Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels

Hanchen Xiong[1], Sandor Szedmak, Justus Piater

Institute of Computer Science, University of Innsbruck, Technikerstr.21a, A-6020 Innsbruck, Austria

Abstract

This paper studies how joint training of multiple support vector machines (SVMs) can improve the effectiveness and efficiency of automatic image annotation. We cast image annotation as an output-related multi-task learning framework, with the prediction of each tag's presence as one individual task. Evidently, these tasks are related via dependencies between tags. The proposed joint learning framework, which we call *joint SVM*, is superior to other related models in its impressive and flexible mechanisms in exploiting the dependencies between tags: first, a linear output kernel can be implicitly learned when we train a joint SVM; or, a pre-designed kernel can be explicitly applied by users when prior knowledge is available. Also, a practical merit of joint SVM is that it shares the same computational complexity as one single conventional SVM, although multiple tasks are solved simultaneously. Although derived from the perspective of multi-task learning, the proposed joint SVM is highly related to structured-output learning techniques, *e.g.* max-margin regression [1], structural SVM [2]. According to our empirical results on several image-annotation benchmark databases, our joint training strategy of SVMs can yield substantial improvements, in terms of both accuracy and efficiency, over training them independently. In particular, it compares favorably with many other state-of-the-art algorithms. We also develop a "perceptron-like" online learning scheme for joint SVM to enable it to scale up better to huge data in real-world practice.

Keywords: image annotation, multi-label learning, output kernels, maximum margin
2014 MSC: 00-01, 99-00

1. Introduction

Automatic image annotation is an important yet challenging machine learning task. The importance is based on the fact that the number of images grows increasingly fast on the internet, and most of them have no link to semantic tags (or keywords, labels). Therefore, automatic annotation is of great significance to generate meaningful meta-data for organizing image collections, and in particular, retrieving images from textual queries. The challenges are usually considered from two classical perspectives [3]: first, *semantic-gap*, *i.e.* the gap from low-level image features to textual tags is large and there exist no reliable way to extract dependencies between them; secondly, *absence of correspondence*, *i.e.* for each tag associated with one image, there is no corresponding region annotated, which hinders learning worse. Meanwhile, when considering contemporary image annotation, one more difficulty to bear is big data. The image data on internet is usually presented in large volumes (million or billion level), so the desired learning method should be capable of working on large-scale data with high learning and prediction efficiency. One straight-forward yet naive strategy is to consider each tag's presence as a binary classification problem. Then, multiple binary classifiers, *e.g.* support vector machines (SVMs), can be trained independently for different tags. This method, however, will suffer from high computational complexity in both training and prediction

Email addresses: hanchen.xiong@uibk.ac.at (Hanchen Xiong), sandor.szedmak@uibk.ac.at (Sandor Szedmak), justus.piater@uibk.ac.at (Justus Piater)
[1]Corresponding author

phases when the number of tags is relatively large. In addition, independently learning multiple SVMs is not expected to work well because it ignores the dependencies between the presences of tags [4], which is a phenomenal characteristic of image annotation tasks (*e.g.*, sky and cloud often co-occur).

In this paper, we propose to interpret image annotation as the learning of multiple related tasks. However, different from most existing multi-task learning frameworks [5] in which tasks are related through their *inputs*, our joint learning method focuses on the relation between *outputs*. Our strategy is motivated by two intuitions. First, by connecting multiple SVM classifiers together, the dependencies between their outputs (the presences of tags), presumably, can be more easily encoded. Secondly, if the outputs of multiple SVMs can be merged into a single vector entity, the optimization problem can be established and solved over vectors, greatly reducing the computational complexity. These two objectives, surprisingly, can be easily achieved by summing up the objectives and constraints in different SVMs, plus an appropriately designed kernel on outputs. We refer to the proposed training strategy as *joint SVM*. The key strength of joint SVM is that it can flexibly offer two mechanisms to exploit the dependencies between tags: first, when there is no prior knowledge on the dependencies, a linear output kernel can be implicitly learned when we train a joint SVM; or a pre-designed, prior-oriented, kernel can be applied on outputs when prior knowledge is available (see section 4). In addition, as we will see in section 3, the training of joint SVM shares almost the same computation complexity as a single regular SVM, which is a practical merit when the number of tags is relative large. Interestingly, although derived from the perspective of multi-task learning, the proposed joint SVM highly relates to structured-output learning techniques, such as max-margin regression [1], structural SVM [2] or max-margin Markov network (M^3N) [6]. More connections between them will be exploited in section 5. In addition, to enable joint SVM to scales up to huge data (million or billion level) in real-world practice, we develop a "perceptron-like" online learning algorithm for joint SVM in section 6. In our experiment (section 7), we tested joint SVM on several benchmark image-annotation databases, with comparison against independent SVMs and other results reported in state-of-the-art algorithms. The experimental results show that our joint SVM can gain impressive improvement over training SVMs independently. In particular, it compares favorably with many other state-of-the-art algorithms.

