ARC: Adversarial Robust Cuts for Semi-Supervised and Multi-Label Classification

Sima Behpour, Wei Xing, and Brian D. Ziebart
Department of Computer Science
University Of Illinois at Chicago
{sbehpo2,wxing3,bziebart}@uic.edu

Abstract

Many structured prediction tasks arising in computer vision and natural language processing tractably reduce to making minimum cost cuts in graphs with edge weights learned using maximum margin methods. Unfortunately, the hinge loss used to construct these methods often provides a particularly loose bound on the loss function of interest (e.g., the Hamming loss). We develop Adversarial Robust Cuts (ARC), an approach that poses the learning task as a minimax game between predictor and "label approximator" based on minimum cost graph cuts. Unlike maximum margin methods, this game-theoretic perspective always provides meaningful bounds on the Hamming loss. We conduct multi-label and semi-supervised binary prediction experiments that demonstrate the benefits of our approach.

1. Background

1.1. Notation and Learning Task

We consider n predicted variables, $\mathbf{y} = (y_1, \ldots, y_n)$, chosen from a fixed set of labels $y_i \in \mathcal{Y}, \forall i \in [n]$, where $[n] = \{1, \ldots, n\}$. We denote the corresponding random variables for these label variables using capitalization, $\mathbf{Y} = (Y_1, \ldots, Y_n)$, and denote vectors and multivariate variables in bold. We denote given information or side information variables using a single vector, $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, with a corresponding random variable denoted as $\mathbf{X}$. (Strict subportions of $\mathbf{x}$ may be relevant to each variable y_i, but for notational simplicity, we do not denote such partitions in our formulation.) Our task in this setting is to make predictions for $\mathbf{y}$ given an input $\mathbf{x}$ and a set of m training example pairs, $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})_{j \in [m]}$, where we index training examples using a parenthetical superscript notation whenever necessary to disambiguate between different examples or denote this distribution as $\tilde{P}(\mathbf{X}, \mathbf{Y})$. Aiding in this task are a set of features relating the input variables to the predicted variables and to one another. We generically denote these feature vectors as $\boldsymbol{\phi}_{\mathbf{c}}(\mathbf{y_c}, \mathbf{x})$ for relationships over variables in some subset of the $\mathbf{y}$ variables denoted by $\mathbf{c} \in \mathcal{C} \subseteq 2^{[n]}$. For a subset $\mathbf{c} = \{c_1, \ldots, c_l\}$ which contains l variables, $\mathbf{y_c} = \{y_{c_1}, \ldots, y_{c_l}\}$ is the corresponding set of label values for the variables in the subset. For pairwise relationships between y_i and y_j that also incorporate input variables, this reduces to feature functions denoted as $\boldsymbol{\phi}_{i,j}(y_i, y_j, \mathbf{x})$ and to $\boldsymbol{\phi}_i(y_i, \mathbf{x})$ for univariate feature functions. For many datasets, variables that are closely related to one another tend to have the same label. For example, pixels with similar characteristics in the same region of an image tend to belong to the same image segment. To capture this property, we define pairwise features that reflect the difference when two variables have different labels, and use a generalized Potts model [5] to penalize assignments that do not have the same label across the edges:

$$\boldsymbol{\phi}_{i,j}(y_i, y_j, \mathbf{x}) = I(y_i \neq y_j)\boldsymbol{\delta}_{i,j}(y_i, y_j, \mathbf{x}), \qquad (1)$$

where $I()$ is a indicator function whose value is 1 only if the inner logical expression is true.

$\mathbf{y} = [\ 1, 1, 0, 0, 1, 1, 0, 0 \ldots\ ]$

Figure 1. An example image and its multilabel annotation vector for label set: *sky*, *clouds*, *trees*, *sunset*, *sea*, *ship*, *mountains*, *desert*, ...

We consider the multilabel prediction task of annotating images as a running example. Each training image, $\mathbf{x}$, has an associated vector of labels, $\mathbf{y}$, corresponding to different descriptors of the image, as illustrated in Figure 1. We define unary and pairwise features for the labels as:

$$\boldsymbol{\phi}_{sky}(y_{sky}, \text{  }) = I(y_{sky}=1)\,\mathbf{imgFeatures}(\text{  }) \qquad (2)$$

$$\boldsymbol{\phi}_{sky,clouds}(y_{sky}, y_{clouds}) = I(y_{sky} \neq y_{clouds}) \qquad (3)$$
$$|\ \mathbf{word2vec}(sky) - \mathbf{word2vec}(clouds)|^{-1},$$

using features from the Mulan dataset [8] for image representations and a deeply learned word embedding[1] for word semantics.[2]

In this paper, we focus on problems evaluated using the Hamming loss:

$$\text{loss}(\hat{\mathbf{y}}, \check{\mathbf{y}}) = \frac{1}{n} \sum_i I(\hat{y}_i \neq \check{y}_i), \tag{4}$$

which measures the fraction of the labels that are correctly predicted in the multilabel annotation task.

