A Review on Multi-Label Learning Algorithms

Min-Ling Zhang and Zhi-Hua Zhou

IEEE Transactions On Knowledge And Data Engineering

January 21, 2018

Overview

Introduction

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y_i})\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f: X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance. $\{(x_i, \boldsymbol{y_i})\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \dots N, x \in X, y \in Y = \{y_1, \dots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance.
  $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \dots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Multi-label learning

Learning $\rightarrow$ build a model from data, to accomplish a task

- Supervised: we have both data *and labels*
- Applied to classification in this study

Supervised classification:

- Given input data X and labels Y, learn the function $f : X \rightarrow Y$
- $f(x_i, y_i) = r$ where $r \in \mathbb{R}$ is the confidence that y_i characterizes x_i
- Assume that x_i belongs to y_i if $r \geq t(x_i)$
  - ✓ $t(\cdot)$ can be a predetermined constant function or learned from X

Single-label learning

- Dataset $\{(x_i, y_i)\}, i = 1, \ldots N, x \in X, y \in Y = \{y_1, \ldots, y_q\}$

Multi-label learning

- Multi-label dataset: Multiple labels per instance.
  $\{(x_i, \boldsymbol{y}_i)\}, i = 1, \ldots N, x \in X, \boldsymbol{y} \in \mathbb{P}(Y)$

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- e.g. for $q = 20$, $|\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20$, $|\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20, |\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
    - Ignore label correlations
    - Often transform M-L problem to multiple, single-label problems and combine the per-label results
    - Simple, scalable, suboptimal
2. Second-order strategies
    - Consider *pairwise* label relations
    - Good trade-off between generalization performance and scalability
    - Lacking in some real-world applications
3. Higher-order strategies
    - Capture more complicated label interdependencies
    - Strong modeling capabilities that can capture complex relationships
    - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20$, $|\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^{6}$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

▷ e.g. for $q = 20$, $|\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Algorithm Strategies

Label search space $\mathbb{S}_{\mathbb{Y}}$ grows exponentially as a function of $|Y| = q$

- ▷ e.g. for $q = 20$, $|\mathbb{S}_{\mathbb{Y}}| = 2^{|\mathbb{P}(\mathbb{Y})|} = 2^{20} \geq 10^6$

Solution: integrate in the learning process potential label correlations.

In this work the authors group M-L algorithms in three categories:

1. First-order strategies
   - Ignore label correlations
   - Often transform M-L problem to multiple, single-label problems and combine the per-label results
   - Simple, scalable, suboptimal
2. Second-order strategies
   - Consider *pairwise* label relations
   - Good trade-off between generalization performance and scalability
   - Lacking in some real-world applications
3. Higher-order strategies
   - Capture more complicated label interdependencies
   - Strong modeling capabilities that can capture complex relationships
   - Computationally demanding, less scalable

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^1 measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β}-measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

$*$ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.

Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

$*$ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^β -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.

Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.

Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Introduction: Evaluation Metrics

Extension of single-label metrics to the M-L case.
Grouped into two categories and perspectives:

- *Example-based:* Evaluates multi-labeled performance on each example, extrapolate to whole dataset
  - *Classification perspective:*
    - Subset Accuracy, Hamming Loss
    - Precision, Recall, F^{β} -measure
  - *Ranking perspective:* one-error, coverage, ranking loss, average precision
- *Label-based:* Evaluates performance on each label separately, extrapolate to whole label set
  - Classification perspective: Macro/Micro averaging techniques for single-label example-based, classification-perspective measures
  - Ranking perspective: Macro/Micro averaging for AUC

∗ Ideally, classifiers should be trained to optimize *multiple* metrics

Multi-label Learning Algorithms

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - “Fit data to algorithm” philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - “Fit algorithm to data” philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

- ✓ Has broad, noteworthy or unique characteristics
- ✓ Has important impact, leading to a number follow-up related methods
- ✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - “Fit data to algorithm” philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - “Fit algorithm to data” philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
  - Transform the learning problem into other, managable (often single-label) learning problems
  - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
  - Adapt popular learning techniques to deal with multi-label data directly
  - "Fit algorithm to data" philosophy