2. Related Work

Prior to our work, there exist many literatures on image annotation in computer vision and machine learning communities [7, 8, 9, 10, 3, 11, 12, 13, 14, 4, 15, 16]. Roughly speaking, all algorithms can be categorized into generative methods or discriminative methods according to how the relevance between image features and textual tags are modeled. On one hand, generative methods, mostly inspired by linguistic translation studies, model the generating or formating procedure of visual features and tags, then tags prediction from a novel image is inferred by leveraging co-occurrence statistics between visual features and tags in training data. Continuous Relevance Model (CRM) [7], Correlation Latent Dirichlet Allocation (CorLDA) [8] and Multiple Bernoulli Relevance Model (MBRM) [9] belong to the generative category. However, one drawback of these method is that usually some statistical assumptions (*e.g.* conditional independence) are imposed on models, which restricts their modeling capabilities. Furthermore, another practical obstacle of most generative methods is the intractability of inference in tag prediction phase, therefore, usually some approximation techniques are applied. On the other hand, discriminative methods directly model the tag-predicting function, out of which TagProp [10], JEC [3] are metric-learning based approaches, rank-SVM [17], LM-K [18] are rank-learning based approaches, M3L [4] and Multi-Label Relationship Learning (MLRL) [16] are maximum-margin based approaches. One notable issue, and also difficulty, in discriminative methods is the dependencies between output tags, of which many state-of-the-art studies [4, 11, 18] have being aware. In several recent studies [10, 3, 11], discriminative methods were reported to displayed empirically superior performance than generative ones on image annotation task. More comparison and analysis on different representative methods can be found in up-to-date reviews [3, 4, 11].

The proposed joint SVM in this paper is a maximum-margin based, discriminative learning framework. Although joint SVM displays strong connections with structured-output learning, the staring point of our work is to improve the annotation performance by exploiting the relationship between individual tag-predictors. A conceptually-related work was concurrently, but independently from us, presented in MLRL

[16], of which the authors explicitly model the relationship as a covariance matrix in matrix-variate normal distribution over individual model parameters. In contrast, in joint SVM, the dependency between different tags are encoded in output kernels. In this sense, our work is also similar to LM-K [18] and M3L [4]. Interestingly, when the output kernel is linear, it is equivalent to the explicit relationship learning in MLRL. Meanwhile, more sophisticated output kernel can flexibly be constructed and utilized in joint SVM, to afford nonlinear, higher-order dependencies, although they are not always of help in practice.

3. Joint Learning of Multiple SVMs

3.1. Support Vector Machines and Input Kernels

In the past two decades, support vector machines (SVMs) have displayed remarkable successes in various application domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [19] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are ith and jth input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Most popularly, a Gaussian (or radial basis function) kernel

$$K_\phi^{Gau}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 / 2\sigma^2\right) \tag{2}$$

is employed because its corresponding RKHS is an unnormalized Gaussian density function:

$$\phi^{Gau}(\mathbf{x}^{(i)}) \propto \mathcal{N}(\tau; \mathbf{x}^{(i)}, \sigma) \tag{3}$$

which is of infinite dimension, and thus greatly improves the representational capability of input data. Another popular kernel function is Polynomial kernel:

$$K_\phi^{Pol}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + c\right)^d \tag{4}$$

In particular, when the degree $d = 1$ and constant term $c = 0$, Polynomial is a simple inner product. Meanwhile, in 2-degree ($d = 2$) Polynomial kernel, corresponding feature map is:

$$\phi^{Pol}(\mathbf{x}) = [x_d^2, \cdots, x_1^2, \sqrt{2}x_d x_{d-1}, \cdots, \sqrt{2}x_2 x_1, \sqrt{2c}x_d, \cdots, \sqrt{2c}x_1, c]^\top \tag{5}$$

Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$ of one binary classification problem, the primal form of training SVM is written

$$\begin{array}{ll} \arg\min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^m \xi^{(i)} \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)})\right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}$ is the normal vector of a linear hyperplane in $\mathcal{H}_\phi$ (here and later we use $\mathcal{H}$ as $dim(\mathcal{H})$ for simplicity when we denote dimensionality) , $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is trade-off parameter de between training error and max-margin regularization. Eq.(6) differs from usual SVM formulation slightly at the absence of a bias term. Here we ignore the bias since it can be absorbed in $\mathbf{w}$ [2]. Actually, eliminating the bias is more critical in predicting multiple dependent labels, check [4] for