1.2. Markov Networks and Intractability

Estimating the conditional probability of label variables using a Markov network is one powerful approach to this structured prediction task. Markov networks can be written as log-linear models when their densities are positive. A Markov network has the following probability distribution:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\boldsymbol{\Psi}(\mathbf{y},\mathbf{x})}, \tag{5}$$

where the potential function $\boldsymbol{\Psi}$ decomposes into a set of potential functions over subsets of the $\mathbf{y}$ variables, $\boldsymbol{\Psi}(\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{c} \in \boldsymbol{\mathcal{C}}} \boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})$, with these subset potentials defined as $\boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x}) = \theta_{\mathbf{c}} \cdot \boldsymbol{\phi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})$ using a vector of estimated weights $\theta_{\mathbf{c}}$ that is specific to each $\mathbf{c}$. Parameter sharing with clique $\mathbf{c}'$, $\boldsymbol{\phi}_{\mathbf{c}} = \boldsymbol{\phi}_{\mathbf{c}'}$, can be employed to reduce the effective number of learned parameters of the model. The structure of these potentials corresponds to an undirected graphical model in which the variables in set $\mathbf{c}$ are connected by undirected edges, forming cliques in the graph.

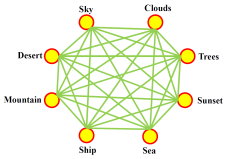

Figure 2. The Markov network corresponding to the multilabel annotation prediction of Figure 1.

In our running example, all unary and pairwise subsets of variables are inlcuded in $\boldsymbol{\mathcal{C}} = \{\{sky\}, \{clouds\}, \{trees\}, \ldots, \{sky, clouds\}, \{sky, trees\}\}$, and the corresponding Markov network is the complete graph over all of these class labels (Figure 2). Unfortunately, even when restricted to pairwise and unary potential functions, the most probable assignment of values, $\mathbf{y}^* = \text{argmax}_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} P(\mathbf{y}|\mathbf{x})$, and the normalization term, $Z(\mathbf{x}) = \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} e^{\sum_{\mathbf{c} \in \boldsymbol{\mathcal{C}}} \boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})}$, are both intractable to compute for Markov networks in general [9]. Restrictions are often placed on the potential functions so that the corresponding undirected graph has low tree-width (e.g., chains, trees), which enables efficient maximization and normalization computations [9].

[1] https://code.google.com/p/word2vec/

[2] We use element-wise operations to compute and invert the differences between each embedded dimension.

1.3. Minimum-Cuts and Associative Markov Networks

Another direction for realizing tractable Markov networks exploits potential functions for which maximization can be solved efficiently, even though normalization is intractable due to the large tree-widths of their graphs. Binary-valued Markov networks with non-negative pairwise potentials are one example of this. Their maximum value assignments can be obtained using minimum-cut/maximum-flow algorithms, as shown in Figure 3. Edges from the source and to the sink nodes are weighted based on unary feature potentials, and edges between predicted variables are weighted based on the pairwise feature potentials. Large potentials prevent certain edge cuts (and corresponding value assignments to the connected sink or source) from the solution. In our running example, for instance, two semantically related words (e.g., *sky* and *clouds*) are likely to have large learned potentials that prevent one from being an included label without the other. This class of models has been employed extensively in computer vision applications for binary image denoising and segmentations problems [3, 2].

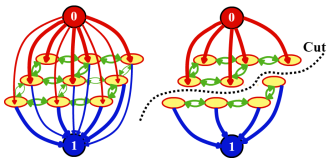

Figure 3. A directed graph used to augment a Markov network (left) so that the minimum cut (right) provides the most probable assignment of each variable based on its connection to the source node (0) or target node (1).

2. Applications

We focus our attention on binary-valued structured prediction tasks for which inference can be efficiently performed without extremely restrictive limitations on the potential functions or using approximation.

