Part 2: Learning Soft Constraints

Luc De Raedt, Andrea Passerini, Stefano Teso

Why soft constraints?

- Deal with conflicting requirements (e.g. multi-objective optimization)
- Combine knowledge and uncertainty (probabilistic relational models, fuzzy logic)
- Combine statistical and relational approaches to learning (statistical relational learning)

Outline

- Learning weights in (weighted) MAX-SAT problems
- Learning weights in (weighted) CSP problems
- Learning weights in (weighted) COP problems
- Selecting constraints and learning weights
- Learning constraints and weights (hints)

Learning weights in (weighted) MAX-SAT problems

E.g: Markov Logic networks [RD06]

Definition

- A Markov Logic Network (MLN) L is a set of pairs (F_i, w_i) where:
  - F_i is a formula in first-order logic
  - w_i is a real number (the weight of the formula)

Example

w_1 $\forall x \, (\texttt{Bird(x)} \Rightarrow \texttt{Flies(x)})$

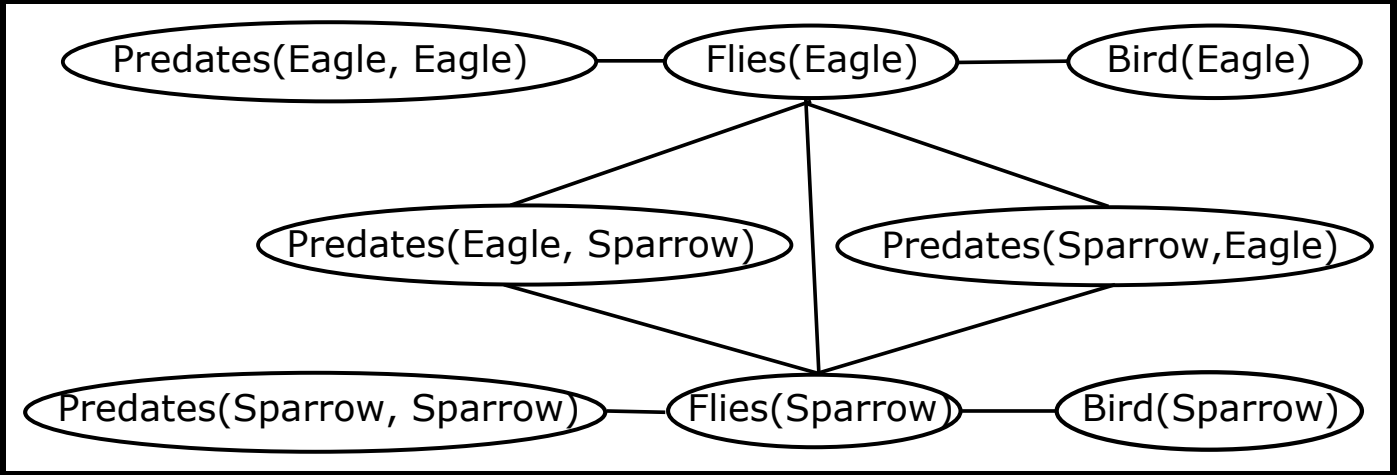
w_2 $\forall x, y \, (\texttt{Predates(x,y)} \wedge \texttt{Flies(y)} \Rightarrow \texttt{Flies(x)})$

Markov Logic networks

Intuition

- A MLN is a template for Markov Networks, based on logical descriptions
- Applied to a set of constants (entities) it defines a Markov Network
- Single atoms in the template generate nodes in the network
- Formulas in the template generate cliques in the network

Markov Logic networks: example



Ground network

- The following MLN:

$$w_1 \quad \forall x \, (\texttt{Bird(x)} \Rightarrow \texttt{Flies(x)})$$

$$w_2 \quad \forall x, y \, (\texttt{Predates(x,y)} \wedge \texttt{Flies(y)} \Rightarrow \texttt{Flies(x)})$$

- applied to a set of two constants $\{\texttt{Sparrow}, \texttt{Eagle}\}$
- generates the Markov Network shown in figure

