Machine Learning 2 - Final Project

Ron Dagani 318170917

Noam Ginio 043543537

Predicting Structured Objects with Support Vector Machines

By Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu, 2009

[1]

A. Articles summary

Abstract:

This study presents a new structural output prediction method development, called Structural SVMs, by utilizing generalization of Support Vector Machines (SVMs).

The method solves the challenge of predicting complex objects like trees, orderings, or alignments that cannot be solved by conventional classification or regression methods. These challenges arise in various fields, such as natural language processing, search engines, and bioinformatics.

1. introduction

First, we'll explain the problem we want to solve through an example. We'll look at the problem of natural language parsing. A parser takes as input a natural language sentence. The desired output is the parse tree decomposing the sentence into its constituents.

As depicted in Figure 1, the desired output is not a simple single value that can be inferred by classification or regression by using methods. Hence, conventional methods such as Boosting, Bagging, and Support Vector Machines, we won't be able to get the desired output prediction.

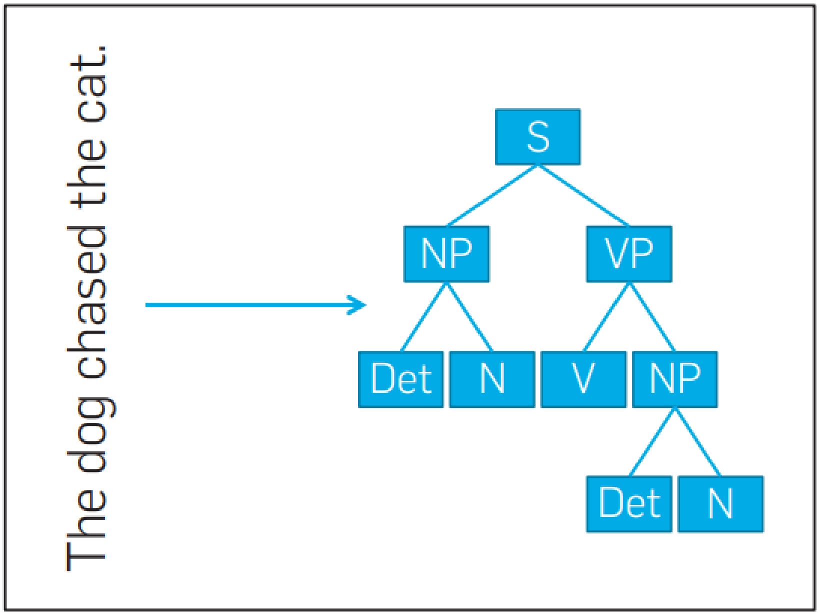


Figure 1: Example of predicting trees in natural language parsing structured output prediction task

A naive thought would be that we might be able to break the complex desired output to simple single values. However, it isn't so straightforward, because there are interdependencies between individual predictions. Furthermore, we would like to be able to evaluate the prediction for the entire structure as a whole and not only by its individual components.

This problem is called Structured output prediction. It is defined as learning tasks that are trying to learn a map function $h: X \rightarrow Y$ s.t $y \in Y$ is a complex structure. Notice, $|Y|$ may be exponential in the length of the input. Meaning, we have a multiclass learning task where the number of the possible classes is exponential in the length of the input, as presented in Figure 2.

