Discriminative Structured Outputs Prediction Model and Its Efficient Online Learning Algorithm

Yang Wu, Zejian Yuan, Yuanliu Liu, Nanning Zheng
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
28 West Xianning Road, Xi'an 710049 Shaanxi, P.R.China
{ywu,zjyuan}@aiar.xjtu.edu.cn, liuyuanliu@stu.xjtu.edu.cn, nnzheng@mail.xjtu.edu.cn

Abstract

There are two big issues emerging in the field of computer vision: one is the explosively increasing large amount of visual data and the other is the demand of deep labeling of objects and scenes. In this paper, we propose a structured outputs prediction framework equipped with a discriminative model and a corresponding efficient online learning algorithm. Instead of doing simple multiclass classification as usual, we aim at outputting structured labels which means different label confusion mistakes may have different costs. Moreover, the online learning algorithm with efficient updating strategy and compact memory management mechanism makes the framework work well on large visual data. Experiments on two representative datasets show an exemplar application of our model.

1. Introduction

As the cameras get cheaper and cheaper, and the storage and computation resources become much more affordable than before, people pay more and more attention to taking pictures and videos, for recording their live experiences or just for fan. The high speed Internet connection makes sharing pictures and videos with other people all around the world possible and fast. Especially in the recent few years, various services provided by the industry greatly stimulate the development of visual data sharing. Therefore, we can get tremendous visual data from the Internet. However, we are not good at handling them intelligently. There are many academic problems to be solved. Among which, scalable online learning algorithms and deep recognition with a large amount of data are two important ones.

When a large dataset is used for training or the data naturally comes in an online way, online learning seems to be a necessary choice. Take the max-margin based discriminative model for example, computing the full kernel matrix for all the training data is impractical. Ten years ago, Platt [9] introduced the famous sequential minimal optimization (SMO) algorithm to train the SVM classifier. It greatly improved the speed of SVM and made the algorithm scalable to large datasets. However, the actual data sometimes is given sequentially, which requires an efficient online learning treatment. Instead of remembering all the given examples gradually, focusing on the supporting patterns only is a plausible solution. Some early work simply updates the whole model when a new example comes, and then selects the supporting patterns under this updated model [7]. Undoubtedly, this strategy needs a lot of computation and pays much on memory accessing. Grammer et al. [4, 3] proposed a multiclass online learning algorithm based on the multiclass Perceptron updating strategy. More recently, Bordes et al. [1] combined the SMO algorithm with the perceptron updating strategy, and presented an adaptive schedule to automatically balance the model updating induced by the new pattern and that coming from the updating of formerly seen support vectors. Their so-called "**LaRank**" approach runs much faster than the state-of-the-art online and batch algorithms on multiclass classification while providing comparable results to them. We follow the routine of this work on online learning, but using it to solve a more general and complex structured output prediction problem.

On the side of deep recognition, a typical case is predicting structured outputs. Instead of outputting a single class label which might be too coarse for understanding an object or even a scene, we are aiming at predicting a label vector which represents structural information of the object/scene. For example, an image of a Toyota car can output several information corresponding to different conceptual levels, such as "man-made tools", "vehicle", "car", "Toyota car" and so on. These labels represents a state of a structured output space. Different pairs of labels may have different distances, due to where the difference lies in the space and how you measure it. Predicting structured outputs is more difficult than traditional multiclass classification, but it can result in deeper understanding of the data and richer applications of many types.

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

A naive strategy to handle this problem is the so called "divide-and-conquer" philosophy, which divides the structured output into several subproblems, and learn separate models for each of them. Accordingly, the data is grouped into subclasses for these subproblems [6, 5]. This approach has the problems of training on very few samples, limited performance and high overall computational cost. Recently, machine learning society has made remarkable progress on directly optimizing structured output prediction problems. Taskar et al. [12, 11] proposed max-margin models for structured prediction problems. Tsochantaridis et al.[13] introduced loss functions into the max-margin model and proposed an optimization method. Rousu et al. [10] presented an effective solution to the tree-structured output predicting problem using kernels. Most of their work was applied to the problem of text classification for its simplicity.

In this paper we combine the work on both sides and propose a general structured outputs prediction framework with an efficient online learning algorithm. We are aiming at applying it to large visual data for the task of deep recognition or labeling. In this paper, we present some primary results on two well-known handwritten digit datasets to demonstrate the effectiveness and efficiency of our approach.

The rest of this paper is organized as follows. First, the problem of structured outputs prediction and our model for it are represented in details in section 2. Second, an efficient and effective online learning algorithm is introduced to do the optimization. Then we present various experimental results on two datasets to demonstrate the advantages of the proposed framework. Finally, some discusses on possible extensions are given.

2. Structured Outputs Prediction

Unlike the traditional concept on recognition which either refer to the one-class detection or multi-class categorization, we focus on the problem of "structured outputs prediction", which outputs hierarchically structured labels with multiple acceptable categorization levels. More generally, the outputs can have any kind of structure, either trees or graphs, with symmetric or dissymmetric distances among the possible outputs. Usually, the learning algorithm for achieving this goal demands complex computation and large memory support. Inspired by recent literatures on machine learning, we are able to get an efficient online learning algorithm for it.