Joint probability

- A ground MLN specifies a joint probability distribution over possible worlds (i.e. truth value assignments to all ground atoms)
- The probability of a possible world x is:

$$p(x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right)$$

where:

  - the sum ranges over formulas in the MLN (i.e. clique templates in the Markov Network)
  - $n_i(x)$ is the number of true groundings of formula F_i in x
  - The partition function Z sums over all possible worlds (i.e. all possible combination of truth assignments to ground atoms)

Inference

- Compute value of x with maximal probability

$$x^* = \underset{x}{\text{argmax}} \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right) = \underset{x}{\text{argmax}} \sum_{i=1}^{F} w_i n_i(x)$$

  $\Rightarrow$ boils down to weighted MAX-SAT.

- Compute value of x with max. probability given evidence e

$$x^* = argmax_x p(x|e)$$

  where evidence fixes the value of some of the variables in x

Learning

- Learn weights of (given) formulas
  - parameter learning [LD07, HM09]
- Learn both formulas and weights
  - structure learning [KD05]

Weight Learning in Markov Logic Networks: example

- Set of formulas

f_1 $\forall x\,(\texttt{Bird(x)} \Rightarrow \texttt{Flies(x)})$

f_2 $\forall x, y\,(\texttt{Predates(x,y)} \wedge \texttt{Flies(y)} \Rightarrow \texttt{Flies(x)})$

- Dataset (only true facts)
  - `Bird(Hawk),Flies(Hawk),Bird(Sparrow),Flies(Sparrow), Predator(Hawk,Sparrow)`
  - `Bird(Eagle),Flies(Eagle),Predator(Eagle,Rabbit)`
  - `Bird(Falcon),Flies(Falcon),Bird(Crow),Flies(Crow), Predator(Falcon,Crow)`
  - `Bird(Turkey),Predator(Fox, Turkey)`
- Weight learning will assign higher weight to f_2 wrt f_1.

Learning weights in (weighted) CSP problems

Weighted Constraint Satisfaction Problems (wCSP)

Definition

Given

- A set of pairs $\{(c_i, w_i)\}_{i=1}^{n}$ where:
  - c_i is a (soft) constraint
  - $w_i \in \mathbb{R}$ is a weight
- An indicator function $\mathbb{1}\{x \models c\}$ evaluating to one if c is satisfied by x, and zero otherwise

Find

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\, f(x) = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \sum_{i=1}^{n} w_i \cdot \mathbb{1}\{x \models c_i\}$$

Note

Hard constraints are incorporated in $\mathcal{X}$

Learning weights for wCSP

Preference learning [FH10]

Given

- A set of (soft) constraints $\{(c_i)\}_{i=1}^{n}$
- A set of examples pairs $\mathcal{D} = \{(x_j, x'_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$f(x_j) > f(x'_j)$$

- An **objective** function $J(w, \mathcal{D}) = \underbrace{\Omega(w)}_{\textit{model complexity}} + \lambda \underbrace{\ell(w, \mathcal{D})}_{\textit{training loss}}$

Find

- A set of weights $\hat{w}$ minimizing the objective:

$$\hat{w} = \operatorname*{argmin}_{w} J(w, \mathcal{D})$$

Preference learning for wCSP

E.g. SVM ranking [Joa02]

$$\begin{array}{ll} \min\limits_{w} & ||w||^2 + \lambda \sum_{j=1}^{m} \xi_j \\ s.t & f(x_j) - f(x'_j) \geq 1 - \xi_j \quad \forall j \in [1, n] \end{array}$$

where:

- ξ_j is a penalty for not ranking x_j higher than x'_j with a large enough margin
- $||w||^2$ is a regularization term (margin is $2/||w|| \rightarrow$ large margin separation)
- $\lambda \in R^+$ is a parameter trading off margin and correct rankings

SVM ranking for wCSP: example

- Constraints

c_1 $\neg$`ParkNearby(x)` $\Rightarrow$ `Garden(x)`

c_2 $\neg$`FreeParkingNearby(x)` $\Rightarrow$ `Garage(x)`

c_3 `NumberOfBathrooms(x) > 1`

- Dataset
  - {`NumberOfBathrooms(Home1) > 1`} preferred to {`Garden(Home2)`, `FreeParkingNearby(Home2)`}
  - {`Garage(Home3)`} preferred to {`Garden(Home4)`}
- SVM ranking assigns decreasing weights to c_3, c_2, c_1.