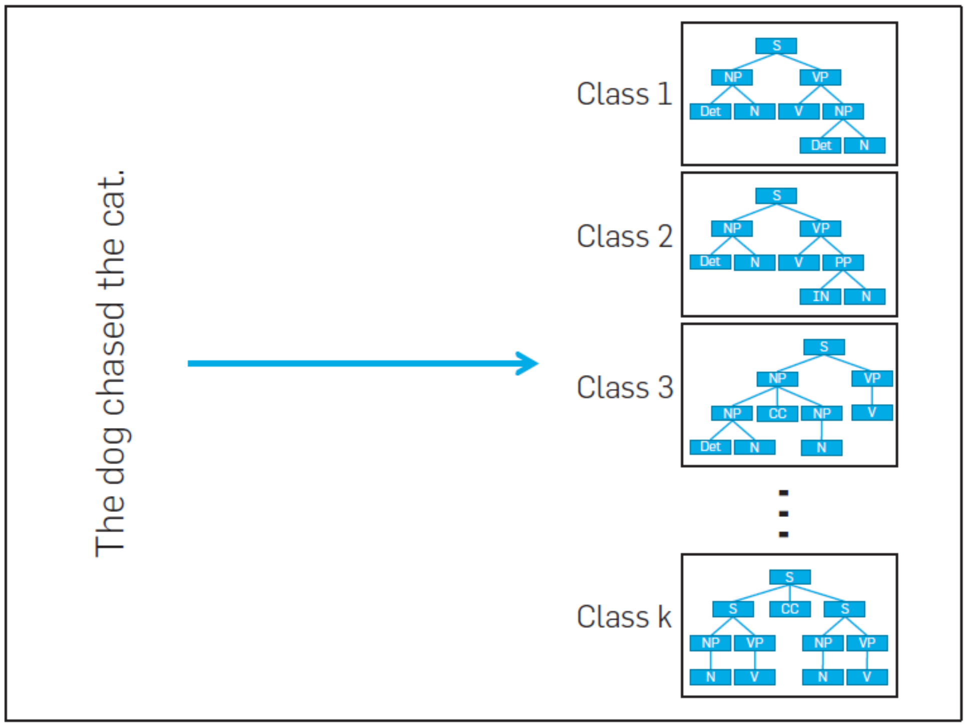


Figure 2: Structured output prediction as a multiclass problem.

This study utilizes a generalization of SVMs to develop a new method called Structural SVMs [2,3,4]. This model can be used for an extensive range of structured and complex output prediction tasks. Structural SVMs method inherit the properties of conventional SVMs such as convex training problems,, the opportunity to learn nonlinear rules via kernels and the flexibility in the choice of loss function. Also, the method inherits the expressive power of generative models (e.g., Markov Random Fields) without requiring the independence assumptions made by conventional generative methods.

2. Structural SVMs

This method is based upon a line of research on discriminative training for structured output prediction, including generalizations of Neural Nets [5], Logistic Regression [6], and other conditional likelihood methods [7]. In particular in Lafferty et al's research [6], it was shown that structured prediction training can be formulated as a convex optimization problem. Thus, it can be solved with various conventional tools of Quadratic Programming (QP). The model is built on structural Perceptron and protein threading methods extended to a large-margin formulation with an efficient training algorithm.

2.1 Formulation the model:

We start the derivation of the structural SVM from the multiclass SVM [8]. The goal is to fit between inputs x and classes y. Multiclass SVMs use a weight vector w_y to predict class y for a given input X. Each input has a score for each class. Define $\Phi(x)$ to be a binary or numeric features extracted from x, the score of sample x to class y would be $f(x, y) \equiv w_y \cdot \Phi(x)$.

Every feature will have an additively weighted influence in the modeled compatibility between inputs x and classes y.The prediction: the classification of x is the prediction rule h(x) that chooses the highest-scoring class:

(1) $$h(x) \equiv argmax_{y \in Y} f(x, y)$$

to be the prediction (output). Y would get the correct result for input x provided the weights $w = (w_1, \ldots, w_k)$that have been chosen such that the inequalities $f(x, \hat{y}) < f(x, y)$ hold for all incorrect outputs $\hat{y} \neq y$.

Giving a training sample $(x_1, y_1), \ldots, (x_n, y_n)$ leads to a hard-margin formulation of the learning problem by requiring a fixed margin, meaning that the margin is equal to one, separation of all training examples (using the norm of w as a regularizer) as followed:

(2) $$min_w \frac{1}{2} ||w||^2, \ s.t. \ f(x_i, y_i) - f(x_i, \hat{y}) \geq 1 \ (\forall i, \hat{y} \neq y_i)$$

The optimization problem of classifying k classes has a $n(k - 1)$ inequalities that are linear and can expand $f(x_i, y_i) - f(x_i, \hat{y}) = (w_{yi} - w_{\hat{y}}) \cdot \Phi(x_i)$. Therefore, it is a convex quadratic program.

