Abstract - Training Non-linear Structured Prediction Models with Stochastic Gradient Descent

Thomas Gärtner THOMAS.GAERTNER@IAIS.FRAUNHOFER.DE
Shankar Vembu SHANKAR.VEMBU@IAIS.FRAUNHOFER.DE
Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany

1. Introduction

Recently, we proposed a structured prediction approach (Gärtner & Vembu, 2008) that does not depend on a separation oracle for training. The resulting formulation was an unconstrained polynomially-sized quadratic program. Our approach can be trained in polynomial time even in applications such as predicting dicycles or posets, where separation is a computationally hard problem. In this work, we investigate the scalability of our approach with online optimisation.

Structured prediction models (Taskar et al., 2005; Tsochantaridis et al., 2005) infer a joint scoring function on input-output pairs and, for a given input, predict the output that maximises this function. These algorithms make different assumptions in order to ensure polynomial time training. For example, the approach of (Taskar et al., 2005) can be trained in polynomial time only if deciding whether no output with a score higher than a given output exists (*optimality*) is in NP. Often stronger assumptions are needed (Tsochantaridis et al., 2005) and in applications considered thus far these do hold. For several applications such as predicting dicycles in a graph or predicting posets, optimality is coNP-complete and these assumptions do not hold (unless NP=coNP). Our approach (Gärtner & Vembu, 2008) can be trained in polynomial time even in cases where optimality is coNP-complete.

There has been recent work on online structured prediction. Whilst the aproach of (Bordes et al., 2007) can be trained on non-linear models, the subgradient method (Ratliff et al., 2007) is applicable only to linear models. These approaches cannot be trained in polynomial time for applications where optimality is coNP-complete.

2. Count-based Training

We briefly describe the stuctured output learning algorithm proposed in (Gärtner & Vembu, 2008). We consider structured output prediction with the following ingredients: The input space is a set $\mathcal{X}$ with a positive definite kernel $k_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The output space $\mathcal{Y}$ is a set system $(\Sigma, \mathcal{Y})$, where Σ is some fixed set, with a positive definite kernel $k_\mathcal{Y} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ such as the intersection kernel $k_\mathcal{Y}(z, z') = |z \cap z'|$. The goal is to infer an ordering of $\mathcal{Y} \subseteq 2^\Sigma$ for each $x \in \mathcal{X}$. The scoring function is $h \in \mathcal{H} = \mathcal{H}_\mathcal{X} \otimes \mathcal{H}_\mathcal{Y}$ where $\otimes$ denotes the tensor product and $\mathcal{H}_\mathcal{X}, \mathcal{H}_\mathcal{Y}$ are the RKHS of $k_\mathcal{X}, k_\mathcal{Y}$ respectively. Note that the reproducing kernel of $\mathcal{H}$ is then $k[(x, y), (x', y')] = k_\mathcal{X}(x, x')k_\mathcal{Y}(y, y')$. The training set is $\{x_i, Y_i\}_{i \in [\![n]\!]} \subseteq \mathcal{X} \times 2^\mathcal{Y}$, where $[\![n]\!] = \{1, \ldots, n\}$. Often in literature only $|Y_i| = 1$ is considered; in this case we write $\{x_i, y_i\}_{i \in [\![n]\!]} \subseteq \mathcal{X} \times \mathcal{Y}$. For each x_i, we aim at ordering $y \in Y_i$ before $z \in \mathcal{Y} \backslash Y_i$.

The optimisation problem for learning structured outputs is given as

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \lambda \|h\|^2 + \sum_{i \in [\![n]\!]} \mathcal{R}(h, i) \qquad (1)$$

where $\mathcal{R} : \mathcal{H} \times [\![n]\!] \to \mathbb{R}$ is the empirical risk on a training instance, $\lambda \geq 0$ is the regularisation parameter, and $[\![n]\!] = \{1, ..., n\}$. Existing margin-based structured output learning algorithms (Taskar et al., 2005; Tsochantaridis et al., 2005) typically minimise the hinge loss for classification and make different assumptions in order to cope up with the exponentially large output space which results in an exponential number of constraints in the optimisation problem. We take a different approach to arrive at a polynomially-sized unconstrained QP to solve (1). Instead of minimising the hinge loss, we minimse an upper bound on the auc-loss (number of misordered pairs) given as

$$\mathcal{R}_{\text{texp}}(h, i) = \sum_{y \in Y_i} \sum_{z \in \mathcal{Y} \backslash Y_i} \text{texp}\left[h(x_i, z) - h(x_i, y)\right]$$

where $\text{texp}(a) = 1 + a + \frac{1}{2}a^2$. This results in simplifying the problem (1). The representer theorem for structured output states that there is a minimiser h^* of (1) with $h^* \in \mathcal{F} = \textbf{span}\{k[(x_i, z), (\cdot, \cdot)] \mid i \in [\![n]\!], z \in \mathcal{Y}\}$.