2.1. Problem Formulation

Suppose an arbitrary input pattern $\mathbf{x} \in \mathbf{X}$ and its label vector $\mathbf{y} \in \mathbf{Y}$ have a compatibility denoted by $E(\mathbf{x}, \mathbf{y}) \in \mathcal{R}$, then the output prediction of $\mathbf{x}$ can be defined by:

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathbf{Y}} \{E(\mathbf{x}, \mathbf{y})\}. \quad (1)$$

We can define a joint feature space by $\Phi(\mathbf{x}, \mathbf{y}) \in \mathcal{F}$, and a distance measure in an inner product space to specify the compatibility function:

$$E(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle, \quad (2)$$

where $\mathbf{w}$ is the prototype vector or the projection vector in the joint feature space.

An alternative formulation can be inferred by solving the maximum entropy problem with joint feature representation [15]. Then it results in a MAP problem $f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathbf{Y}} \{\mathbf{p}(\mathbf{y}|\mathbf{x})\}$, where,

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}. \quad (3)$$

Here the denominator $Z(\mathbf{w}, \mathbf{x})$ is an integration over $\mathbf{Y}$, which is usually computational expensive or even unbounded. Comparatively, the formulation of equation (1) directly optimize the output prediction problem without estimating the statistical probabilities. Therefore, we adopt it as our problem formulation.

Note that the output here is a label vector, different dimensions of which can stand for different conceptual meanings. For example, it may represent a state of a tree structure as shown in Figure 1. In this case, $\mathbf{y} = (y_1, y_2, \cdots, y_m)^T$, where $y_j \in \{0, 1\}, j = 1, 2, \cdots, m$ stands for a vertex of the tree, and an instance of $\mathbf{y}$ may be restrained to be an indication of the shortest path from a vertex in this tree to the root.

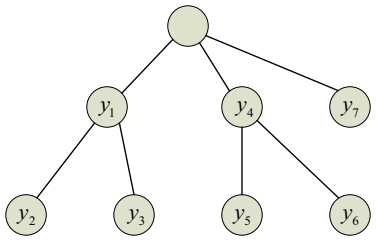


Figure 1: An example of the structured output.

2.2. Loss-sensitive Max-Margin Learning Model

Consider training examples $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$. For each pattern $\mathbf{x}_i \in \mathbf{X}$, we expect that the compatibility $E(\mathbf{x}_i, \mathbf{y}_i)$ between it and its actual label vector $\mathbf{y}_i \in \mathbf{Y}$ is greater than any other $E(\mathbf{x}_i, \mathbf{y}), \mathbf{y} \neq \mathbf{y}_i$. Following the max-margin formulation, we can easily get a constrained geometrical margin maximizing problem.

However, since in our case $\mathbf{y}$ is a structured output and different $\mathbf{y}$ may have different distances. Therefore, mistaking different pairs of them should have different costs. Aware of this, we can assign different margin demands to different output pairs using a loss function $l(\mathbf{y}_i, \mathbf{y}_j)$. Then, the max-margin model can be written as:

$$\begin{aligned} &\min_{\mathbf{w}} \left\{ \tfrac{1}{2}\|\mathbf{w}\|_{\mathbf{2}}^{\mathbf{2}} + \mathbf{C1}^{\mathbf{T}}\xi \right\} \\ &s.t. \quad \left\{ \begin{array}{ll} \langle \mathbf{w}, \Delta\Phi(\mathbf{x}_i, \mathbf{y}) \rangle \geq l(\mathbf{y}_i, \mathbf{y}) - \xi_i & , \quad \forall i,\ \mathbf{y} \neq \mathbf{y}_i \\ \xi_i \geq 0 & , \quad \forall i \end{array} \right. \end{aligned}$$

where $\Delta\Phi(\mathbf{x}_i, \mathbf{y})$ stands for $\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y})$. The constraint with the lost function is the so-called partial ranking. Different values of the lost function indicate different demands on the margins of partial ranking.

The dual program of this formation can be derived as:

$$\begin{aligned} &\max_{\alpha} \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_i(\mathbf{y})\, l(\mathbf{y}_i, \mathbf{y}) \ - \\ &\quad \tfrac{1}{2} \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \sum_{j, \bar{\mathbf{y}} \neq \mathbf{y}_j} \alpha_i(\mathbf{y})\, \alpha_j(\bar{\mathbf{y}}) \langle \Delta\Phi(\mathbf{x}_i, \mathbf{y}), \Delta\Phi(\mathbf{x}_j, \bar{\mathbf{y}}) \rangle \\ &s.t. \quad \left\{ \begin{array}{l} \alpha_i(\mathbf{y}) \geq 0 \quad , \quad \forall i, \mathbf{y} \neq \mathbf{y}_i \\ \sum\limits_{\mathbf{y} \neq \mathbf{y}_i} \alpha_i(\mathbf{y}) \leq C \quad , \quad \forall i \end{array} \right. \end{aligned}$$

From the constraints of the primal problem, we know that the dual variable α has a dimension of $|\mathbf{X}| \times (|\mathbf{Y}| - 1)$. Suppose the optimal solution of the dual problem is α, then the prediction function is

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathbf{Y}} \sum_{i, \bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i^*(\bar{\mathbf{y}}) \langle \Delta\Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}, \mathbf{y}) \rangle.$$