2.2 Efficient prediction:

Main challenge that occurs while trying to use this method on structured outputs is that while there is generalization across inputs x, there is no generalization across outputs since each class y has its own weight vector w_y. Moreover, since $|Y|$ might be very large or even infinite, this method of naively reducing structured output prediction to multiclass classification will lead to exponential growth in the overall parameters number.

So we could find a weight vector w that is built from pairs of input and output using a joint feature map $\Psi(x, y)$ (instead of $\Phi(x)$) that is linear. Such that on each training example it scores the correct output higher (by a fixed margin) than every other potential output while keeping low complexity by L2 regularization factor (i.e. small norm ||w||). $\Psi(x, y)$ would be designed specific for each problem. This allows us to get compatibility functions with contributions from combined properties of inputs and outputs.

The joint features will generalize the outputs and define meaningful scores even for outputs that were never actually observed in the training data.

This way, the number of parameters will equal the number of features extracted via Ψ , since we define compatibility functions via $f(x, y) \equiv w \cdot \Psi(x, y)$. Therefore, the number of parameters may not depend on $|Y|$. With this change, we can use the formulation (2) with the flexible definition of f via Ψ to arrive at the following hard-margin optimization problem for structural SVMs.

Since $|Y|$ may be exponential in the length of the input, search over Y may not always be feasible and computing $h(x)$ can be problematic. We'll require a structural matching between the compositional structure of the outputs and the joint feature map Ψ (not a specific sort). For example, in the problem of natural language parsing mentioned above, the suggested feature map will lead to a compatibility function where parse trees are scored by associating a weight with each production rule and by combining all weights additively. This way, a prediction can be computed efficiently using an existing algorithm called CKY-Parsing algorithm [4].

2.3 Inconsistent training data

Another challenge lies in the training data. Although we assumed that there is a solution to the equation:

$$(3) \qquad min_w \frac{1}{2} ||w||^2 , \ s.t \ w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{y}) \geq 1(\forall i, \hat{y} \neq y_i),$$

it may not be true due to training data inconsistency or because our model class is not powerful enough. Therefore, if we want to allow the model to make mistakes, we need to be able to quantify the degree of mismatch between a prediction and the correct output. The reason for that is that usually different incorrect predictions vary in quality. This is addressed by the use of a loss function (that fits specifically to the problem at hand). The loss function would measure the score of our predictions. Expected loss is minimized in the training by optimization algorithm. Similarly to the soft-margin SVM, we won't strictly enforce constraints, and we'll allow violations, but penalizes them in the overall objective function. One approach is to introduce slack variables that quantify the actual violation:

(4) $$f(x_i, y_i) - f(x_i, \hat{y}) \geq 1 - \xi_{i\hat{y}} (\forall \hat{y} \neq y_i)$$

as we can see, by choosing $\xi_{i\hat{y}} > 0$ we could get a smaller separation margin. We'll define a penalty for the margin violations. A common choice is to use $\frac{1}{n}\sum_{i=1}^{n} max_{\hat{y} \neq y_i}\{\Delta(y_i, \hat{y})\xi_{i\hat{y}}\}$.

Using this, we penalize violations more heavily for constraints associated with output that have a high loss with regard to the observed y_i. We can convert this back into a quadratic problem (with the slack rescaling formulation):

(5) $$min_{w,\xi\geq 0}\frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \quad s.t. \quad C > 0: \ w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{y}) \geq 1 - \frac{\xi_i}{\Delta(y_i, y_i)} (\forall i, \hat{y} \neq y_i)$$

C trades-off constraint violations with the geometric margin where effectively given by $\frac{1}{||w||}$).
Another option would be to rescale the margin to upper-bound the training loss. This leads to the following quadratic program:

(6) $$min_{w,\xi\geq 0}\frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \quad s.t. \quad w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{\ } y) \geq \Delta(y_i, \hat{y}) - \xi_i (\forall i, \hat{y} \neq y_i)$$

Which is easier to use, so we'll use it for now on.