[2]When a Polynomial kernel is used, a bias term is already in its corresponding feature map. When a Gaussian kernel is used, an input vector can be augmented with one extra constant.

detailed explanations. The computational advantage of kernels become obvious when the primal form of SVM (Eq.(6)) is reformulated to its dual form by introducing Lagrange multipliers α_i for each constraints:

$$\begin{array}{rl} \arg\min_{\alpha_1,\alpha_2,\ldots,\alpha_m} & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y^{(i)}y^{(j)}K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t.} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{7}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^m \alpha_i y^{(i)}\phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \operatorname{sgn}\left(\mathbf{w}^\top\phi(\hat{\mathbf{x}})\right) = \operatorname{sgn}\left(\sum_{i=1}^m \alpha_i y^{(i)}K_\phi(\mathbf{x}^{(i)},\hat{\mathbf{x}})\right). \tag{8}$$

It can be seen that the kernel function K_ϕ enables the learning of a high-dimensional (even infinite) $\mathbf{w}$ without explicit computation in $\mathcal{H}_\phi$. Eq.(8) shows that the kernel function yields a similarity measurement between two input instances, and the prediction is working as a weighted-sum of all outputs in the training data.

3.2. Joint SVM

The automatic image annotation task seeks to predict the presence of tags given an input image. Assume d-dimensional visual features are extracted from input images and there are T tags in a pre-defined dictionary, then the annotation learning task is to seek a function $f : \mathbf{x} \in \mathbb{R}^d \to \{-1,+1\}^T$. If we consider prediction of each tag's occurrence as a binary classification problem, we can list as many SVMs as the number of tags. Similar to other multi-task learning frameworks [5], we connect the learning tasks of different SVMs by simply summing up their objectives and constraints respectively in the primal form

$$\begin{array}{rl} \min & \frac{1}{2}\sum_{t=1}^T ||\mathbf{w}_t||^2 + C\sum_{t=1}^T\sum_{i=1}^m \xi_t^{(i)} \\ \text{w.r.t.} & \mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi\times 1} \\ \text{s.t.} & \sum_{t=1}^T y_t^{(i)}\left(\mathbf{w}_t^\top\phi(x^{(i)})\right) \ge T - \sum_{t=1}^T \xi_t^{(i)} \end{array} \tag{9}$$

where t indexes different tags or learning tasks, and T is the total number of tags. By denoting $\mathbf{y}^{(i)} = [y_1^{(1)},\ldots,y_T^{(i)}]$ and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T};\ldots;\frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (Eq.(9)) as a joint SVM:

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{T\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{10}$$

where $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$, and $\bar{\xi}^{(i)} = \frac{1}{T}\sum_{t=1}^T \xi_t^{(i)}$. Eq.(10) is referred to as *joint SVM*. One rationale of Eq.(10) is that within the joint form of objectives and constraints, learning easy tasks can help the learning of challenging tasks. For example, if training data $(\mathbf{x}^{(i,)}, y_p^{(i)})$ can be easily classified correctly in the pth task (*i.e.*, $y^{(i)}(\mathbf{w}_p^\top\mathbf{x}^{(i)})/T > \frac{1}{T}$), it can offer some 'freedom' to other challenging tasks before violating constraint $\left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle_{\mathcal{H}} \ge 1$. Meanwhile, a more critical strength of Eq.(10) is that a linear output kernel is implicitly learned and absorbed in the model parameters $\mathbf{W}$. More rigorous explanation will be presented later in section 4.1. In addition, another key functionality joint SVM can afford is that we can also, based on our prior knowledge, explicitly define kernel functions on outputs $\mathbf{y}$ to improve their representational power (*e.g.* dependencies). Assume the kernel function defined on outputs are $K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})$ (the output kernel will be explained later) and the corresponding feature map is $\psi : \{-1,+1\}^T \to \mathcal{H}_\psi$, then Eq.(10) is modified to

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{11}$$

Similarly to a single conventional SVM, joint SVM Eq.(11) can be converted to its dual form

$$\begin{array}{rl} \arg\min_{\alpha_1,\cdots,\alpha_m} & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i\alpha_j K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{12}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)})\phi(\mathbf{x}^{(i)})^\top$. It can be seen, with the kernel matrix on outputs pre-computed, that the computational complexity of joint learning (Eq.(12)) is the same as the learning of one single SVM (Eq.(7)), which is a great advantage in efficiency.