2.3. Reparametrizing for Model Compaction

Inspired by the reparametrizing strategy proposed in **LaRank**, the above dual problem can be simplified by replacing α with an $|\mathbf{X}| \times |\mathbf{Y}|$ dimensional variable γ:

$$\gamma_i(\mathbf{y}) = \left\{ \begin{array}{ll} -\alpha_i(\mathbf{y}), & \mathbf{y} \neq \mathbf{y}_i \\ \sum\limits_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) & \mathbf{y} = \mathbf{y}_i \end{array} \right.$$

It can be seen that the free dimension of γ is still $|\mathbf{X}| \times (|\mathbf{Y}| - 1)$, for it satisfies the constraint:

$$\sum_{\mathbf{y}} \gamma_i(\mathbf{y}) = 0 \quad , \quad \forall i.$$

By defining $l(\mathbf{y}_i, \mathbf{y}_i) = 0$, it can be derived that the dual problem equals to the following problem (see Appendix 1):

$$\begin{aligned} &\max_{\gamma} \left\{ \begin{array}{l} -\sum\limits_{i, \mathbf{y}} \gamma_i(\mathbf{y})\, l(\mathbf{y}_i, \mathbf{y}) \\ \quad -\tfrac{1}{2} \sum\limits_{i, \mathbf{y}} \sum\limits_{j, \bar{\mathbf{y}}} \gamma_i(\mathbf{y})\, \gamma_j(\bar{\mathbf{y}}) \langle \Phi(\mathbf{x}_i, \mathbf{y}), \Phi(\mathbf{x}_j, \bar{\mathbf{y}}) \rangle \end{array} \right\} \\ &s.t \quad \left\{ \begin{array}{l} \gamma_i(\mathbf{y}) \leq 0, \ \ \forall i, \mathbf{y} \neq \mathbf{y}_i \\ \gamma_i(\mathbf{y}_i) \leq C, \forall i \\ \sum\limits_{\mathbf{y}} \gamma_i(\mathbf{y}) = 0, \forall i \end{array} \right. \end{aligned} \tag{4}$$

The prediction function with the optimal γ^* becomes

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathbf{Y}} \sum_{i, \bar{\mathbf{y}}} \gamma_i^*(\bar{\mathbf{y}}) \langle \Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}, \mathbf{y}) \rangle$$

2.4. Loss Function Design

The loss function in the above optimization problem modulates the dual variable γ. It can be viewed as a prior of the output space. By designing proper loss functions, different structured outputs problem can be solved using the presented discriminative model.

For example, in the case of tree structured output predicting problem like the one presented in Figure 1, a loss function might be the difference of two paths on the tree. In Figure 2, we may let $\mathbf{y}_2$ and $\mathbf{y}_7$ indicate the two paths ended at vertex y_2 and y_7 respectively. Then the loss $l(\mathbf{y}_2, \mathbf{y}_7)$ of mixing these two can be represented as the accumulated cost along the shortest path from vertex y_2 to vertex y_7. In practice, any cost can be defined on each edge along the path, according to different application backgrounds.

Generally, any lost function can be adopted, as long as it represents the structural information of the output space. It does not need to be symmetric as in the tree structure case. Dissymmetry can represent directional relationships among different outputs.

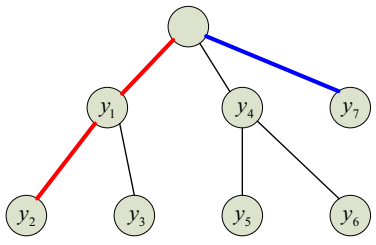


Figure 2: Loss function of a tree structure.

2.5. Kernel Factorization

In the dual optimization problem of equation (4), there is an inner production which operates in the joint feature

space of both the patterns and their labels. By designing or choosing proper kernel function, we can avoid explicit joint feature mapping and inner product computation in the high dimensional feature space. The joint kernel can be defined as:

$$K\left(\mathbf{x}_i, \mathbf{y}, \mathbf{x}_j, \bar{\mathbf{y}}\right) = \left\langle \Phi\left(\mathbf{x}_i, \mathbf{y}\right), \Phi\left(\mathbf{x}_j, \bar{\mathbf{y}}\right)\right\rangle.$$

A critical issue is the specification of the joint kernel for capturing the intrinsic similarity between the pair of examples while simplifying the computation. An common strategy is to factorize the joint kernel into two parts: the input kernel and the output kernel.

$$K\left(\mathbf{x}_i, \mathbf{y}, \mathbf{x}_j, \bar{\mathbf{y}}\right) = K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) K_{\mathbf{y}}\left(\mathbf{y}, \bar{\mathbf{y}}\right)$$

Suppose φ, ϕ are the feature mappings of the input pattern and the output label respectively, then the two kernels can be viewed as inner productions in their corresponding feature spaces:

$$\begin{array}{l} K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \varphi\left(\mathbf{x}_i\right), \varphi\left(\mathbf{x}_j\right)\right\rangle \\ K_{\mathbf{y}}(\mathbf{y}, \bar{\mathbf{y}}) = \left\langle \phi\left(\mathbf{y}\right), \phi\left(\bar{\mathbf{y}}\right)\right\rangle. \end{array}$$