2.4 Efficient training

As said before, we want to keep our training as efficient as possible. Therefore, when we try to find the best weight vector w for the quadratic program, we can't enumerate every option (the options are constraint due to $\hat{y}$ being constrained) in order to find the best solution to the problem. Hence, the suggested solution is using the cutting-plane Algorithm 1 or a similar algorithm for slack-rescaling such as, Stochastic Gradient Descent.

Algorithm 1 for training structural SVMs (margin-rescaling).

```
1:  Input: $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)), C, \varepsilon$
2:  $\mathcal{W} \leftarrow \emptyset$, $\boldsymbol{w} = \mathbf{0}$, $\xi_i \leftarrow 0$ for all $i = 1, \ldots, n$
3:  repeat
4:     for i=1, ...,n do
5:        $\hat{\mathbf{y}} \leftarrow \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) + \boldsymbol{w} \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}})\}$
6:        if $\boldsymbol{w} \cdot [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}})] < \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \xi_i - \varepsilon$ then
7:           $\mathcal{W} \leftarrow \mathcal{W} \cup \{\boldsymbol{w} \cdot [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}})] \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \xi_i\}$
8:           $(\boldsymbol{w}, \boldsymbol{\xi}) \leftarrow \text{argmin}_{\boldsymbol{w}, \xi \geq 0} \frac{1}{2} \boldsymbol{w} \cdot \boldsymbol{w} + \frac{C}{n} \Sigma_{i=1}^{n} \xi_i$ s.t. $\mathcal{W}$
9:        end if
10:    end for
11: until $\mathcal{W}$ has not changed during iteration
12: return($\boldsymbol{w}, \boldsymbol{\xi}$)
```

The algorithm first initializes the constraints to w = 0. Then, iteratively construct a working set of constraints w that is equivalent to the full set of constraints in the original quadratic program (up to a specified precision ε). For a given ε, the algorithm terminates when it has found an ε -accurate solution, since it verifies in Line 5 that none of the constraints of the quadratic program in (Eq. 6) is violated by more than ε. Terminate time is polynomial and independent of the output space cardinality (refined version of the algorithm even can terminate after adding only at most $O(C\varepsilon^{-1})$ constraints to W). Because the number of constraints is independent of |Y|, and independent of the number of training examples n, it makes it a very attractive training algorithm even for conventional SVMs.

The argmax in line 5 might be expensive to compute if the number of iterations is small but it's very related to the argmax for computing a prediction h(x). Hence, to utilize this algorithm to a new structured prediction problem, it is required to implement only $\Psi(x, y)$, $\Delta(y, \hat{y})$, and $argmax_{\hat{y} \in Y}\{\Delta(y_i, \hat{y}) + w \cdot \Psi(x_i, \hat{y})\}$. makes the algorithm general and easily transferable to new applications.

2.5 related approaches

The structural SVM method is closely connected to other methods, such as Conditional Random Fields and Maximum Margin Networks. The critical connection to those methods, which rely on probability, is to explain the joint feature map as conditional probability.

$$P(y|x) = \frac{1}{Z(x)} exp(w \cdot \Psi(x, y)) \tag{7}$$

Where z(x) is normalization constant representing all the possible options in the sampled space. This equation represent the general case of logistic regression where $\Psi(x, y) = y\Phi(x)$ for $y \in \{-1, 1\}$.

The corresponding function used in structural SVM governs the conditional probability shown in The equation. However, opposed to Algorithm 1, in this case we need to compute the expected sufficient statistics Σy P(y|x)$\Psi(x, y)$. Moreover, learning CRFs does not allow dual formation, unlike structural SVM, which raises additional computational challenges. For example, the dual formation enables the use of kernel functions which simplify the computation.

The Maximum Margin Markov Networks (M3N) method [9] is also using prior knowledge of probability distributions but instead of likelihood maximization intent to directly solve equation 6. This approach is a different method to the cutting plane algorithm presented here, but its implementation is more limited.