Given a test input $\hat{\mathbf{x}}$, the prediction $\phi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^{m} \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{13}$$

Meanwhile, one computational issue is that there is no direct way (say, by inverting Eq.(13)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned} \hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \sum_{i=1}^m \alpha_i \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \end{aligned} \tag{14}$$

In general, there is no closed-form solution to Eq.(14), so usually approximate dynamic programming (ADP) is applied in searching for the optimum $\hat{\mathbf{y}}^*$. Here, we employ a simpler yet effective strategy. Since the number of tags associated with one image is rather small, most of the $\mathbf{y}$ in $\{+1, -1\}^T$ space are bad solutions. Therefore, when the training data size is large, the most likely solutions of Eq.(14), presumably, are covered by the outputs in training data $\{\mathbf{y}\}_{i=1}^m$. Consequently, we can find the optimum $\hat{\mathbf{y}}^*$ via a similar neighbour-based label transferring theme as [3, 10]:

$$\hat{\mathbf{y}}^* = \left(\sum_{k=1}^{K} \mathbf{y}^{(k)} w_k \right) / \sum_{k=1}^{K} w_k \tag{15}$$

$$w_j = \sum_{i=1}^{m} \alpha_i \beta_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) \tag{16}$$

where $k = \{j \in [1, m] : w_j > 0\}$ and maximum $K = 10$ neighbours are taken into account. Since α_i are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ were already computed in the training phase, only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing. Thus, the complexity in predicting is $\mathcal{O}(m)$.

4. Implicit and Explicit Linear Output Kernels on Tag-Sets

To transform the pairwise and triplet-wise dependencies between tags into the inner product of two outputs containing those tags, 2-degree and 3-degree Polynomial kernels are tried in [18] and it was reported that 2-degree is better than 3-degree. In [4, 16, 20], linear feature maps were exploited also for pairwise dependencies. In particular, linear output kernels and models were simultaneously learned in [16, 20], while the output kernel in [4] is pre-computed as a correlation matrix over output vectors. In this paper, based on the experience from previous literatures, we also only focus pairwise dependencies and study linear kernels (although higher-order kernels will also be tried in our experiments, and the performance among different kernels can be checked in section 7). Here we adopted strategies both in [16, 20] and in [4]. At first, we present that the linear output kernel can be implicitly, but more simply compared to [16, 20], learned when we train a joint SVM. Secondly, we developed a novel pre-designed linear kernel function, which can be seen as a replacement of the kernel with correlation matrix used in [4].

4.1. Implicit linear output kernel learning

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$matrix $\mathbf{P}$, via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{17}$$

where $\mathbf{\Omega} = \mathbf{P}^\top\mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top\mathbf{W}$, we can rewrite joint SVM (Eq.(11)) as:

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1,\ldots,m\} \end{array} \tag{18}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in Eq.(18) will be pointless. In [20] one extra regularization on $\mathbf{\Omega}$, $\frac{1}{2}||\mathbf{\Omega}||_F^2$, was added into the objective function, while $||P||_F = 1$ was used in [16]. By contrast, a pseudo regularization on $\mathbf{P}$ is used in [11] via the re-construction loss from manually-corrupted data and $\mathbf{P}$. Here we apply a simpler strategy by using a compact regularizer, $\frac{1}{2}\mathbf{W}^\top\mathbf{\Omega}\mathbf{W}$, resulting in:

$$\begin{array}{rl} \arg\min_{\mathbf{U}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{U}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1,\ldots,m\} \end{array} \tag{19}$$

Remarkably, Eq.(19) is equivalent to Eq.(11) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a simple joint SVM with no explicit kernel on outputs.

4.2. Odds-ratio based kernel

In this paper, we also explicitly design an odds-ratio based kernel over tag-sets to capture pairwise dependencies. The dependency between tags measures how much the appearance of one tag increases or decreases the chance of another tag to occur in the same label set. At first, we can estimate the probability of co-occurrence of two labels, w_r and w_s, form training data:

$$P(w_r, w_s) = \frac{\sum_{i=1,\ldots,m} \mathbf{y}_r^{(i)} = 1 \text{ and } \mathbf{y}_s^{(i)} = 1}{m}. \tag{20}$$

according to which, we can compute the odds ratio, a measure, of the dependency between those words by the well known formula [21]:

$$O_{rs} = \frac{P(w_r, w_s)P(\overline{w_r}, \overline{w_s})}{P(w_r, \overline{w_s})P(\overline{w_r}, w_s)}, \tag{21}$$

where $\overline{w_r}$ means the complement of w_r (counting those sample items where w_r does not occur). Then the odds ratio is symmetrized by taking its logarithm, where the 0 value expresses the independence and the positive (or negative) value corresponds to higher (or lower) co-occurrence of those words than the random case. The higher of the magnitude of the log-odds-ratio shows stronger deviation from the independence.

$$Q_{rs} \leftarrow \log(O_{rs}) \tag{22}$$

The odds-ratio based kernel on a pair of outputs can then be computed:

$$K_\psi^{Odd}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top}\mathbf{Q}\mathbf{y}^{(j)} \tag{23}$$

where $\mathbf{Q}$ is the log-odds-ratio matrix with $\mathbf{Q}_{rs} = Q_{rs}$.