Moreover, the joint kernel can be represented as an inner product in the tensor feature space of $\mathcal{H}_\varphi \otimes \mathcal{H}_\phi$:

$$\begin{array}{l} K\left(\mathbf{x}_i, \mathbf{y}, \mathbf{x}_j, \bar{\mathbf{y}}\right) = \left\langle \varphi\left(\mathbf{x}_i\right), \varphi\left(\mathbf{x}_j\right)\right\rangle \left\langle \phi\left(\mathbf{y}\right), \phi\left(\bar{\mathbf{y}}\right)\right\rangle \\ = \left\langle \varphi\left(\mathbf{x}_i\right) \otimes \phi\left(\mathbf{y}\right), \varphi\left(\mathbf{x}_j\right) \otimes \phi\left(\bar{\mathbf{y}}\right)\right\rangle \end{array}$$

As it can be seen, the computation cost of the joint kernel depends on the volume of the input space and that of the output space, more specifically $|K| = |\mathbf{X}|^2 \times |\mathbf{Y}|^2$. In the case of structured output problem, $|\mathbf{Y}|$ might be large. To reduce the kernel storage expense and the computational cost, we must make the output kernel as sparse as possible.

The output kernel is a measurement of the similarity between different output labels, so it can be used to capture the structure of the output space. Fortunately, we already have the loss function $l(\mathbf{y}, \bar{\mathbf{y}})$ to do the task as mentioned before, so here we can make the output kernel as simple as possible. The simplest choice is the identity matrix, namely,

$$K_{\mathbf{y}}\left(\mathbf{y}, \bar{\mathbf{y}}\right) = \left\{ \begin{array}{ll} 1 & \mathbf{y} = \bar{\mathbf{y}} \\ 0 & \mathbf{y} \neq \bar{\mathbf{y}} \end{array} \right.$$

Then the joint kernel is simplified into

$$K\left(\mathbf{x}_i, \mathbf{y}, \mathbf{x}_j, \bar{\mathbf{y}}\right) = \left\{ \begin{array}{cc} K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) & \mathbf{y} = \bar{\mathbf{y}} \\ 0 & \mathbf{y} \neq \bar{\mathbf{y}} \end{array} \right.$$

Its volume is just that of the input kernel. And following this, the optimization problem of (4) becomes

$$\begin{array}{l} \max\limits_{\gamma} \left\{ -\sum\limits_{i,\mathbf{y}} \gamma_i\left(\mathbf{y}\right) l\left(\mathbf{y}_i, \mathbf{y}\right) \right. \\ \left. -\frac{1}{2} \sum\limits_{i,j} \sum\limits_{\mathbf{y}} \gamma_i\left(\mathbf{y}\right) \gamma_j\left(\mathbf{y}\right) K_{\mathbf{x}}\left(\mathbf{x}_i, \mathbf{x}_j\right) \right\} \\ s.t \quad \left\{ \begin{array}{l} \gamma_i\left(\mathbf{y}\right) \leq 0, \quad \forall i, \mathbf{y} \neq y_i \\ \gamma_i\left(\mathbf{y}_i\right) \leq C \ , \quad \forall i \\ \sum\limits_{\mathbf{y}} \gamma_i\left(\mathbf{y}\right) = 0 \ , \quad \forall i \end{array} \right. \end{array} \tag{5}$$

Once the optimal γ^* is obtained, the output prediction function is as simple as

$$f\left(\mathbf{x}\right) = \arg\max_{\mathbf{y}} \sum_i \gamma_i^*\left(\mathbf{y}\right) K_{\mathbf{x}}\left(\mathbf{x}_i, \mathbf{x}\right). \tag{6}$$

3. Online Learning and Inference

In the former section we factorized the joint kernel and simplified the output kernel to reduce the computational cost as much as possible. However, the input kernel $K_{\mathbf{x}}\left(\mathbf{x}_i, \mathbf{x}_j\right)$ is not easy to be simplified. An possible solution is online learning, which deals with the input patterns sequentially and updates only a part of the model when a new pattern is processed. Furthermore, we can accelerate both the learning and the prediction by fast inference in the output space, when the structure of the space can be broken down into local connections. This section will discuss these two issues.

3.1. Online Learning Algorithm

The most critical part of online learning

The most critical part of an efficient discriminative online learning algorithm is to choose the most effective patterns and update their most promising coefficients. When a new pattern comes, and the current model cannot correctly predict its label, then some of its coefficients surely need to be updated. Besides of that, the current support vectors may also need to be updated by adjusting some of their weights (coefficients). About when and how to update the existing support vectors, we follow the adaptive selection schedule of **LaRank**.

Maximum gradient criterion

Once the pattern to be updated is fixed, it comes to the problem of choosing promising outputs whose corresponding coefficients have the highest gradients on the dual concave optimization function. Therefore, we compute the derivatives $g_i(\mathbf{y})$ of the dual objective function with respect to the coefficients $\gamma_i(\mathbf{y})$ on the chosen pattern $\mathbf{x}_i$.

$$\begin{array}{l} g_i\left(\mathbf{y}\right) = \frac{\partial J(\gamma)}{\partial \gamma_i(\mathbf{y})} \\ = -l\left(\mathbf{y}_i, \mathbf{y}\right) - \sum\limits_j \gamma_j\left(\mathbf{y}\right) K_{\mathbf{x}}\left(\mathbf{x}_i, \mathbf{x}_j\right) \end{array} \tag{7}$$