Learning weights for wCSP

Structured-output learning [BHS+07]

Given

- A set of (soft) constraints $\{(c_i)\}_{i=1}^{n}$
- A set of input-output pairs $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$y_j = \operatorname*{argmax}_{y \in \mathcal{Y}} f(x_j, y_j)$$

- An **objective** function $J(w, \mathcal{D})$

Find

- A set of weights $\hat{w}$ minimizing the objective:

$$\hat{w} = \operatorname*{argmin}_{w} J(w, \mathcal{D})$$

E.g. Structured-output SVM [TJHA05]

$$\begin{array}{ll} \min\limits_{w} & ||w||^2 + \lambda \sum_{j=1}^{m} \xi_j \\ s.t & f(x_j, y_j) - f(x_j, y'_j) \geq 1 - \xi_j \quad \forall j \in [1, n] \quad \forall y'_j \neq y_j \end{array}$$

where:

- ξ_j is a penalty for not ranking y_j higher than any alternative output y'_j with a large enough margin

Problem

The number of constraints is equal to $m \times (|\mathcal{Y}| - 1)$ and is typically exponential in the number of output variables

Cutting plane algorithm

1. Initialize weights $w = 0$ and set of constraints $\mathcal{S}_j = \varnothing$ for each example j
2. While constraint added, for each example (x_j, y_j)
   - 2.1 Check penalty using current S_j

$$\xi_j = \max_{y'_j \in S_j} 1 + f(x_j, y'_j) - f(x_j, y_j) \quad [\text{weighted CSP problem!!}]$$

   - 2.2 Check penalty in full space $\mathcal{Y}$

$$\xi_j^{new} = \max_{y'_j \neq y_j} 1 + f(x_j, y'_j) - f(x_j, y_j) \quad [\text{weighted CSP problem!!}]$$

   - 2.3 If $\xi_j^{new} - \xi_j > \epsilon$
     - 2.3.1 Add constraint and update S_j
     - 2.3.2 Retrain

Structured-output SVM for wCSP: example

- Constraints

  c_1 $\texttt{Interact(x,x')} \Rightarrow \texttt{Loc(x,y)} \wedge \texttt{Loc(x',y)}$

  c_2 $\texttt{Loc(x, Extra)} \Rightarrow \texttt{Signal(x)}$

- Dataset
  - {Interact(P1, P2), Loc(P1, Nucleus), Loc(P2, Nucleus)},
  - {Loc(P3, Nucleus), Loc(P4, Cytosol)},
  - {Interact(P5, P6), Loc(P5, Cytosol), Loc(P6, Cytosol)},
  - {Interact(P7, P8), Loc(P7, Membrane), Loc(P8, Extra), Signal(P8)},
  - {Loc(P9, Extra), Signal(P9)},
  - {Loc(P10, Extra), Signal(P10)}
- Structured-output SVM assigns larger weight to c_2 than c_1

Learning weights in (weighted) COP problems

Weighted Constraint Optimization Problems (wCOP)

(possible) Definition

Given

- A set of triplets $\{(c_i, w_i, \phi_i)\}_{i=1}^{n}$ where:
  - c_i is a (soft) constraint
  - $w_i \in \mathbb{R}$ is a weight
  - ϕ_i is a cost function mapping c_i and x to a real value (e.g. a measure of how far x is from satisfying c_i)

Find

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x) = \underset{x \in \mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \cdot \phi(x, c_i)$$

Note

It is more natural to model the problem as cost minimization

Learning weights for wCOP [TSP17]

Preference learning

Given

- A set of (soft) constraints and cost functions $\{(c_i, \phi_i)\}_{i=1}^{n}$
- A set of examples pairs $\mathcal{D} = \{(x_j, x'_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$f(x_j) > f(x'_j)$$