5. Relation to Structured-Output Learning

Interestingly, although derived from a rather different starting point, our joint SVM (Eq.(11)) is the same as Maximum Margin Regression (MMR) [1], wherein the motivation is to seek a linear operator in arbitrary tensor product space $\psi(\mathbf{y}^{(i)}) \otimes \phi(\mathbf{x}^{(i)})$. In addition, Eq.(11) is also related to structural SVM [6, 2] by sharing the same objective, yet with different constraints. An empirical comparison of these two methods on hierarchical-label learning is in [22]. The solution of the MMR stands close to the Minimum Description Length Principle, see for example in [23], by providing a highly compressed description to complex learning problems. In particular, when a linear output kernel and Hamming loss function are used in structural SVM. Structural SVM can be converted to a rather similar formulation as joint SVM by decomposing Hamming loss and $\mathbf{y}$ element-wisely. The detailed derivation was presented in [4].

6. Online Learning of Joint SVM

In real-word applications, the number of images can be very huge and beyond the memory storage and computing capacities of normal PCs. For instance, millions of images are uploaded to Facebook^{TM} and Flicker^{TM} every day. Obviously, the computation of kernel matrix for even daily volume is impractical. The formulation of joint SVM also suggests an implementation of a "perceptron-like" algorithm . For simplicity, here we present the case where no output kernel is applied. We aim to demonstrate the transparency of the formulation of joint SVM, which allows us to inherit most of the machine learning techniques developed earlier. Consider the optimization problem in Eq.(10) when only the error term is minimized

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_{\mathcal{H}_y}) \\ \text{subject to} \quad & \{\mathbf{W} | \mathbf{W} : \mathcal{H}_x \to \mathcal{H}_y, \mathbf{W} \text{ a linear operator}\}, \end{aligned} \tag{24}$$

where λ is a prescribed margin, and the function $h(u)$ denotes the Hinge loss, that is

$$h(u) = \begin{cases} u & \text{if } u > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The error function that we are going to minimize has subgradient with respect to $\mathbf{W}$ and this can be computed independently in an incremental way for each term occurring in the summation Eq.(24). The reader can consult to [24] and [25] for details of incremental subgradient methods. The term-wise subgradient is equal to

$$\partial h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y})|_{\mathbf{W}} = \begin{cases} -\mathbf{y}^{(i)}\phi(x^{(i)})^T & \text{if } \lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

We can define the learning speed with a step size, denoted by s, and we obtain the "perceptron-like" algorithm given in Figure 1. In that algorithm $\mathbf{W}^{norm}$ denotes the L_2 normalized linear operator.

Input of the learner: The sample S, step size s
Output of the learner: $\mathbf{W} \in \mathbb{R}^{\mathcal{H}_y \times \mathcal{H}_x}$
Initialization: $t = 0$; $\mathbf{W}_t = \mathbf{0}$; $\mathbf{W}_t^{norm} = \mathbf{0}$; $\|\mathbf{W}_t\| = 0$
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input-output pair: $(\mathbf{x}_i, \mathbf{y}_i)$
    $\beta_i = \langle \mathbf{y}_i, \mathbf{W}_t^{norm}\boldsymbol{\phi}(x_i)\rangle_{\mathcal{H}_y}$
    if $\beta_i < \lambda$ **then**
      $\mathbf{W}_{t+1} = \mathbf{W}_t + s\mathbf{y}_i\boldsymbol{\phi}(x_i)^T$
      $t = t + 1$
      $\|\mathbf{W}_{t+1}\|^2 = \|\mathbf{W}_t\|^2 + s^2\|\mathbf{y}_i\|^2\|\phi(\mathbf{x}_i)\|^2 + 2s\beta_i$
      $\mathbf{W}_{t+1}^{norm} = \mathbf{W}_{t+1}/\|\mathbf{W}_{t+1}\|$
    end if
  end for
until

(27)

Figure 1: Primal "perceptron-like" online learning algorithm for joint SVM.

The departure from the original perceptron algorithm, see for example in [26] and [27], is very moderate. Here we need to learn a matrix realizing the projection of the input vectors into the output space. The incremental subgradient based update employs the direct product of the corresponding output and input vectors to update the projection matrix. Furthermore a normalization step is also included as a certain regularization step, similar approach is proposed in [28].