3. Applications

In order to applying a structural SVM to a problem, as expected, we would need to define the following functions:

(8) $$\Psi(x, y),$$

(9) $$\Delta(y, \hat{y}),$$

(10) $$argmax_{\hat{y} \in Y}\{w \cdot \Psi(x_i, \hat{y})\},$$

(11) $$argmax_{\hat{y} \in Y}\{\Delta(y_i, \hat{y}) + w \cdot \Psi(x_i, \hat{y})\}.$$

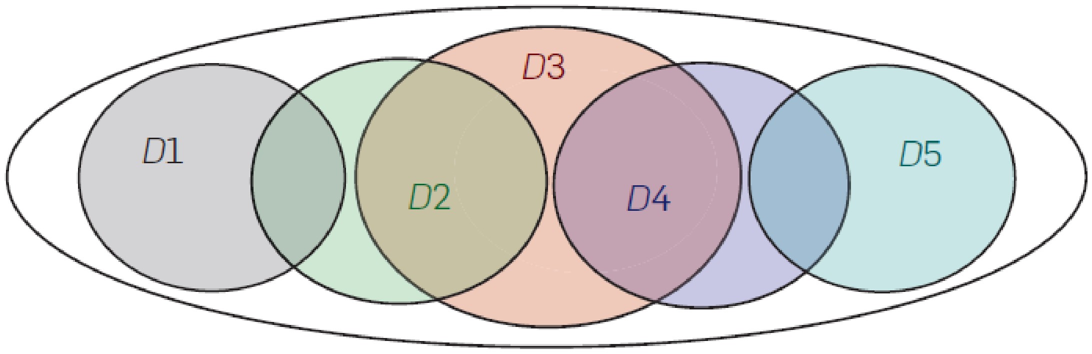
3.1 optimizing diversity in search engines:

Modern predictive models often use machine learning models for ranking function optimization. Although efficient to some degree, traditional machine learning methods give independent scores to each document. Hence, cannot explain the information diversity, clearly required, according to recent studies [10,11]. Those studies highlight the need to model the dependencies within different parts of the document, which is basically a structural prediction problem. To this end, we implemented a structural SVM model to model how to diversify, given a dataset of queries and information diversity labels for each document [12].

For a query in this case of (x, y) learning problems, let the applicant documents required for the model ranking denoted by $x = \{x_1, ..., x_n\}$. The labeling (targets) for x is set of subtopics $T = \{T_1, ..., T_n\}$, where T_i denotes the document x_i subtopics. The estimation goal is to predict a subset $y \subset x$ of size K maximizing subtopic estimation, and hence maximizing the information

covered by the model. The loss function $\Delta(T, y)$is defined to be the subtopics weighted portion not covered by y (proportionally weighted to subtopic appearance).
Even if subtopic targets are known, reckoning the optimal y can be a challenging task. As this is a budget maximum problem due to documents overlap within subtopics (i.e., $\exists i, j$: $T_i \cap T_j \neq \emptyset$). Conventional rapacious methods reach estimation bound of $1 - 1/e$, and commonly return sufficient predictions and provide leading documents ranking.
A conceptual representation of this prediction task is shown in figure 3.

Figure 3. Different documents (shaded regions) cover different information (subtopics) of a query. Here, the set {*D*2, *D*3, *D*4} contains the three documents that individually cover the most information, but {*D*1, *D*3, *D*5} collectively covers more information.



The colored regions depict applicant documents x for query, and each region area coverage is the "information" covered by that document, represented as subtopics T. When T is known the rapacious method can be used to a solution with high subtopic diversity. For example, the optimal solution for $K = 3$ is $y = \{D1, D3, D5\}$.