Two updating pairs with extreme gradients

As usual, we choose to update only two most promising pairs $(\mathbf{x}_i, \mathbf{y}^+)$ and $(\mathbf{x}_i, \mathbf{y}^-)$ on pattern $\mathbf{x}_i$. In which $\mathbf{y}^+$ denotes the one with the greatest gradient while $\mathbf{y}^-$ stands for the opposite, the smallest gradient. To describe the updating procedure, we denote that the current model is the result of the k_{th} step, and we are approaching step $k+1$. According to equation 7, the current two gradients are:

$$\begin{aligned} g_i^k(\mathbf{y}^+) &= -l(\mathbf{y}_i, \mathbf{y}^+) - \sum_j \gamma_j^k(\mathbf{y}^+) K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) \\ g_i^k(\mathbf{y}^-) &= -l(\mathbf{y}_i, \mathbf{y}^-) - \sum_j \gamma_j^k(\mathbf{y}^-) K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{8}$$

Updating method (Perceptrons)

Since the object function of (4) is concave, to approach to the maximum point, $\gamma_i(\mathbf{y}^+)$ has to be increased while $\gamma_i(\mathbf{y}^-)$ has to be decreased. Considering the constraint $\sum_{\mathbf{y}} \gamma_i(\mathbf{y}) = 0$, a single coefficient increment variable $\lambda \geq 0$ can be used for the updating:

$$\begin{aligned} \gamma_i^{k+1}(\mathbf{y}^+) &= \gamma_i^k(\mathbf{y}^+) + \lambda \\ \gamma_i^{k+1}(\mathbf{y}^-) &= \gamma_i^k(\mathbf{y}^-) - \lambda \end{aligned} \tag{9}$$

Optimize updating step

Feed equation (9) into the objective function (4) and derive the best unconstrained value of λ as (see Appendix 2):

$$\lambda^* = \frac{g_i^k(\mathbf{y}^+) - g_i^k(\mathbf{y}^-)}{2K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i)}$$

Considering the constraints on $\gamma_i(\mathbf{y})$, we can get the final expression for λ:

$$\lambda = \max\left(0, \min\left(\lambda^*, C\delta(\mathbf{y}^+, y_i) - \gamma_i^k(\mathbf{y}^+)\right)\right).$$

Update related gradients

According to the gradient expression in (7), for outputs $\mathbf{y}^+$ and $\mathbf{y}^-$, once an arbitrary pattern changes coefficients on them, the gradients of this pattern on $\mathbf{y}^+$ and $\mathbf{y}^-$ need to be updated:

$$\begin{aligned} g_j^{k+1}(\mathbf{y}^+) &= g_j^k(\mathbf{y}^+) - \lambda K_{\mathbf{x}}(\mathbf{x}_j, \mathbf{x}_i) \quad , \quad \forall j, (\mathbf{x}_j, \mathbf{y}^+) \in S \\ g_j^{k+1}(\mathbf{y}^-) &= g_j^k(\mathbf{y}^-) + \lambda K_{\mathbf{x}}(\mathbf{x}_j, \mathbf{x}_i) \quad , \quad \forall j, (\mathbf{x}_j, \mathbf{y}^-) \in S \end{aligned}$$

where S denote the set of support vectors.

A sequential minimal optimization (SMO) algorithm for each updating step, can be constructed. It is not listed here due to the space, and one can refer to that of **LaRank** for details.

3.2. Rapid Inference Algorithm

In the above online learning algorithm, at each step, two most promising pairs $(\mathbf{x}_i, \mathbf{y}^+)$ and $(\mathbf{x}_i, \mathbf{y}^-)$ on pattern $\mathbf{x}_i$ need to be found. Here $\mathbf{y}^+$ and $\mathbf{y}^-$ are two points in the structured output space (or the supporting subspace as mentioned in **LaRank**). When the space is small, we can just test each possible value and choose the ones we need in a brute force way. However, usually the output space is exponentially large respect to the dimension of $\mathbf{y}$. So a effect inference algorithm is needed. This has to do with the structure of the output space. Local connections can be used to speed up the inference. For example, when the output space is tree-structured, we can redefine the lost function as a summarization of loses on the edges (then the dual variable is also defined on the edges), and use dynamic programming to do the inference as in [10], or even use a pruned dynamic programming algorithm which is suboptimal but faster as proposed in [2].

4. Experiments

We report some primary results on two handwritten digit recognition datasets. Although the proposed model and its online learning algorithm work on problems with strong structures in the output space, this paper focuses on a more general problem in which the loss function can be arbitrarily defined. The experiments presented here are not demonstrating explicit structured output prediction, but one can see that it can work on those problems by only changing the loss function, making it reflect the output structure.

4.1. Datasets

Aiming at proposing the model and the concept behind it, we chose not to involve too many techniques irreverent to our approach on handling complex visual tasks like multilabel object recognition or scene recognition. Therefore, we did our experiments on the classical problem of handwritten digit recognition, with two well-known datasets: USPS and MNIST. The former one contains normalized digits (16 x 16 grayscale each) automatically scanned from envelopes by the U.S. Postal Service, while the later is a mixed subset of NIST's two special databases on handwritten digits. Details of these two datasets can be find easily on the Internet, so here we just list the information related to our experiments, as shown in Table 1. Note that for comparison with **LaRank**, we used the same parameters and kernels as it.

