Table 13: Results of cross-validation on Musk1 dataset

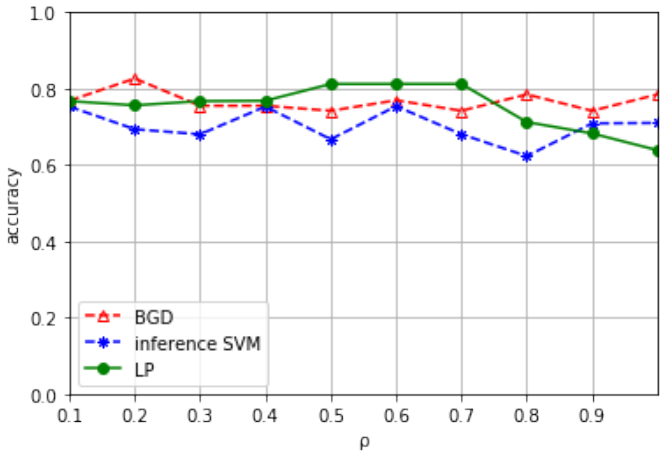


Figure 17: Comparison of accuracy based on RMIMN potential parameters on Musk1 dataset

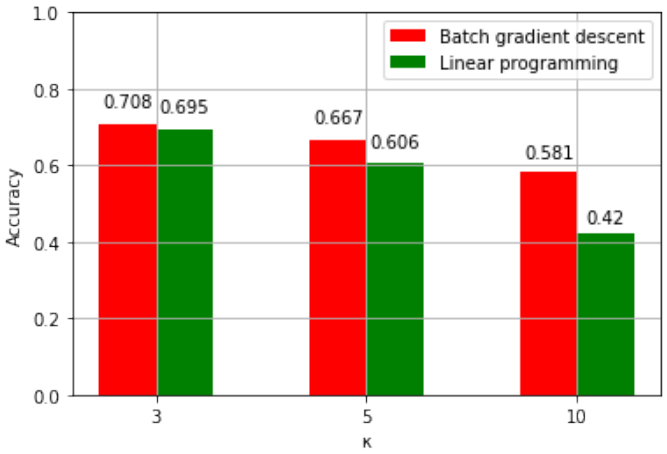


Figure 18: Comparison of accuracy based on GMIMN potential parameters on Musk1 dataset

C.5 Musk2 dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-4	0.01	x	x	0.7 $\pm$ 0.139
SVM	x	MIMN	x	0.01	x	x	0.636 $\pm$ 0.131
LP	x	MIMN	x	1	x	x	0.692 $\pm$ 0.076
BGD	0.7	RMIMN	1e-4	0.01	1.0	x	0.707 $\pm$ 0.058
SVM	x	RMIMN	x	0.01	0.4	x	0.576 $\pm$ 0.081
LP	x	RMIMN	x	1	0.4	x	**0.746** $\pm$ 0.132
BGD	0.7	GMIMN	1e-4	0.01	x	3	0.630 $\pm$ 0.096
LP	x	GMIMN	x	1	x	5	0.669 $\pm$ 0.069

Table 14: Results of cross-validation on Musk2 dataset

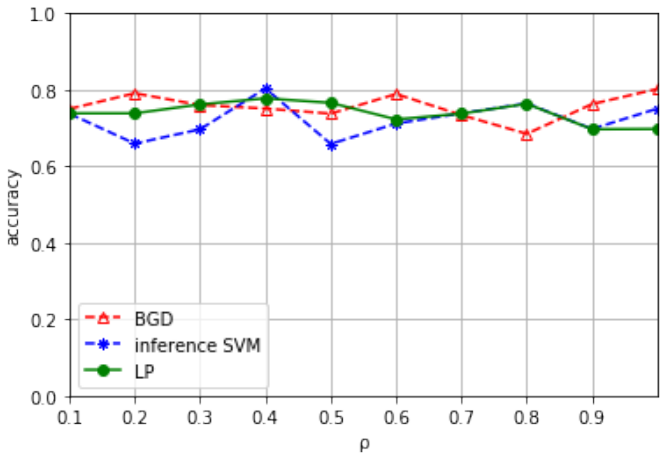


Figure 19: Comparison of accuracy based on RMIMN potential parameters on Musk2 dataset

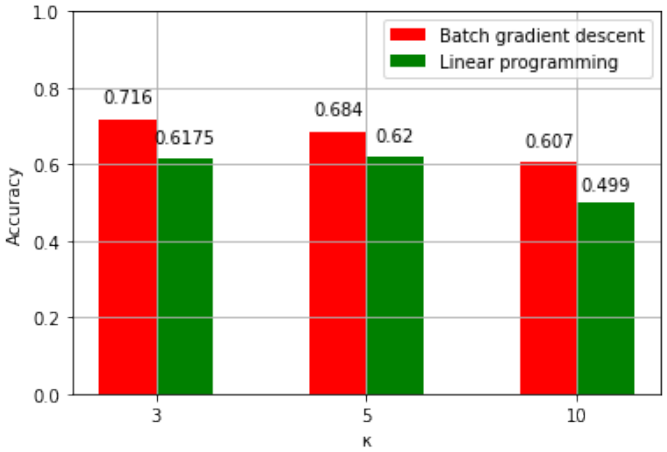


Figure 20: Comparison of accuracy based on GMIMN potential parameters on Musk2 dataset