In the general case, though, the subtopics of a new query are unknown. However, the information diversity is represented by the words constituting the document. To optimize subtopic cover we should detect more distinct words, with proportionally weighting more informative words. Following the Essential pages [10] approach, word significance is measured mainly by its proportionate recurrence within x. For an example of joint feature representation Ψ, let $\emptyset(v, x)$ denote feature vector describing word frequency v over documents in x. i.e., we can formulate $\emptyset(v, x)$ as,

$$\phi(\nu, \mathbf{x}) = \begin{pmatrix} 1[\nu \text{ appears in at last 15\% of } \mathbf{x}] \\ 1[\nu \text{ appears in at least 35\% of } \mathbf{x}] \\ \cdots \end{pmatrix}.$$

Let $V(y)$ denote the union of words included in predicted subset y documents. Now we can form Ψ as,

$$\Psi(x, y) = \sum_{v \in V(y)} \emptyset(v, x).$$

In a model vector w, the gain of covering word v in x is $w \cdot \emptyset(v, x)$, and is performed when v is contained in a document in y. Due to word overlapping between documents, the latter is also a budgeted maximum coverage task. We can effectively solve loss-augmented inference (Eq.11) and prediction (Eq. 10) by a standard rapacious method. We can bound the solution's precision even though the most violated constraints are found [13]. Effective implementation needs more refined Ψ [12].

4. Conclusion

When prediction of complex objects is required, using structural SVMs would be a good option since they're flexible, efficient and work with a wide range of potential applications.

Thanks to the universality of the cutting-plane training algorithm, only relatively small API changes are required for any new application, which make it comfortable for implementation. This study lays the path for further research of efficient training of structural SVMs with kernels in an efficient manner, due to slow existing methods [14] and how solving the algorithm with approximations will affect the quality of the method to predict the desired outputs [13].

B. Theoretical Part

In this part we are going to make some modifications in the algorithm from the Joachim et al. [1] and explore the differences between the original algorithm and the modified algorithm. Then, we'll perform an experiment by implementing the algorithms on a python code via google colab platform (attached in annex 1) to evaluate the actual difference between the two methods.

The algorithm that developed in the article [1] is an important modeling tool for a wide range of tasks, but still it is less frequently chosen because of its relatively slow training speed. Due to this fact, we'll try to make few modifications that may speed up the training. The first modification is using stochastic subgradient descent. The training algorithm would be:

Algorithm 2 Stochastic Subgradient Descent Struct-SVM

```
0.  Input: $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big), \Psi, \Delta, C$ (as described in the first part)
and $\eta_t$ $for\ j = 1, \ldots, T\ to\ be\ the\ stepsize\ in\ each\ iteration$
1.  $w = \vec{0}$
2.  for t =1,…,T :
3.      pick random training pair $(x_k, y_k)$
4.      $y' = argmax_{y \in Y}\ [\Delta(y_k, y) + < w, \Psi(x_k, y) >]$
5.      $w = w - \eta_t \left(w - \frac{C}{n}\big(\Psi(x_k, y') - \Psi(x_k, y_k)\big)\right)$
```

As presented in algorithm 2, for each iteration, we'll pick a random pair, then compute the argmax as detailed on line 4. Finally, we update as required (while η_t can be chosen by several methods). The output would be the prediction function $f(x) = argmax_{y \in Y} < w, \Psi(x, y) >$, while each update of w takes only 1 update of the argmax-prediction.

In addition to using the Subgradient Descent we can solve a Struct-SVM like a non-linear SVM by computing the lagrangian dual problem. A big motivation for that was taking from Balamurugan et al. [15], where they proposed a sequential dual method for structural SVMs that optimizing the dual variables associated with it. The proposed method is fast and work well. When working with the dual problem, we try to maximize instead of minimizing and the problem. The new definition of the problem would be (using lagrangian):

$$\boxed{\max_{\alpha \in \mathbb{R}_+^n} \sum_{\substack{i=1 \\ y \in Y}}^{n} \alpha_{iy} \Delta(y_i, y) - \frac{1}{2} \sum_{\substack{y, \hat{y} \in Y \\ i, \hat{\imath} = 1}}^{n (for\ both\ i\ , \hat{\imath})} \alpha_{iy} \alpha_{\hat{\imath}\hat{y}} < \delta\Psi(x_i, y_i, y), \delta\Psi(x_{\hat{\imath}}, y_{\hat{\imath}}, \hat{y}) > \quad s.t\ \Sigma_{y \in Y} \alpha_{iy} \leq \frac{C}{n}}$$

Then, we could use the kernel trick.