4.2. Loss Function

As far as we are aware, almost all of the others who worked on the handwritten digits recognition treated it as a simple flat multiclass problem. However, is it really flat? Are the classes equally difficult? The answer is no. We investigated into the recognition results provided by dozens

Table 1: Datasets and the most important parameters.

DataSet	Train Ex.	Test Ex.	Features	C	Kernel
USPS	7291	2007	256	10	$e^{-0.025\|\mathbf{x}-\bar{\mathbf{x}}\|^2}$
MNIST	60000	10000	780	1000	$e^{-0.02\|\mathbf{x}-\bar{\mathbf{x}}\|^2}$

of people, and we had never seen a flat confusion matrix (without the diagonal). Some of the digits are easier to be misclassified than the others, such as '4', '9', '2', '7', and so on. And some digit may be easy to be classified into another one while the opposite is unusual, for example '3' sometimes looks like an '8', while '8' is unlikely been recognized as a '3'. Figure 3 shows some misclassification examples from the literature.

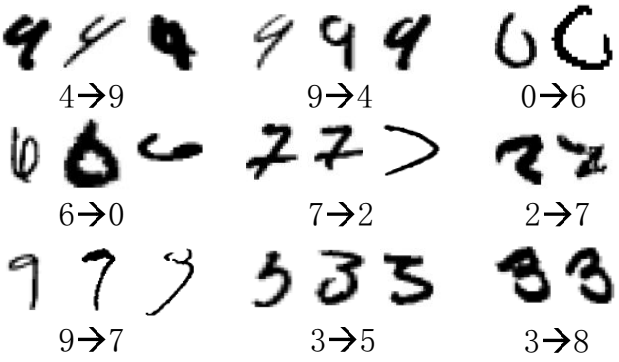


Figure 3: Some typical mistakes on digits recognition.

Since some of the classes are easier to be misclassified than the others, like '4' and '9', we shouldn't demand as much from them as from the others. If the algorithm pays much on separating '4' and '9', it may not perform well on the classes that can easily be classified. Therefore, some tolerance should be given to the confusion of '4' and '9' while being strict on mixing '4'and '3'. Inspired by this idea, we looked for a proper loss function or loss matrix that characterize the relationship among the 10 handwritten digit classes. However, it is not an easy task to do this. Finally, we adopted a simple strategy: averaging the confusion matrixes reported in the literature on the same datasets and deriving loss functions from them. From [14] we got a confusion matrix on the whole dataset of MNIST, while [8] provided us three different confusion matrixes on the same subset of NIST. Since the larger the misclassification error is, the smaller its corresponding loss should be, we simply chose the exponential function to represent this mapping:

$$l(\mathbf{y}_i, \mathbf{y}_j) = e^{-\rho \cdot \mathbf{err}(\mathbf{y}_i, \mathbf{y}_j)} \tag{10}$$

where $\mathbf{err}(\mathbf{y}_i, \mathbf{y}_j)$ is the misclassification error of the pair $(\mathbf{y}_i, \mathbf{y}_j)$ presented in the confusion matrix, and ρ is a parameter controlling the variation of the loss. To be easier to understand, we use a new parameter

$$\eta = \frac{l_{min}}{l_{max}} = \frac{\min\limits_{i,j} l(\mathbf{y}_i, \mathbf{y}_j)}{\max\limits_{i,j} l(\mathbf{y}_i, \mathbf{y}_j)} \tag{11}$$

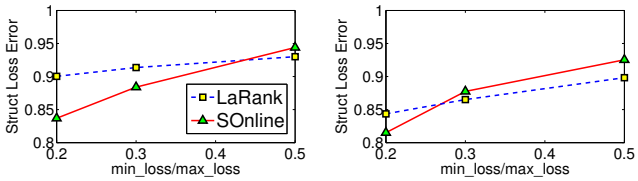
to indicate different settings of the loss function on the same confusion matrix. To represent the different loss function from the MNIST and NIST datasets, we denote them as l_{MNIST} and l_{NIST} respectively. Both of these two loss functions are normalized before usage.

When the loss function (either l_{MNIST} or l_{NIST}) are involved in the learning, the optimization objective is effected by the loss, and accordingly the evaluation of the performance should also takes the loss function into account. As the literature on structured output learning often does, we use the accumulated loss as our new performance measurement, called "structured loss error".

4.3. Experimental Results

Using the two loss functions l_{MNIST} and l_{NIST} and varying the parameter of η, we got a batch of results on the two digit datasets. Figure 4 shows both the results of our algorithm (named "**SOnline**") and those of the most similar algorithm "**LaRank**" to ours which aims directly at multiclass classification. Each subgraph shows the results on a dataset with a specific loss function. Structured loss error of three runs with different η (0.2, 0.3 and 0.5) are shown in each subgraph.

Since **LaRank** has nothing to do with loss function, the learning was done as a pure multiclass classification problem, while the results shown in Figure 4 are evaluated by the structured loss error to be compared with our algorithm. It can be seen that for **LaRank**, the smaller η is, the smaller the structured loss error is. This indicates that the loss functions briefly coincides with the original 0-1 errors of **LaRank**. Our algorithm **SOnline** also has this trend, while only the second point in Figure 4(c) is an exception (may due to the randomness in the online learning algorithm). When η is large, the two algorithms have similar performance, probably due to the fact that the lost functions do not coincide well with the data, some of their values are helpful and some of the others might be harmful. However, **SOnline** performs significantly better than **LaRank** when η is small. It tells us that in these two loss functions the most typical errors got proper loss values, so



(a) USPS using l_{MNIST} (b) USPS using l_{NIST}

(c) MNIST using l_{MNIST} (d) MNIST using l_{NIST}

Figure 4: Results on two datasets using two different lost functions.