A dual version of perceptron algorithm can be derived to learn vector outputs. Assume $\mathbf{W}$ is expressible by the training instances, then we have the optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \sum_{j=1}^{m} \alpha_j \overbrace{\langle \mathbf{y}^{(i)}, \mathbf{y}^{(j)} \rangle}^{\kappa_{ij}^{y}} \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}^{(i)}), \boldsymbol{\phi}(\mathbf{x}^{(j)}) \rangle}^{\kappa_{ij}^{\phi}}) \\ \text{subject to} \quad & \alpha_j \geq 0, \ j = 1, \ldots, m, \end{aligned} \tag{28}$$

The partial derivatives for $\alpha_i, \ k = 1, \ldots, m$ equals to

$$\partial h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi})|_{\alpha_i} = \begin{cases} -\kappa_{ij}^{y} \kappa_{ij}^{\phi} & \text{if } h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Finally the corresponding dual perceptron algorithm is formulated according to Figure 2. An analogue of

Input of the learner: The training set S, step size s,
Output of the learner: $(\alpha_j), \ j = 1, \ldots, m,$
Initialization: $\alpha_j = \mathbf{0}; \ j = 1, \ldots, m,$
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input: $\mathbf{x}^{(i)} \in \mathbb{R}^n$;
    if $(\sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) < \ \lambda$ **then**
      for $j = 1, 2, \ldots, m$ **do**
        $\alpha_j = \alpha_j + s \kappa_{ij}^{y} \kappa_{ij}^{\phi}$
      endif
    end if
  end for
until

(30)

Figure 2: Dual "perceptron-like" online learning algorithm for joint SVM.

the standard Novikoff theorem provides an upper bound on the number of updates and a lower bound on the achievable margin in the primal formulation. Here we follow the derivation that was presented in [29]. Let us define the margin for perceptron learner as

$$\gamma(\mathbf{W}, S, \phi) = \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) \in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W} \boldsymbol{\phi}(\mathbf{x}^{(i)}) \rangle_F}{\|\mathbf{W}\|_F}. \tag{31}$$

Then we can claim the following statement not assuming the normalization step in the algorithm:

Theorem 1. *Let $S = \{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\} \subset (\mathcal{Y} \times \mathcal{X}), \ i = 1, \ldots$ be a sample set independently and identically drawn from an unknown distribution and let $\phi : \mathcal{X} \to \mathcal{H}_\phi$ be an embedding into a Hilbert space, furthermore assume that $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$ for all i, and that the learning rate, the step size, s is a fixed positive real number. Suppose there exists a linear operator $\mathbf{W}^*$ such that $\|\mathbf{W}^*\|_F = 1$ and*

$$\gamma(\mathbf{W}^*, S, \phi) \geq \Gamma, \tag{32}$$

and the algorithm stops when the functional margin 1 is achieved.

1. *Then the number of updates made by Algorithm (1) is bounded by*

$$t \leq \frac{1}{\Gamma^2} \left(1 + \frac{2}{s}\right). \tag{33}$$

2. *Then for the solution* $\mathbf{W}_t$ *in Algorithm (1) we have*

$$\gamma(\mathbf{W}_t, S, \phi) \geq \frac{\Gamma}{s+2}. \tag{34}$$

Proof 1. 1. *Following the proof of the original Novikoff theorem [30], we first upper bound the norm of the matrix* $\mathbf{W}_t$ *obtained after* t *updates:*

$$\begin{aligned} \|\mathbf{W}_t\|_F^2 &= \|\mathbf{W}_{t-1}\|_F^2 + 2s\langle \mathbf{y}^{(i)}\mathbf{W}_{t-1}\boldsymbol{\phi}(x^{(i)})\rangle_{\mathcal{H}_y} + s^2\|\mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T\|_F^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2\|\mathbf{y}^{(i)}\|^2\|\boldsymbol{\phi}(x^{(i)})\|^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2 \\ &\leq ts(s+2). \end{aligned} \tag{35}$$

We now provide a reverse inequality for the inner product with $\mathbf{W}^\star$*:*

$$\begin{aligned} \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T, \mathbf{W}^\star\right\rangle_F \\ &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}, \mathbf{W}^\star\boldsymbol{\phi}(x^{(i)})\right\rangle_{\mathcal{H}_y} \\ &\geq \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\Gamma \\ &\geq ts\Gamma. \end{aligned}$$

Then we can create the squeezing inequality:

$$ts(s+2)\|\mathbf{W}^\star\|_F^2 \geq \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2. \tag{36}$$

implying the result.