Let us define a function $F: (X \times Y) \times (X \times Y) \to \mathbb{R}$ that measures the similarity between pairs.

$$F\big((x, y), (x, y)\big) = < \Psi(x, y), \Psi(\hat{x}, \hat{y}) >$$

Remembering the expression from above, we'll get:

$$< \delta\Psi(x_i, y_i, y), \delta\Psi(x_{\hat{\imath}}, y_{\hat{\imath}}, \hat{y}) > = < \Psi(x_i, y_i) - \Psi(x_i, y), \Psi\big(x_{\hat{\imath},} y_{\hat{\imath}}\big) - \Psi(x_{\hat{\imath}}, \hat{y}) > =$$

$$< \Psi(x_i, y_i), \Psi\big(x_{\hat{\imath},} y_{\hat{\imath}}\big) > - < \Psi(x_i, y_i), \Psi(x_{\hat{\imath}}, \hat{y}) > - < \Psi(x_i, y), \Psi\big(x_{\hat{\imath},} y_{\hat{\imath}}\big) > + < \Psi(x_i, y), \Psi(x_{\hat{\imath}}, \hat{y}) > =$$

$$F\left((x_i, y_i), (x_{\hat{\imath},} y_{\hat{\imath}})\right) - F\big((x_i, y_i), (x_{\hat{\imath}}, \hat{y})\big) - F\left((x_i, y), (x_{\hat{\imath},} y_{\hat{\imath}})\right) + F\big((x_i, y), (x_{\hat{\imath}}, \hat{y})\big) =: F_{i\hat{\imath}y\hat{y}}$$

Now, we can rewrite the problem to be a Kernelized Struct-SVM with F:

$$\max_{\alpha \in \mathbb{R}_+^{n|Y|}} \sum_{\substack{i=1 \\ y \in Y}}^{n} \alpha_{iy} \Delta(y_i, y) - \frac{1}{2} \sum_{\substack{y, \hat{y} \in Y \\ i, \hat{\imath}=1}}^{n(for\ both\ i\ , \hat{\imath})} \alpha_{iy} \alpha_{\hat{\imath}\hat{y}} F_{i\hat{\imath}y\hat{y}} \qquad s.t\ \Sigma_{y \in Y} \alpha_{iy} \leq \frac{C}{n}$$

The kernelized prediction function is

$$f(x) = argmax_{y \in Y} \sum_{iy'} \alpha_{iy'} F(\,(x_i, y_i), (x, y))$$

By performing the kernel trick, we optimize the speed of the calculation of the dual problem.

The experiment

The data set we chose is "scene" – a structured dataset of pystruct (attached as an annex to submission files in 'scene.pickle'). This dataset is used for multiclass classification problems and consist of labeled image.