Authors include algorithms that:

✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Multi-label Learning Algorithms: Categorization

Group algorithms in two categories:

- *Problem Transformation Methods:*
    - Transform the learning problem into other, managable (often single-label) learning problems
    - "Fit data to algorithm" philosophy
- *Algorithm Adaptation Methods:*
    - Adapt popular learning techniques to deal with multi-label data directly
    - "Fit algorithm to data" philosophy

Authors include algorithms that:

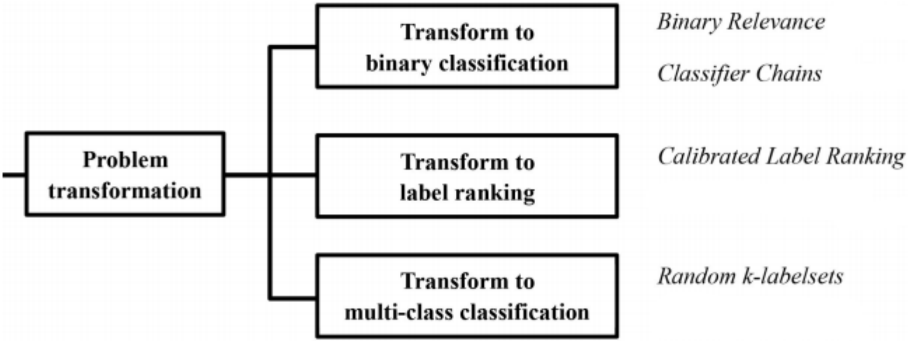
✓ Has broad, noteworthy or unique characteristics

✓ Has important impact, leading to a number follow-up related methods

✓ Is influential and highly-cited in multi-label learning

Problem Transformation Methods

Multi-label Learning Algorithms: Problem Transformation Methods



Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Binary Relevance

- Decompose multi-label problem to $|Y| = q$ independent binary classification problems
- Construct q binary (one-vs-all) training sets (one for each y_i)
- Independently train each classifier $h_i(x)$ on its respective dataset
- Predict labels of an unseen x by evaluating each classifier $h_i(x)$
- Assign label y_i according to $r = h_i(x)$ and the thresholding setting

Pros & cons:

- Simple, one-vs-rest scheme $\rightarrow$ easily parallelizable
- Sensitive to class-imbalanced data[1]
- Ignores label correlations (first-order method)

[1] Very different number of pos. and neg. examples for a class

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Classifier chains

- Reorder the label Y set using a permutation function f_p
- Transform into a *chain* of binary classification problems
- Enrich representations at step j by concatenating each x_i with the confidence of preceeding $1, \ldots, j-1$-th classifiers
- Predict relevant labels for unknown instances by iteratively traversing the classifier chain

Pros & cons:

- High-order method: exploitation of label correlations to a degree, but in a random manner
- Highly sensitive to f_p. Running multiple chain executions in an ensemble attempt to overcome this dependency.
- Iterative operation prevents parallel implementation

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
    - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
    - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Calibrated Label Ranking

- Transform into a label pairwise comparison ranking problem
- For q labels, generate $q(q-1)/2$ binary classifiers by pairwise comparison and use a binary algorithm $h_{jk}(x)$
- Construct training sets $D_{jk} : \{x_i, Y_i | y_j \in Y_i \oplus y_k \in Y_i\}$. System votes for each example and if $h_{jk}(x) > 0$,x_i is associated with y_j, otherwise with y_k
- For an unknown instance, all classifiers' votes are aggregated and resulting labels are ranked according to the total confidence r
- A virtual label y_v is learned as a threshold, serving as the artificial splitting point between relevant and irrelevant labels

Pros & cons:

- Second-order approach algorithm. One-vs-one scheme.
- Pairwise comparison smooths out the class-imbalance problem
- Number of classifiers is quadratic to $|Y|$ (compared to linear for BR)
  - Pruning methods to reduce the search space