- An **objective** function $J(w, \mathcal{D})$

Find

- A set of weights $\hat{w}$ minimizing the objective:

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, J(w, \mathcal{D})$$

E.g. SVM ranking

$$\begin{aligned} \min_{w} \quad & ||w||^2 + \lambda \sum_{j=1}^{m} \xi_j \\ s.t \quad & f(x_j) - f(x'_j) \geq 1 - \xi_j \quad \forall j \in [1, n] \end{aligned}$$

where:

- ξ_j is a penalty for not ranking x_j higher than x'_j with a large enough margin
- $||w||^2$ is a regularization term (margin is $2/||w|| \rightarrow$ large margin separation)
- $\lambda \in R^+$ is a parameter trading off margin and correct rankings

SVM ranking for wCOP: example

- Decision variables

var	description	var	description
x_1	has garden	x_2	has park nearby
x_3	crime rate	x_4	distance from parents
x_5	distance from kindergarten		

- Constraints and cost functions

$$c_1 = (\neg x_2 \Rightarrow x_1) \qquad \varphi_1 = \mathbb{1}\{\neg c_1\}$$
$$c_2 = (x_3 \leq \theta_1) \qquad \varphi_2 = \mathbb{1}\{\neg c_2\} \cdot (x_3 - \theta_1)$$
$$c_3 = (x_4 \leq \theta_2) \qquad \varphi_3 = \mathbb{1}\{\neg c_3\} \cdot (x_4 - \theta_2)$$
$$c_4 = (x_5 \leq \theta_3) \qquad \varphi_4 = \mathbb{1}\{\neg c_4\} \cdot (x_5 - \theta_3)$$

Learning weights for wCOP

Structured-output learning

Given

- A set of (soft) constraints and cost functions $\{(c_i, \phi_i)\}_{i=1}^{n}$
- A set of input-output pairs $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$y_j = \operatorname*{argmin}_{y \in \mathcal{Y}} f(x_j, y_j)$$

- An **objective** function $J(w, \mathcal{D})$

Find

- A set of weights $\hat{w}$ minimizing the objective:

$$\hat{w} = \operatorname*{argmin}_{w} J(w, \mathcal{D})$$

E.g. Structured-output SVM

$$\begin{aligned} \min_{w} \quad & ||w||^2 + \lambda \sum_{j=1}^{m} \xi_j \\ s.t \quad & f(x_j, y'_j) - f(x_j, y_j) \geq 1 - \xi_j \quad \forall j \in [1, n] \quad \forall y'_j \neq y_j \end{aligned}$$

where:

- ξ_j is a penalty for not giving y_j a lower cost than to any alternative output y'_j with a large enough margin

Problem

The number of constraints is equal to $m \times (|\mathcal{Y}| - 1)$ and is typically exponential in the number of output variables

Structured-output SVM for wCOP

Cutting plane algorithm

1. Initialize weights $w = 0$ and set of constraints $\mathcal{S}_j = \varnothing$ for each example j
2. While constraint added, for each example (x_j, y_j)
   - 2.1 Check penalty using current S_j

     $$\xi_j = \max_{y'_j \in S_j} 1 + f(x_j, y_j) - f(x_j, y'_j) \quad [\text{weighted COP problem!!}]$$

   - 2.2 Check penalty in full space $\mathcal{Y}$

     $$\xi_j^{new} = \max_{y'_j \neq y_j} 1 + f(x_j, y_j) - f(x_j, y'_j) \quad [\text{weighted COP problem!!}]$$

   - 2.3 If $\xi_j^{new} - \xi_j > \epsilon$
     - 2.3.1 Add constraint and update S_j
     - 2.3.2 Retrain

Selecting constraints and learning weights

Selecting constraints and learning weights for wCOP [Pas16]

Constraint Selection and Preference learning

Given

- A set of candidate (soft) constraints and cost functions $\mathcal{C} = \{(c_i, \phi_i)\}_{i=1}^{n}$
- A set of examples pairs $\mathcal{D} = \{(x_j, x'_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$f(x_j) > f(x'_j)$$