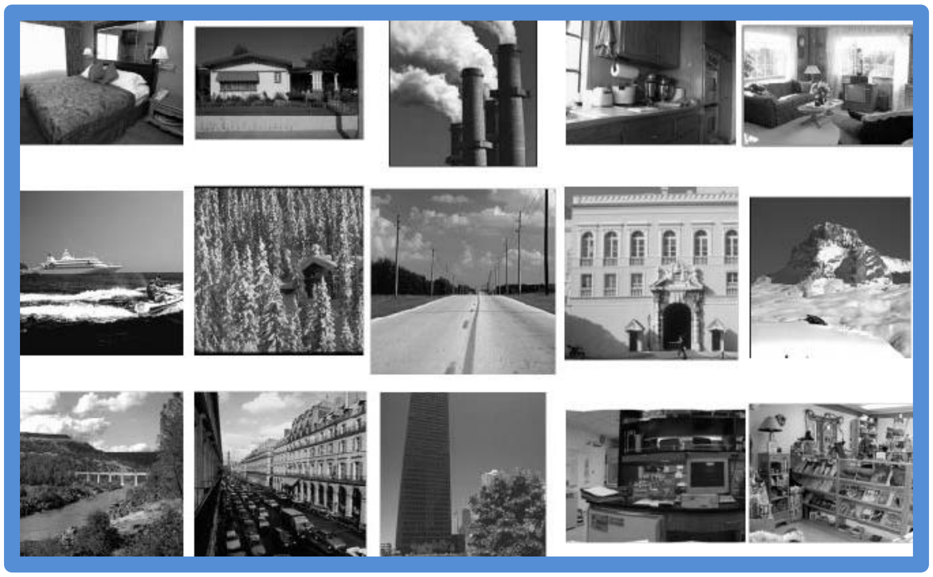


Figure 4: representative images of "scene" dataset

The label describes the general scene of the input image, such as "sunset", "beach" etc. (15 labels in total) as presented in figure 4. An image can be explained in several ways so accordingly, a picture could be classified to several classes (labels).

In the attached code we implemented and compared the two methods – Joachim et al. algorithm 1 and the modified algorithm 2 detailed above on the scene dataset (using public code as referenced in the notebook). The results of the experiment, as shown in the notebook, are accuracy scores of 0.866 and 0.789 of the original and modified algorithms respectively, Meaning the modified model did not perform better than the original one, and is not good enough to replace the original model.

This result may be caused by the dataset that we've chose and may yield different results when tested on an alternative dataset, as finding to which datasets this model will fit for prediction is beyond the scope of this study. Another possible explanation might be the use of the kernel function. Choosing a suitable kernel function the problem is a critical part of the solution and may change the accuracy results of the model. Hence, obtaining an optimal kernel function is a complicated task and a scope for a standalone study for a given dataset, since it is mainly a trial an error exploratory problem. Therefore, we propose further research for implementation of algorithm 2 on various datasets to evaluate and validate its accuracy. Then, when the proposed model is cross validated and as expected will achieve better results than the former one, additional exploration for a preferred kernel function can be performed to yield the maximal accuracy scores.

References

1. Joachims, Thorsten & Hofmann, Thomas & Yue, Yisong & Yu, Chun-Nam. (2009). Predicting Structured Objects with Support Vector Machines. Commun. ACM. 52. 97-104. 10.1145/1592761.1592783.
2. Joachims,T., Finley, T., Yu, C.-N. Cutting-plane training of structural svms. Machine Learning Journal (2009) DOI 10.1007/S 10994-009- 5108–8.
3. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y. Support vector machine learning for interdependent and structured output spaces. In International Conference on Machine Learning (ICML) (2004), 104–112.
4. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y. Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. (JMLR), 6 (September 2005), 1453–1484.
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 86, 11 (1998), 2278–2324, November.
6. Lafferty, J., McCallum, A., Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In International Conference on Machine Learning (ICML) (2001).

7. McCallum, A., Freitag, D., Pereira, F. Maximum entropy Markov modelsfor information extraction and segmentation. In International Conference on Machine Learning (ICML) (2000), 591–598.
8. Crammer, K., Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. (JMLR) 2 (2001), 265–292.
9. Taskar, B., Guestrin, C., Koller, D. Maximum-margin markov networks. In Advances in Neural Information Processing Systems (NIPS) (2003).
10. Swaminathan, A., Mathew, C., Kirovski, D. Essential pages. Technical Report MSR-TR-2008 015, Microsoft Research (2008).
11. Zhai, C., Cohen, W.W., Lafferty, J. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR) (2003).
12. Yue, Y., Joachims, T. Predicting diverse subsets using structural SVMs. In International Conference on Machine Learning (ICML) (2008), 271–278.
13. Finley, T., Joachims, T. Training structural SVMs when exact inference is intractable. In International Conference on Machine Learning (ICML) (2008), 304–311.
14. Yu, C.-N.J., Joachims, T. Training structural svms with kernels using sampled cuts. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2008), 794–802.
15. Shevade, Shirish & Balamurugan, P. & Sellamanickam, Sundararajan & Keerthi, S.. (2011). A Sequential Dual Method for Structural SVMs. 223-234. 10.1137/1.9781611972818.20.