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Random k-Labelsets

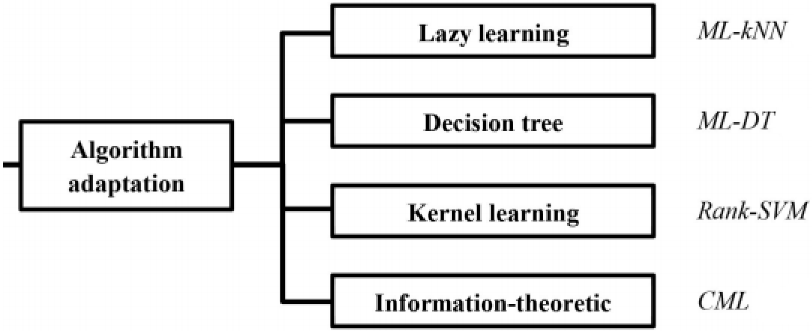
- Decompose to an ensemble of multi-class classification problems
- Each component targets a random subset of $\mathbb{P}(Y)$ (that also appears in X), classified with Label Powerset (LP) techniques:
  - Transform to single-label data by treating each distinct labelset as a new class $\rightarrow$ multi to single label problem
  - Each example is reassigned with the new mapped class and classified through regular single-label classification
  - M-L classify x with y_i when the votes received for y_i from the ensemble exceed half the max possible it can get

Pros & cons:

- High-order approach algorithm
- Data-sensitive: Cannot generalize to labelsets not in the training set, too few examples for some labelsets
- Large Y implies high training complexity.
- Improve by invoking an ensemble on random k-sized labelsets

Algorithm Adaptation Methods

Multi-label Learning Algorithms: Alogrithm Adaptation Methods



Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\widetilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)} [\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\widetilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\tilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)} [\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\widetilde{\kappa_j}(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-Label k-Nearest Neighbour

- Consider events $H_j \equiv (y_j \in Y_i)$, $C_j \equiv (\sum_{x_k \in N(x_i)}[\delta(y_j \in Y_k)])$
- Compares the MAP probabilities: $P(K|C_j), K \in \{H_j, \neg H_j\}$ to decide if to include y_j in the prediction. Compute with the Bayes' theorem.
- Single-label priors $P(K)$ computed by smoothed frequency counting in the training data
- Likelihoods $P(C_j|K)$ are computed as a function of:
  - $\kappa_j(r)$, the number of examples labelled y_j with r neighbours labelled y_j
  - $\widetilde{\kappa}_j(r)$, the number of examples not labelled y_j with r neighbours labelled y_j

Pros & cons:

- First-order approach, Bayesian reasoning
- Decision boundary can be modified on-line via new instances
- M-L class imbalance can be overcome by calculating the priors
- Extensions proposed for label correlation exploitations

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \to \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \rightarrow \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Multi-label Decision Tree

- Multi-label entropy is used to build a decision tree recursively
- Split node at the l-th feature of x such that the information gain criterion is maximized. Node partitions data according to $x_l = \theta$: $T \to \{T^-, T^+\}$, where $x_{il} \leq \theta, x_i \in T^-$ and $x_{il} > \theta, x_i \in T^+$
- Recurse on subtrees until a stopping criterion is met (e.g. child size)
- Single-label entropy is computed by considering labelsets as new classes
- Assumes independence among labels to ensure low computational cost
- Unseen instances are assigned the label of the majority of members of the leaf they arrive

Pros & cons:

- First-order algorithm, highly efficient
- Assumes label independence
- Pruning and ensemble strategies proposed

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j,k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j,k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Ranking Support Vector Machine

- q linear classifiers $h_i(x)$, optimized with the empirical ranking loss
- Combine classifiers to discriminate label *pairs*. Label pair (j, k) decision hyperplane is: $h_{jk} = g(h_j, h_k) = \langle w_j - w_k, x_i \rangle + b_j - b_k = 0$
- Consider signed distance of x_i from the boundary of every relevant-nonrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$.
- The M-L margin is the minimum distance of x_i from every $Y_i \times \bar{Y}_i$
- SVM $\rightarrow$ minimize loss *and* maximize margin.
- Ranks labels according to the singed distance
- Non-linearity achievable through feature mapping and the kernel trick

Pros & cons:

- Second-order approach, maximum margin strategy
- Optimization is a convex QP problem, solved by any QP solver
- Kernel SVM, kernel selection can be done with MKL techniques
- Adaptable learning by selecting an appropriate loss function

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, y)] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, y)] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem → solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, y)] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
    - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
    - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
    - Solve with Lagrange multipliers, assuming gaussian priors
    - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
    - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
    - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
    - Solve with Lagrange multipliers, assuming gaussian priors
    - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\, p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
    - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
    - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
    - Solve with Lagrange multipliers, assuming gaussian priors
    - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\, p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y \; p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Collective Multi-Label Classifier

- A Conditional Random Field model for M-L classification
- Encodes y as binary random vectors $\boldsymbol{y} = \{-1, 1\}^q$. Learn joint probability distribution $p(x, \boldsymbol{y})$
- Compute conditional $p(\boldsymbol{y}|x)$ with the maximum entropy criterion:
  - Maximize $H_p(x, y)$ subject to K constraints $\mathbb{E}[f_k(x, \boldsymbol{y})] = F_k$
  - $f_k(\cdot)$ directly model label correlations - the F_k values are learned from the training data
  - Solve with Lagrange multipliers, assuming gaussian priors
  - Optimal solution is in the form of a Gibbs probability distribution
- Unseen instances labelled with $Y = \text{argmax}_y\ p(y|x)$

Pros & cons:

- Second-order approach. All label pairs considered, not just relevant-irrelevant ones.
- Convex constraint optimization problem $\rightarrow$ solvable by any CP solver
- Intractable argmax operation for a large label space without pruning.
- Extensible: Different forms of constraints yield various CML variants

Multi-label Learning Algorithms: Summary

Algorithm	Basic Idea	Order of Correlations	Complexity [Train/Test]	Tested Domains	Optimized Metric
Binary Relevance [5]	Fit multi-label data to q binary classifiers	first-order	$\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d))/$ $\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d))$	image	classification (hamming loss)
Classifier Chains [72]	Fit multi-label data to a chain of binary classifiers	high-order	$\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d+q))/$ $\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d+q))$	image, video text, biology	classification (hamming loss)
Calibrated Label Ranking [30]	Fit multi-label data to $\frac{q(q+1)}{2}$ binary classifiers	second-order	$\mathcal{O}(q^2 \cdot \mathcal{F}_{\mathcal{B}}(m, d))/$ $\mathcal{O}(q^2 \cdot \mathcal{F}'_{\mathcal{B}}(d))$	image, text biology	Ranking (ranking loss)
Random k-Labelsets [94]	Fit multi-label data to n multi-class classifiers	high-order	$\mathcal{O}(n \cdot \mathcal{F}_{\mathcal{M}}(m, d, 2^k))/$ $\mathcal{O}(n \cdot \mathcal{F}'_{\mathcal{M}}(d, 2^k))$	image, text biology	classification (subset accuracy)
ML-kNN [108]	Fit k-nearest neighbor to multi-label data	first-order	$\mathcal{O}(m^2 d + qmk)/$ $\mathcal{O}(md + qk)$	image, text biology	classification (hamming loss)
ML-DT [16]	Fit decision tree to multi-label data	first-order	$\mathcal{O}(mdq)/\mathcal{O}(mq)$	biology	classification (hamming loss)
Rank-SVM [27]	Fit kernel learning to multi-label data	second-order	$\mathcal{O}(\mathcal{F}_{\mathrm{QP}}(dq + mq^2, mq^2)$ $+q^2(q+m))/\mathcal{O}(dq)$	biology	Ranking (ranking loss)
CML [33]	Fit conditional random field to multi-label data	second-order	$\mathcal{O}(\mathcal{F}_{\mathrm{UNC}}(dq + q^2, m))/$ $\mathcal{O}((dq + q^2) \cdot 2^q)$	text	classification (subset accuracy)