As usually $|\mathcal{Y}|$ is exponential in the input of our learning problem, we can not optimise over functions in $\mathcal{F}$ directly. If, however, $\mathbf{span}\{k_{\mathcal{Y}}(z,\cdot) \mid z \in \mathcal{Y}\}$ has a basis $u_1(\cdot), \dots u_m(\cdot)$ with m polynomial in the input of our learning problem it is sufficient to optimise over nm variables only, as $\mathbf{span}\{k_{\mathcal{X}}(x_i,\cdot) \otimes u_l(\cdot) \mid i \in [\![n]\!], l \in [\![m]\!]\} = \mathcal{F}$. We thus have

$$f_{\boldsymbol{\alpha}}(x,z) = \sum_{i\in[\![n]\!],l\in[\![m]\!]} \boldsymbol{\alpha}_{il} k_{\mathcal{X}}(x_i,x) \langle u_l(\cdot), k_{\mathcal{Y}}(z,\cdot)\rangle \ .$$

We now consider finite dimensional output embeddings in which we have a $\phi : \mathcal{Y} \to \mathbb{R}^d$ with $k_{\mathcal{Y}}(y,y') = \langle\phi(y),\phi(y')\rangle$ and the vector $\Phi = \sum_z \phi(z)$ as well as the matrix $C = \sum_z \phi^\top(z)\phi(z)$ can be computed in polynomial time for all of $\mathcal{Y}$ or a relaxation $\tilde{\mathcal{Y}}$. In this case the optimisation problem can be solved in polynomial time. Let Y be the matrix $Y \in \mathbb{R}^{n\times d}$ such that $Y_{i\cdot} = \sum_{y\in Y_i} \phi^\top(y)$ and K be the kernel matrix such that $K_{ij} = k_{\mathcal{X}}(x_i,x_j)$. Let $\circ$ denote the Hadamard product, let $\mathbf{tr}$ denote the trace operator, and let $\mathbf{diag}$ be the operator that maps a square matrix to the column vector corresponding to its diagonal as well as a column vector to the corresponding diagonal matrix. We obtain the optimisation problem

$$\begin{aligned}\operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^{d\times n}} &\lambda\, \mathbf{tr}\, \boldsymbol{\alpha} K \boldsymbol{\alpha}^\top + \frac{1}{2}\mathbf{tr}\, K\boldsymbol{\alpha}^\top C \boldsymbol{\alpha} K \\ &+ 2\Phi^\top \boldsymbol{\alpha} K \mathbf{1} + \frac{|\mathcal{Y}|}{2}\|\mathbf{diag}(Y\boldsymbol{\alpha}K)\|^2 \\ &- 2|\mathcal{Y}|\, \mathbf{tr}\, Y\boldsymbol{\alpha}K - \Phi^\top\boldsymbol{\alpha}K\mathbf{diag}(Y\boldsymbol{\alpha}K) \ .\end{aligned} \tag{2}$$

that can be solved using conjugate gradient methods.

3. Online Optimisation

The optimisation problem (2) can be solved using online algorithms like stochastic gradient descent (SGD) and stochastic meta descent (SMD). In this work, we derive SGD updates in RKHS $\mathcal{H}$.

3.1. Stochastic Gradient Descent

At iteration t, let $K_t = K_{[\![t]\!][\![t]\!]} \in \mathbb{R}^{t\times t}$, $k_t = (K_{[\![t]\!][\![t]\!]})_{\cdot t}$, $y_t = Y_{t\cdot}$, and let $\boldsymbol{\alpha}_t \in \mathbb{R}^{d\times t}$ be the parameter matrix. In the following, we omit the subscript t. The instantaneous objective of 2 can be written as