Table 2: Compared test error rates and training times.

		USPS	MNIST
MCSVM	0-1 loss error (%)	**4.24**	1.44
	Structured loss	N/A	N/A
	Training time (s)	**60**	25000
	Kernels ($\times 10^6$)	51.2	6908
SVMstruct	0-1 loss error (%)	4.38	1.40
	Structured loss	N/A	N/A
	Training time (s)	6300	265000
	Kernels ($\times 10^6$)	1063.3	158076
LaRank x1 (online)	0-1 loss error (%)	4.29	1.45
	Structured loss	0.87	1.53
	Training time (s)	70	8167
	Kernels ($\times 10^6$)	7.75	**2096**
SOnline (online)	0-1 loss error (%)	4.93	**1.39**
	Structured loss	**0.83**	**1.46**
	Training time (s)	132	8377
	Kernels ($\times 10^6$)	**7.25**	2116

when they are paid special attention to, the goodness of using loss functions comes up. As mentioned before, the proposed framework not only works on this type of loss function, but should also work on loss functions for structured outputs, which is the original motivation of this work.

Table 2 shows the efficiency of **SOnline** compared to **LaRank** and the other two approaches. Note that the data of **MCSVM** and **SVMstruct** are directly cited from [1] since the experimental setups are exactly the same. However, the results of **LaRank** are generated by ourselves using the code provided by the author for the new evaluation on structured loss error. One significant difference from the published data in the original paper of **LaRank** is that we didn't restrict the kernel cache size, so the kernel numbers on MNIST dataset are much larger. Even though, the computational cost (in both space and time) of the online **LaRank** is much better than **MCSVM** and **SVMstruct**, while our **SOnline** is comparable to **LaRank** though we work on a more complex problem of predicting structured outputs with loss functions. As shown in Figure 4, we had run several experiments with different parameters, so here we just put in the average result from the two different lost function with $\eta = 0.2$. At this setting, **SOnline** performs better than **LaRank** using structured loss error while the results measured by 0-1 loss error may be the opposite. That is acceptable, because decreasing the cost of misclassifying those hard classes may result in an increase of the number of misclassified examples.

5. Conclusion and Future Work

In this paper we have presented a framework for discriminatively predicting structured outputs in an online learning fashion. There are four key issues in this framework: 1) loss function, 2) kernel function, 3) online updating strategy, and 4) inference algorithm. The general form of loss function in the model makes it applicable to problems with different output spaces. A kernel-based online learning strategy using various simplification and acceleration techniques ensures the efficiency of the proposed approach while at the same time keeps its performance. However, we haven't gone into the details of how to deal with the inference problem since the data used in the primary experiments does not naturally represents special structures like trees.

Future work could be two folds: one is extending it to multilabel recognition problems with efficient inference strategies in the output space and the other is applying it to other large-scale visual recognition problems.

Acknowledgment

This work was supported by the National Basic Research Program of China under Grant No. 2007CB311005, and the National Natural Science Foundation of China under Grant No. 90820017.

References

[1] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with larank. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 89–96, New York, NY, USA, 2007. ACM.

[2] Y. Chen, L. Zhu, C. Lin, A. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 289–296. MIT Press, Cambridge, MA, 2008.

[3] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, 2003.

[4] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-Taylor, and B. Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2001.

[5] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. pages I: 446–453, 2005.

[6] M. Jones and P. Viola. Fast multi-view face detection. Technical Report 96, Mitsubishi Electric Research Laboratories, 2003.

[7] K. W. Lau and Q. H. Wu. Online training of support vector classifier. *Pattern Recognition*, 36(8):1913–1920, August 2003.

[8] E. G. Miller. *Learning from One Example in Machine Vision by Sharing*. PhD thesis, Massachusetts Institute of Technology, 2002.

[9] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[10] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

[11] B. Taskar. *Learning structured prediction models: A large margin approach*. PhD thesis, Stanford University, CA., 2004.

[12] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks, 2003.

[13] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[14] H.-W. Wu. Handwriting recognition with elementary geometric and algorithmic methods. Master's thesis, University of Chicago, 2006.

[15] C. S. Zhu, N. Y. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling, November 1997.

Appendix 1

For simplification, we use the following abbreviations:

$$\gamma_{i\mathbf{y}} = \gamma_i(\mathbf{y}); l_{i\mathbf{y}} = l(\mathbf{y}_i, \mathbf{y});$$
$$\Phi_{ii} = \Phi(\mathbf{x}_i, \mathbf{y}_i); \Phi_{i\mathbf{y}} = \Phi(\mathbf{x}_i, \mathbf{y}).$$

Others can be similarly defined.