Figure: Summary of Representative Multi-Label Learning Algorithms

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
    - Instead of labeled instances, get binary-labeled *bags of instances*
    - Assign positive label to bag if at least one member is positive
    - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
    - Assume label relevance is not binary, but soft
    - Produce a vector of ordinal graded membership
    - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
    - Multiple tasks trained in parallel, sharing information
    - Knowledge from related tasks used as an inductive bias to improve generalization
    - Shared or different feature space, small task workload
- Data streams classification
    - Real-world objects are generated online and processed in real-time
    - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Related Learning Methods

- Multi-instance learning
  - Instead of labeled instances, get binary-labeled *bags of instances*
  - Assign positive label to bag if at least one member is positive
  - Models complex semantics of x_i in input space, rather than its output
- Ordinal classification
  - Assume label relevance is not binary, but soft
  - Produce a vector of ordinal graded membership
  - Transform M-L problem to a set of ordinal set of problems
- Multi-task learning
  - Multiple tasks trained in parallel, sharing information
  - Knowledge from related tasks used as an inductive bias to improve generalization
  - Shared or different feature space, small task workload
- Data streams classification
  - Real-world objects are generated online and processed in real-time
  - Concept drift problem

Conclusion

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion

Summary:

- Multi-label learning problem definition
- Multi-label learning representative algorithms
- Related learning methods

Future goals:

- Formal characterization on the underlying concept / mechanism on the appropriate usage of label correlations, especially on large output spaces
- Thorough experimental comparative study to discover pros and cons of different multi-label learning algorithms

Conclusion: Online resources

Resource Type	Resource URL and Descriptions
Tutorial	*http://www.ecmlpkdd2009.net/program/tutorials/learning-from-multi-label-data/* (In conjunction with ECML PKDD 2009)
	http://cig.fi.upm.es/index.php/presentations?download=4 (In conjunction with TAMIDA 2010)
Workshops	*http://lpis.csd.auth.gr/workshops/mld09/* (**MLD'09**: in conjunction with ECML PKDD 2009)
	http://cse.seu.edu.cn/conf/MLD10/ (**MLD'10**: in conjunction with ICML/COLT 2010)
	http://cse.seu.edu.cn/conf/LAWS12/ (**LAWS'12**: in conjunction with ACML 2012)
Special Issue	*http://mlkd.csd.auth.gr/events/ml2010si.html* (**Machine Learning Journal** Special Issue on Learning from Multi-Label Data [96])
Software	*http://mulan.sourceforge.net/index.html* (The MULAN [93] open-source Java library)
	http://meka.sourceforge.net/ (The MEKA project based on WEKA [38])
	http://cse.seu.edu.cn/people/zhangml/Resources.htm#codes_mll (Matlab codes for multi-label learning)
Data Sets	*http://mulan.sourceforge.net/datasets.html* (Data sets from MULAN)
	http://meka.sourceforge.net/#datasets (Data sets from MEKA)
	http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html (Data sets from LIBSVM [11])

Figure: Online Resources for Multi-Label Learning

Conclusion: Related Work

- Madjarov, Gjorgji, et al. "An extensive experimental comparison of methods for multi-label learning." Pattern Recognition 45.9 (2012): 3084-3104
- Tsoumakas, Grigorios, and Ioannis Katakis. "Multi-label classification: An overview." International Journal of Data Warehousing and Mining 3.3 (2006)
- Zhang, Min-Ling, and Kun Zhang. "Multi-label learning by exploiting label dependency." Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010

Conclusion: Additional Bibliography

- Vens, Celine, et al. "Decision trees for hierarchical multi-label classification." Machine Learning 73.2 (2008): 185-214.
- Tsoumakas, Grigorios, and Ioannis Vlahavas. "Random k-labelsets: An ensemble method for multilabel classification." Machine learning: ECML 2007 (2007): 406-417.
- Zhang, Min-Ling, and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." Pattern recognition 40.7 (2007): 2038-2048.
- Elisseeff, Andr, and Jason Weston. "A kernel method for multi-labelled classification." Advances in neural information processing systems. 2002.
- Ghamrawi, Nadia, and Andrew McCallum. "Collective multi-label classification." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.
- Cheng, Weiwei, Eyke Hllermeier, and Krzysztof J. Dembczynski. "Graded multilabel classification: The ordinal case." Proceedings of

Thank you
Questions?