2.1. Semi-Supervised Classification

We first consider cut-based semi-supervised classification [1] using four datasets from the UCI repository [4]. The characteristics of these datasets are summarized in Table 1. The vector $\mathbf{y}$ corresponds to the examples of each dataset. Following previous work, we seek to leverage the relationships between each example, in terms of input values $\mathbf{x}_i$, and its label, y_i, along with relationships between pairs of labels (y_i, y_j) (and their inputs $(\mathbf{x}_i, \mathbf{x}_j)$). We construct unary features directly from each example's input vector $\mathbf{x}_i$ and pairwise features as the inverse of the absolute difference of the two corresponding nodes features. We share the same unary and pairwise parameters across all edges so that these potentials can be applied to previously unseen examples at

Table 1. Semi-supervised classification dataset characteristics; training/testing hinge loss and testing Hamming loss for SSVM; number of testing-time cuts, training/testing game value and testing Hamming loss for our ARC approach.

Dataset Information				SSVM			ARC			
Name	#training	#testing	#features	Hinge_{tr}	Hinge_{te}	Hamming_{te}	#cuts	Value_{tr}	Value_{te}	Hamming_{te}
Diabetes	600	168	8	1.27	1.22	0.37	9	0.39	0.35	0.31
Breast Cancer	500	183	10	0.44	0.58	0.12	8	0.42	0.23	0.10
Gisette	800	200	4971	1.26	0.54	0.21	17	0.45	0.29	0.16
Spect	187	80	22	1.30	1.28	0.29	10	0.38	0.34	0.26

Table 2. Multi-label dataset information and average testing Hamming loss for binary relevance (BR), multi-label KNN (ML-KNN), and Rank-based support vector machines (Rank SVM), and our ARC approach (with average number of cuts).

Dataset Information					Test Hamming Loss				
Name	Domain	#Instances	#Features	#Labels	BR	MLKNN	Rank SVM	ARC	#cuts
Bibtex	text	7395	1836	159	0.015 ± 0.001	0.017 ± 0.001	0.120 ± 0.014	0.015 ± 0.001	20.8
Bookmarks	text	87856	2150	202	0.238 ± 0.018	0.149 ± 0.011	0.176 ± 0.016	0.141 ± 0.014	19.3
Birds	audio	645	260	19	0.156 ± 0.106	0.063 ± 0.001	0.124 ± 0.106	0.062 ± 0.010	7.4
CAL500	music	502	68	174	0.159 ± 0.016	0.113 ± 0.012	0.124 ± 0.016	0.102 ± 0.018	14.9
Emotions	music	593	72	6	0.261 ± 0.018	0.198 ± 0.016	0.183 ± 0.012	0.174 ± 0.010	13.7
Flags	images	194	19	7	0.271 ± 0.220	0.236 ± 0.014	0.234 ± 0.011	0.212 ± 0.010	18.5
Scene	images	2407	294	6	0.139 ± 0.010	0.144 ± 0.012	0.241 ± 0.015	0.110 ± 0.016	12.1
Yeast	biology	2417	103	14	0.238 ± 0.015	0.195 ± 0.110	0.210 ± 0.090	0.186 ± 0.014	11.6
NUS-WIDE	images	269648	128	81	0.120	0.028	0.102	0.020	14.5
				Average	0.177	0.127	0.168	0.113	14.8

test time. During training time, we only incorporate labeled training examples. At testing time, we incorporate both the training set and the unlabeled testing set on which predictions are desired.

We compare our ARC approach with a structured support vector machine [6, 7] on the same feature representation.

2.2. Multi-Label Prediction

The second application that we investigate is multi-label classification, like our running example. In this setting, multiple labels can be attached to each example and the prediction task is that of predicting some subset of the label set for each example. We treat each of the labels as a binary variable and follow the structure presented in the previous section to train an adversarial multi-label predictor by learning to make adversarial cuts. Most of the features we employ as unary and pairwise features are taken from the Mulan dataset [8]. As shown in Table 2, our ARC approach performs at least as well as the other methods on each individual dataset, and much better on average.

3. Discussion

We investigated a robust approach for learning to make cuts in graphs. It operates by making worst-case approximations to the training labels. This has benefits theoretically—providing meaningful bounds on losses—and in practice, as illustrated by our experiments. In future work, we plan to investigate the benefits of our game formulation for multiclass problems where only approximately optimal graph cuts can be obtained. We expect that because the equilibrium is defined over many different cuts, rather than the single best alternative (as in structured SVM's hinge loss), that approximations will have a less detrimental impact on our approach.

References

[1] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26. Morgan Kaufmann Publishers Inc., 2001.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[3] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[4] M. Lichman. UCI machine learning repository, 2013.

[5] R. B. Potts. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pages 106–109. Cambridge Univ Press, 1952.

[6] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the ICML*, pages 896–903. ACM, 2005.

[7] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the ICML*, page 104. ACM, 2004.

[8] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *JMLR*, 12:2411–2414, 2011.

[9] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.