Then, the objective function of equation (4) becomes:

$$\begin{aligned}
&J(\gamma) = -\sum_{i,\mathbf{y}} \gamma_{i\mathbf{y}} l_{i\mathbf{y}} \\
&-\tfrac{1}{2} \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Delta\Phi_{i\mathbf{y}}, \Delta\Phi_{j\bar{\mathbf{y}}} \rangle \\
&= -\sum_{i,\mathbf{y}} \gamma_{i\mathbf{y}} l_{i\mathbf{y}} - \tfrac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{i\mathbf{y}}, \Phi_{j\bar{\mathbf{y}}} \rangle \\
&-\tfrac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \left[\langle \Phi_{ii}, \Phi_{jj} \rangle - \langle \Phi_{ii}, \Phi_{j\bar{\mathbf{y}}} \rangle - \langle \Phi_{i\mathbf{y}}, \Phi_{jj} \rangle \right] \\
&= -\sum_{i,\mathbf{y}} \gamma_{i\mathbf{y}} l_{i\mathbf{y}} - \tfrac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{i\mathbf{y}}, \Phi_{j\bar{\mathbf{y}}} \rangle \\
&-\tfrac{1}{2} \left\{ \sum_{i,j} \langle \Phi_{ii}, \Phi_{jj} \rangle \sum_{\mathbf{y},\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} - \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{ii}, \Phi_{j\bar{\mathbf{y}}} \rangle \right. \\
&\left. - \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{i\mathbf{y}}, \Phi_{jj} \rangle \right\} \\
&= -\sum_{i,\mathbf{y}} \gamma_{i\mathbf{y}} l_{i\mathbf{y}} - \tfrac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{i\mathbf{y}}, \Phi_{j\bar{\mathbf{y}}} \rangle \\
&-\tfrac{1}{2} \left\{ -\sum_{i,j} \sum_{\bar{\mathbf{y}}} \gamma_{j\bar{\mathbf{y}}} \langle \Phi_{ii}, \Phi_{j\bar{\mathbf{y}}} \rangle \sum_{\mathbf{y}} \gamma_{i\mathbf{y}} \right. \\
&\left. - \sum_{i,j} \sum_{\mathbf{y}} \gamma_{i\mathbf{y}} \langle \Phi_{i\mathbf{y}}, \Phi_{jj} \rangle \sum_{\bar{\mathbf{y}}} \gamma_{j\bar{\mathbf{y}}} \right\} \\
&= -\sum_{i,\mathbf{y}} \gamma_{i\mathbf{y}} l_{i\mathbf{y}} - \tfrac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\bar{\mathbf{y}}} \gamma_{i\mathbf{y}}\gamma_{j\bar{\mathbf{y}}} \langle \Phi_{i\mathbf{y}}, \Phi_{j\bar{\mathbf{y}}} \rangle .
\end{aligned}$$

Appendix 2

Feed equation (9) into the objective function (4), we can get

$$\begin{aligned}
&J\left(\gamma_{i\mathbf{y}^+}^{k+1}, \gamma_{i\mathbf{y}^-}^{k+1}\right) = J(\lambda) \\
&= -\sum_{\mathbf{y}=\mathbf{y}^+,\mathbf{y}^-} \gamma_{i\mathbf{y}}^{k+1} l_{i\mathbf{y}} - \sum_{j} \sum_{\mathbf{y}=\mathbf{y}^+,\mathbf{y}^-} \gamma_{i\mathbf{y}}\gamma_{j\mathbf{y}} K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) \\
&+\tfrac{1}{2} \sum_{\mathbf{y}=\mathbf{y}^+,\mathbf{y}^-} \gamma_{i\mathbf{y}}\gamma_{i\mathbf{y}} K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i) \\
&= -\left[\gamma_{i\mathbf{y}^+}^k + \lambda\right] l_{i\mathbf{y}^+} - \left[\gamma_{i\mathbf{y}^-}^k - \lambda\right] l_{i\mathbf{y}^-} \\
&- \left[\left(\gamma_{i\mathbf{y}^+}^k + \lambda\right) \sum_{j\neq i} \left[\gamma_{j\mathbf{y}^+}^k K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j)\right] \right. \\
&+ \left(\gamma_{i\mathbf{y}^-}^k - \lambda\right) \sum_{j\neq i} \left[\gamma_{j\mathbf{y}^-}^k K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j)\right] \\
&+ \left(\tfrac{1}{2}\left(\gamma_{i\mathbf{y}^+}^k\right)^2 + \lambda\gamma_{i\mathbf{y}^+}^k + \tfrac{1}{2}\lambda^2\right) K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i) \\
&\left. + \left(\tfrac{1}{2}\left(\gamma_{i\mathbf{y}^-}^k\right)^2 - \lambda\gamma_{i\mathbf{y}^-}^k + \tfrac{1}{2}\lambda^2\right) K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i) \right] \\
&= \lambda \left\{ -l_{i\mathbf{y}^+} - \sum_{j} \gamma_{j\mathbf{y}^+}^k K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j) \right. \\
&\left. + l_{i\mathbf{y}^-} + \sum_{j} \left[\gamma_{j\mathbf{y}^-}^k K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j)\right] \right\} \\
&-\lambda^2 K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i) + \text{constant}.
\end{aligned}$$

This is a quadratic function of λ. Considering equation (8), it is very easy to get the optimal solution of λ:

$$\lambda^* = \frac{g_i^k(\mathbf{y}^+) - g_i^k(\mathbf{y}^-)}{2K_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_i)}.$$