$$\begin{aligned}\operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^{d\times t}} &\lambda\, \mathbf{tr}\, \boldsymbol{\alpha}K\boldsymbol{\alpha}^\top + \frac{1}{2}k^\top\boldsymbol{\alpha}^\top C\boldsymbol{\alpha}k + 2\Phi^\top\boldsymbol{\alpha}k \\ &+\frac{|\mathcal{Y}|}{2}(y\boldsymbol{\alpha}k)^2 - 2|\mathcal{Y}|y\boldsymbol{\alpha}k - \Phi^\top\boldsymbol{\alpha}ky\boldsymbol{\alpha}k\end{aligned} \tag{3}$$

with gradient ∇,

$$\begin{aligned}&2\lambda\boldsymbol{\alpha}K + C\boldsymbol{\alpha}kk^\top + 2\Phi k^\top + |\mathcal{Y}|y\boldsymbol{\alpha}ky^\top k^\top \\ &-2|\mathcal{Y}|y^\top k^\top - \Phi k^\top y\boldsymbol{\alpha}k - \Phi^\top\boldsymbol{\alpha}ky^\top k^\top \ .\end{aligned} \tag{4}$$

With step size η_t, we update $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \eta_t\nabla$.

3.2. Kernel Expansion Coefficients

As the function (3) is minimised in RKHS, the parameter matrix $\boldsymbol{\alpha}$ grows incrementally with time by adding a single row in every iteration. In order to speed up computations, we truncate all parameters that were updated in the past beyond time τ. This is justified for regularised loss functions because at every iteration, $\alpha_{i\cdot}$ with $i < t$ is shrunk by a factor $(1 - \lambda\eta_t)$ and therefore the contribution of old parameters in the computation of the kernel expansion decreases with time (Vishwanathan et al., 2006). We use this simple technique to speed up computations in our experiments.

3.3. Step Size Adaptation

The step size plays an important role in the convergence of stochastic approximation techniques and has been the focus of recent research (Vishwanathan et al., 2006). We set the step size $\eta_t = \frac{p}{\lambda t}$ where p is a parameter that has to be fine tuned to obtain good performance. A better way to set the step size would be to consider SMD updates .

4. Applications and Experiments

4.1. Multilabel Prediction

We consider multilabel ($\mathcal{Y} = 2^\Sigma$) prediction with $\phi : \mathcal{Y} \to \mathbb{R}^\Sigma$ defined as $\phi_i(z) = 1$ if $z = i$ and 0 otherwise. Here $\Phi = 2^{|\Sigma|-1}\mathbf{1}, C = 2^{|\Sigma|-2}\mathbf{I} + 2^{|\Sigma|-2}\mathbf{1}$. Exact decoding is very simple. For a given (test) instance x let $\kappa \in \mathbb{R}^n$ with $\kappa_i = k_{\mathcal{X}}(x_i,x)$ such that $\hat{y} = \{e \in \Sigma \mid [\boldsymbol{\alpha}\kappa]_e \geq 0\}$.

We trained a simple multilabel classification model to learns the identity function $f : \{0,1\}^d \to \{0,1\}^d$. The goal was to compare batch and online training in terms of the final objective value in (2) and training time. We also studied the effects of the truncation parameter τ on speed and final objective. In our experiments, we trained SGD in a single pass of the data set.

Figure 1 summarises the results for multilabel classification on an artificial data set with 5 features and labels. In the left figure, we set the truncation parameter τ to $0.15 \times n$, where n is the number of training instances. We see that the final solution of SGD is comparable to that of NCG, and the speed up achieved by SGD is apparent for large data sets. The effect of τ on training time and objective is also shown in the figure (center). The training time of SGD increases with τ and attains the training time of NCG at around 19%

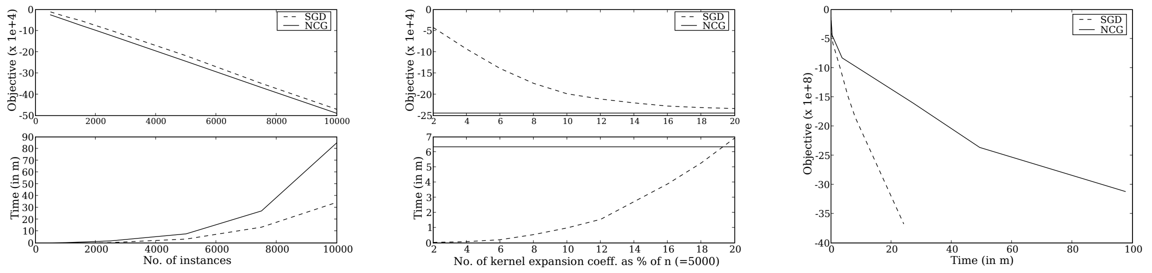

Figure 1. Left: Comparison of SGD and NCG on multilabel classification. *Center:* Effect of τ on training time and final objective value on multilabel classification. *Right:* Comparison of SGD and NCG on dicycle policy estimation.

of n. Beyond this value, we found that SGD was taking longer time than NCG. This underlines the effect of τ when performing SGD updates in RKHS.