2. *Taking the bound Eq.(33) for* t *and substituting into Eq.(35) we arrive at*

$$\|\mathbf{W}_t\|_F \leq \frac{s+2}{\Gamma}. \tag{37}$$

Then for the margin we have

$$\gamma(\mathbf{W}_t, S, \phi) \geq \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W}_t\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_F}{\|\mathbf{W}_t\|_F} \tag{38}$$

$$\geq \frac{1}{\|\mathbf{W}_t\|_F} \tag{39}$$

$$\geq \frac{\Gamma}{s+2}, \tag{40}$$

which proves the statement.

Sparsity bounds [31] can also be used to translate this bound on the number of updates into a corresponding bound on the generalization of the resulting classifier.

All results included in this paper are assumed the normalization conditions, $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$, of Theorem 1. By forcing the normalization of $\|\mathbf{W}_t\|$ for all t in Algorithm 1 allows us to simplify and sharpen the proof of Theorem 1. In this case Expression (35) collapses into a identity of both sides of the equation, therefore instead of (36) we have

$$1 = \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2, \tag{41}$$

from which we gain that

$$t \leq \frac{1}{s\Gamma}, \tag{42}$$

Dataset	labels	Number of training instances	test instances	average labels
Corel5k	260	4500	500	3.3965
Espgame	268	18689	2081	4.6859
Iaprtc12	291	17665	1962	5.7187

Table 1: Statistics of three benchmark datasets.

and in case of the margin we can write

$$\gamma(\mathbf{W}_t, S, \phi) \quad \geq \quad 1, \tag{43}$$

which statements are significantly stronger than those appearing in the general case. The price that we need to pay for this result is the slower algorithm.

In comparing our algorithm with other online learning schemes of maximum margin based learning methods, e.g. SVM, (see some realizations in [32] and [33]), we need to bear in mind that our methods learns to predict all components of the label vector within one optimization problem. Those methods which can deal only with binary classification problems have to solve as many binary label subproblems as the number of labels independently, therefore their overall computational complexity turns to be significantly higher than our approach.

7. Experiment

7.1. Databases

In our experiments, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies [7, 8, 9, 10, 3, 11] with performance evaluations reported therein. Therefore, we can easily compare our method with others. Statistics of three benchmark datasets are summarized in Table 1. Readers are referred to [3] for more details of three datasets.

7.2. Feature Extraction

In our experiment, we worked with 15 visual features extracted in [10]. More concretely, they contain one Gist descriptor, six global color histograms and eight histograms of local bag-of-words texture features [3]. The description of 15 features are summarized in Table 2. Readers are referred to [10] for more detail on extracting these features. These features were also used in [10] and [11]. A similar visual feature set without layout was extracted and used in [3], while 30 visual feature with spatial layouts were used in [9].

7.3. Evaluation metric

In our experiment, we evaluated annotation performance using *precision (P), recall (R), F-1 measure (F)*, which were commonly used in previous studies. For each tag, the precision is computed as ratio between the number of images assigned the tag correctly and total number of images predicted to have the tag, while the recall is the number of images assigned the tag correctly, divided by the number of images which truly have the tag. Then precision and recall are averaged across all tags. At last, F1 measure is calculated as $F = 2\frac{P \times R}{P+R}$.

7.4. Model selection

In three original databases, training/test data are already divided in advance. Therefore, given a learned model, there exist no variance in prediction performance on fixed test data. Hyper-parameters in Gaussian kernels, polynomial kernels and odds-ratio based kernels are found by cross validation restricted to the training data, namely it is divided into validation test and validation training parts. Then the learner is trained only on the validation training items. At the end those values of the parameters have been chosen which maximize the $F1$ score on the validation test.

[3] All features are available on http://lear.inrialpes.fr/people/guillaumin/data.php.

Feature	Dimension	Source	Descriptor	Location	Layout
DenseHue	100	texture	hue	dense	no
DenseHueV3H1	300	texture	hue	dense	yes
DenseSift	1000	texture	sift	dense	no
DenseSiftV3H1	3000	texture	sift	dense	yes
Gist	512	-	holistic	-	-
HarrisHue	100	texture	Hue	Harris points	no
HarrisHueV3H1	300	texture	Hue	Harris points	yes
HarrisSift	1000	texture	sift	Harris points	no
HarrisSiftV3H1	3000	texture	sift	Harris points	yes
Hsv	4096	color	HSV	-	no
HsvV3H1	5184	color	HSV	-	yes
Lab	4096	color	LAB	-	no
LabV3H1	5184	color	LAB	-	yes
Rgb	4096	color	RGB	-	no
RgbV3H1	5184	color	RGB	-	yes

Table 2: Description of 15 visual features tried in our experiments.