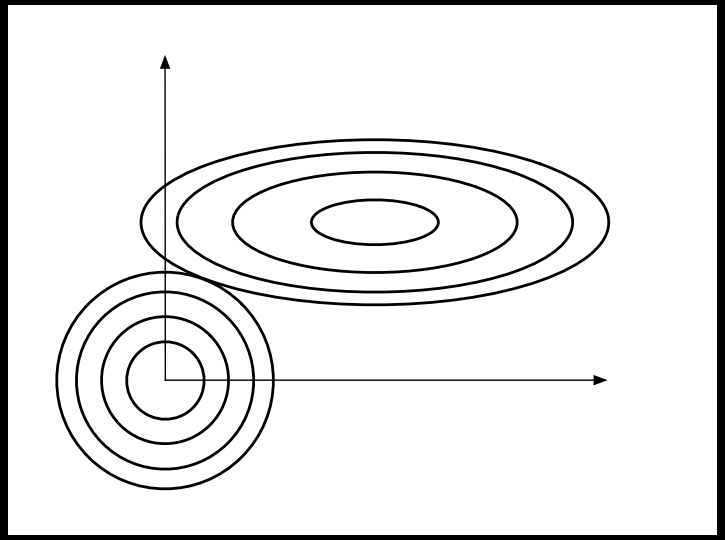
- An **objective** function $J(w, \mathcal{D})$

Find

- A subset of the constraints $\hat{\mathcal{C}} \subseteq \mathcal{C}$ and a set of weights $\hat{w}$ s.t.:

$$(\hat{\mathcal{C}}, \hat{w}) = \underset{\mathcal{C}' \subseteq \mathcal{C}}{\operatorname{argmin}} \underset{w \in \mathbb{R}^{|\mathcal{C}'|}}{\operatorname{argmin}} J(w, \mathcal{D})$$

Regularization for Constraint Selection

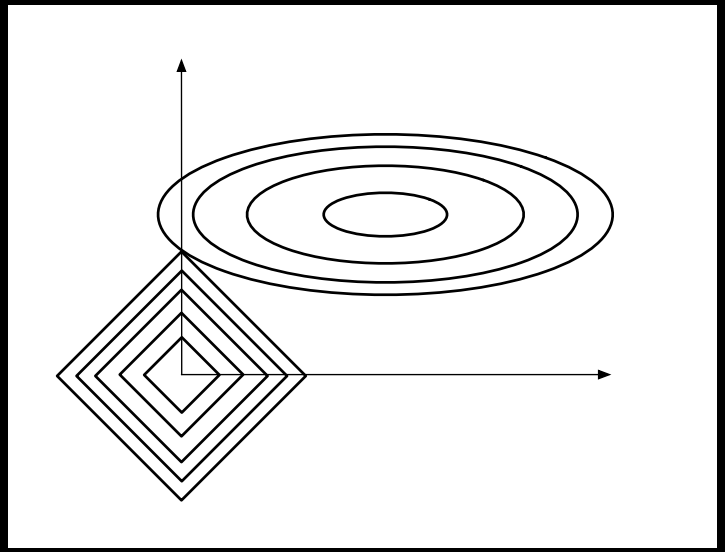


2-norm regularization

$$J(w) = ||w||^2 + \lambda E(w)$$

- Penalizes weights by (squared) Euclidean norm
- Weights with less influence on error get smaller values
- No explicit bias towards exactly zero weights

Regularization for Constraint Selection



1-norm regularization [Tib94]

$$J(w) = |w| + \lambda E(w)$$

- Penalizes weights by sum of absolute values
- Encourages less relevant weights to be exactly zero (sparsity inducing norm)

E.g. SVM ranking with 1-norm regularization

$$\begin{aligned} \min_{w} \quad & |w| + \lambda \sum_{j=1}^{m} \xi_j \\ s.t \quad & f(x_j) - f(x'_j) \geq 1 - \xi_j \quad \forall j \in [1, n] \end{aligned}$$

where:

- ξ_j is a penalty for not ranking x_j higher than x'_j
- $|w|$ is a sparsity inducing regularization term
- $\lambda \in R^+$ trades off sparsity and correct rankings