4.2. Dicycle Policy Estimation

We consider digraphs $\Sigma = \{(u, v) \mid u, v \in [\![N]\!] \wedge u \neq v\}$ with $\mathcal{Y} \subseteq 2^{\Sigma}$ such that all $y \in \mathcal{Y}$ form a directed cycle $([\![N]\!], y)$. Let $\phi : \mathcal{Y} \to \mathbb{R}^{\Sigma}$ with $\phi_{(uv)}(z) = 1$ if $(u, v) \in z \wedge (v, u) \notin z$, $\phi_{(uv)}(z) = -1$ if $(v, u) \in z \wedge (u, v) \notin z$, and $\phi_{(uv)}(z) = 0$ otherwise. Then we have $\Phi = \mathbf{0}$. With $b(p, q, r) = \sum_{i=p}^{N} \binom{N-q}{i-q} (i - r)!$ we find $|\mathcal{Y}| = b(2, 0, 1)$ and $C_{(u,v),(u',v')} = (2\sigma[\{u, u'\} + \{v, v'\} - \{u, v'\} - \{v, u'\}] - 1) \cdot b(3, 2 - |\{u, v\} \cap \{u, v'\}|, 2 - |\{u, v\} \cap \{u, v'\}|)$. Exact decoding is hard in this case, but one can obtain approximation guarantees (Gärtner & Vembu, 2008).

We simulate the problem of predicting the cyclic tour of different people. We assume there is a hidden policy for each person and each person takes the route that (approximately) maximises the reward of the route. The learned function is linear in the output space ($f(x_i, y) = \langle f^i_{\boldsymbol{\alpha}}, \phi(y) \rangle$) and for testing we can check how well $f^i_{\boldsymbol{\alpha}}$ approximates the hidden policy in the test ($i \in [\![n']\!]$) set. The data is constructed as follows: (i) generate uniformly at random M matrices $A^{(i)} \in \mathbb{R}^{\Sigma \times \Sigma}$ with entries in the interval $[-1, 1]$ and $A^{(i)}_{uv} = -A^{(i)}_{vu}$; ($ii$) generate uniformly $(n + n')M$ random numbers between 0 and 1 to form the inputs $x_i \in \mathbb{R}^M$; (iii) create the output structures $y_i \approx \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{(u,v) \in y, j \in [\![M]\!]} x_{ij} A^{(j)}_{uv}$ for training, that is $i \in [\![n]\!]$. On the test set we measure the performance of our algorithm by cosine similarity of the learned policy and the true policy.

We trained SGD and NCG on data sets of varying size from 100 to 5000 with $M = 15$ and $\Sigma = 10$. We fixed τ to 500 kernel expansion coefficients. Figure 1 (right) shows a plot of final objective versus training time of SGD and NCG on the different data sets. The plot shows that NCG takes a much longer time to attain the same final objective value as SGD. Note that as we perform single pass training, with fixed amounts of training instances NCG attains a smaller value of the objective function than SGD. However, as SGD can deal with much more training instances in the same time, after a fixed amount of time, SGD attains a smaller value of the objective function than NCG.

5. Conclusions

We investigated stochastic gradient descent methods for online training of non-linear structured prediction models. In future, we will explore SMD updates and perform large scale training on real world data sets.

References

Bordes, A., Bottou, L., Gallinari, P., & Weston, J. (2007). Solving multiclass support vector machines with larank. *Proc. ICML.*

Gärtner, T., & Vembu, S. (2008). *A counting approach to structured output learning* (Technical Report). Fraunhofer IAIS. In preparation.

Ratliff, N., Bagnell, J. A., & Zinkevich, M. (2007). (Online) subgradient methods for structured predition. *Proc. AISTATS.*

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. (2005). Learning structured prediction models: A large margin approach. *Proc. ICML.*

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, *6*, 1453–1484.

Vishwanathan, S., Schraudolph, N. N., & Smola, A. J. (2006). Step size adaptation in reproducing kernel hilbert space. *JMLR*, *7*, 1107 – 1133.