C.6 Camelyon dataset

Method	α	Cardinality potential	η	C	ρ	κ	Accuracy[%]
BGD	0.7	MIMN	1e-6	10	x	x	**0.899 ± 0.032**
SVM	x	MIMN	x	1000	x	x	0.768 ± 0.102
BGD	0.7	RMIMN	1e-6	10	0.1	x	0.7143 ± 0.018
SVM	x	RMIMN	x	1000	x	0.1	0.0.645 ± 0.079

Table 15: Results of cross-validation on Camelyon dataset

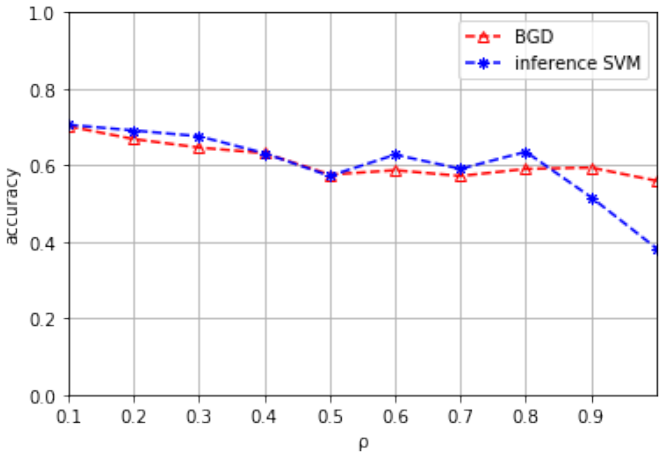


Figure 21: Comparison of accuracy based on RMIMN potential parameters on Camelyon dataset

D List of attachments

1x CD containing implementation of multiple instance learning framework

References

[1] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89(1-2):31–71, January 1997.

[2] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *Advances in neural information processing systems*, pages 570–576, 1998.

[3] D. Liu, Y. Zhou, X. Sun, Z. Zha, and W. Zeng. Adaptive pooling in multi-instance learning for web video annotation. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 318–327, Oct 2017.

[4] Changbo Yang, Ming Dong, and Farshad Fotouhi. Region based image annotation through multiple-instance learning. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 435–438, New York, NY, USA, 2005. ACM.

[5] Christian Leistner, Amir Saffari, and Horst Bischof. Miforests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision*, pages 29–42. Springer, 2010.

[6] Stuart Andrews, Thomas Hofmann, and Ioannis Tsochantaridis. Multiple instance learning with generalized support vector machines. *Proceedings of the National Conference on Artificial Intelligence*, 06 2003.

[7] Trinh-Minh-Tri Do and Thierry Artières. Large margin training for hidden markov models with partially observed states. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 265–272, New York, NY, USA, 2009. ACM.

[8] Geert Litjens, Peter Bandi, Babak Ehteshami Bejnordi, Oscar Geessink, Maschenka Balkenhol, Peter Bult, Altuna Halilovic, Meyke Hermsen, Rob van de Loo, Rob Vogels, Quirine F Manson, Nikolas Stathonikos, Alexi Baidoshvili, Paul van Diest, Carla

Wauters, Marcory van Dijk, and Jeroen van der Laak. 1399 H amp;E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset. *GigaScience*, 7(6), 05 2018. giy065.

[9] H. Hajimirsadeghi a G. Mori. Multi-instance classification by max-margin training of cardinality-based markov networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1839–1852, 2017.

[10] Qi Zhang and Sally A Goldman. Em-dd: An improved multiple-instance learning technique. 2002.

[11] PV. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. In *JMLR Workshop and Conference Proceedings Volume 2: AISTATS 2007*, pages 123–130, Cambridge, MA, USA, March 2007. Max-Planck-Gesellschaft, MIT Press.

[12] Daniel Tarlow, Kevin Swersky, Richard S. Zemel, Ryan Prescott Adams, and Brendan J. Frey. Fast exact inference for recursive cardinality models. *CoRR*, abs/1210.4899, 2012.

[13] Olvi L Mangasarian and Edward W Wild. Multiple instance classification via successive linear programming. *Journal of Optimization Theory and Applications*, 137(3):555–568, 2008.

[14] Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alexander J Smola. Multi-instance kernels. In *ICML*, volume 2, page 7, 2002.

[15] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *Proceedings of the 26th annual international conference on machine learning*, pages 1249–1256. ACM, 2009.

[16] Rahul Gupta and Sunita Diwan, Ajit A. a Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th International Conference on Machine Learning*, pages 329–336, New York, NY, USA, 2007. ACM.

[17] Veronika Cheplygina and David M. J. Tax. Characterizing multiple instance datasets. *CoRR*, abs/1806.08186, 2018.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Lieven Vandenberghe Martin Andersen, Joachim Dahl. Cvxopt, 2004.