Note

Structured-output learning can also be adapted by replacing two-norm with one-norm

Learning constraints and weights (hints)

Constraint Learning and Preference learning

Given

- A language $\mathcal{L}$ defining valid constraints
- A set of examples pairs $\mathcal{D} = \{(x_j, x'_j)\}_{j=1}^{m}$ such that for all j it should hold that

$$f(x_j) > f(x'_j)$$

- An **objective** function $J(w, \mathcal{D})$

Find

- A set of valid constraints according to $\mathcal{L}$ and a set of weights $\hat{w}$ s.t.:

$$(\hat{\mathcal{C}}, \hat{w}) = \underset{\mathcal{C}' \in \mathcal{L}}{\text{argmin}} \ \underset{w \in \mathbb{R}^{|\mathcal{C}'|}}{\text{argmin}} \ J(w, \mathcal{D})$$

Constraint Learning and Preference learning for wCSP (hints)

Two step approach [MLAS05]

1. Run hard constraint learning algorithm to get set of candidate constraints $\mathcal{C}$
2. Run constraint selection and preference learning on $\mathcal{C}$

Combined approach [KD05, LPRF06]

1. Start with empty set of constraints $\mathcal{C} = \varnothing$
2. While no improvement
   - 2.1 Use language bias from hard constraint learning to generate candidate extensions $\mathcal{C}'$ of constraints in $\mathcal{C}$
   - 2.2 Replace $\mathcal{C}$ with $\mathcal{C}'$
   - 2.3 Run constraint selection and preference learning on $\mathcal{C}$
   - 2.4 Discard zero weight constraints from $\mathcal{C}$

Wrapping up

What we saw

- Soft constraints are useful for modelling uncertain knowledge and need for compromise
- Learning weights of known constraints can be cast into existing machine learning approaches
- Sparsification techniques can help in selecting constraints
- Hybrid approaches can (try to) address learning unknown (soft) constraint

Open problems

- Constraint solving/optimization used as subroutines $\rightarrow$ problem of efficiency (need approximate techniques)
- Learning unknown soft constraints is difficult. Only addressed for SAT constraints


Gükhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. **Predicting Structured Data (Neural Information Processing).** The MIT Press, 2007.


Johannes Fürnkranz and Eyke Hüllermeier. **Preference Learning, pages 789–795.** Springer US, Boston, MA, 2010.


Tuyen N. Huynh and Raymond J. Mooney. **Max-margin weight learning for markov logic networks.** In Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 564–579, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.


Thorsten Joachims.
Optimizing search engines using clickthrough data.
In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.


Stanley Kok and Pedro Domingos.
Learning the structure of markov logic networks.
In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 441–448, New York, NY, USA, 2005. ACM.


Daniel Lowd and Pedro Domingos.
Efficient weight learning for markov logic networks.
In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD

2007, pages 200–211, Berlin, Heidelberg, 2007. Springer-Verlag.


Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi.
kfoil: Learning simple relational kernels.
In *AAAI*, pages 389–394. AAAI Press, 2006.


Stephen Muggleton, Huma Lodhi, Ata Amini, and Michael J. E. Sternberg.
Support vector inductive logic programming.
In *Discovery Science, 8th International Conference, DS 2005, Singapore, October 8-11, 2005, Proceedings*, pages 163–175, 2005.


Andrea Passerini.
Learning Modulo Theories, pages 113–146.
Springer International Publishing, Cham, 2016.


Matthew Richardson and Pedro Domingos.
Markov logic networks.
Mach. Learn., 62(1-2):107–136, 2006.


Robert Tibshirani.
Regression shrinkage and selection via the lasso.
Journal of the Royal Statistical Society, Series B, 58:267–288, 1994.


I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun.
Large margin methods for structured and interdependent output variables.
jmlr, 6:1453–1484, 2005.


Stefano Teso, Roberto Sebastiani, and Andrea Passerini.
Structured learning modulo theories.
Artificial Intelligence, 244:166 – 187, 2017.
Combining Constraint Solving with Mining and Learning.