	Corel5k			Espgame			iaprtc12		
Feature	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
DenseHue	33.3	26.0	29.2	28.5	16.4	20.8	26.7	17.5	21.1
DenseHueV3H1	38.1	30.7	34.0	32.9	18.8	23.9	31.8	21.0	25.3
DenseSift	40.2	32.2	35.8	33.3	24.6	28.3	38.4	26.5	31.4
DenseSiftV3H1	**43.7**	**34.6**	**38.6**	**35.2**	**26.3**	**30.1**	**40.5**	**28.3**	**33.3**
Gist	33.7	26.9	29.9	28.3	20.6	23.9	33.2	23.5	27.5
HarrisHue	31.0	24.6	27.4	27.4	16.3	20.4	27.6	18.3	22.0
HarrisHueV3H1	34.5	27.7	30.7	31.5	18.4	23.2	31.9	21.9	26.0
HarrisSift	39.9	32.1	35.6	33.2	25.5	28.9	39.4	26.9	32.0
HarrisSiftV3H1	40.2	33.4	36.5	**34.6**	**26.2**	**29.8**	**40.7**	**29.7**	**34.3**
Hsv	38.3	30.6	34.0	30.0	18.7	23.1	32.6	21.1	25.7
HsvV3H1	40.8	33.8	37.0	33.8	21.6	26.4	35.4	24.1	28.7
Lab	35.1	27.5	30.8	27.2	16.4	20.5	28.4	17.9	22.0
LabV3H1	39.7	30.7	34.6	30.0	18.9	23.1	32.7	20.8	25.4
Rgb	42.0	33.4	37.2	26.2	16.4	20.2	32.8	20.6	25.3
RgbV3H1	**42.1**	**34.5**	**38.0**	29.6	19.2	23.3	35.7	23.0	28.0

Table 3: Performance of joint SVM without explicit output kernel on different individual features.

7.5. Selecting optimal features

In [10, 11], all 15 features were used for predicting tags. However, we believe that there exist some redundancies in all 15 features. Also, some features might be weakly relevant to the annotation task. A more efficient way is to identify a few most relevant features and use them for prediction. To this end, we apply joint SVM without explicit output kernel on different features, and list their discriminative abilities in Table 3, in which the best and second runner-up features are highlighted with bold font. We can see that `DenseSfitV3H1` is consistently more reliable than other features in three datasets. In addition, `HarrisSiftV3H1` is also optimal or close to optimal in Espgame and Iaprtc12 respectively. However, `HarrisSiftV3H1` is inferior to `RgbV3H1` in Corel5k. Therefore, in our later experiments, we used `DenseSfitV3H1`+`RgbV3H1` on Corel5k, while `DenseSfitV3H1`+`HarrisSiftV3H1` on Espgame and Iaprtc12. We combined two features by simply concatenating one feature vector after the other one.

	Training Time (sec)	Testing Time (sec)	Testing Performance		
			Precision (%)	Recall (%)	F1 (%)
Independent SVMs (Gau)	6285.11	117.20	15.3	22.1	18.1
Independent SVMs (Pol)	4612.23	147.9	15.1	29.7	20.0
Joint SVM (Gau)	80.68	**6.92**	40.8	37.1	38.9
Joint SVM (Pol)	**76.48**	9.11	**48.5**	**38.0**	**42.6**

Table 4: Comparison between one joint SVM and multiple SVMs on Corel5k dataset. Two input kernels (Gaussian and 2-degree polynomial) are tried in both learners.

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM [9]	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC [3]	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp [10]	33.0	42.0	37.0	39.0	27.0	**32.0**	45.0	**34.0**	**39.0**
FastTag [11]	32.0	**43.0**	37.0	**46.0**	22.0	30.0	**47.0**	26.0	34.0
JSVM	**48.5**	38.0	**42.6**	32.7	**31.6**	**32.2**	42.2	29.4	34.6
JSVM+Odd	**48.8**	37.1	**42.2**	27.4	27.1	27.2	32.9	28.6	30.6
JSVM+Pol(2)	46.6	37.0	41.3	32.6	24.4	27.9	37.9	26.6	31.2
JSVM+Pol(3)	41.5	31.3	35.7	28.5	21.3	24.4	38.0	26.1	31.0
JSVM-Per	37.5	29.8	33.2	25.0	19.0	21.6	29.2	20.8	24.3

Table 5: Comparison between different versions of joint SVM and other related methods on three benchmark databases.

7.6. Comparison with Independent SVMs

At first, we applied both a joint SVM, and many independent SVMs on Corel5k dataset with the feature combination selected above. To ensure fairness, no user-designed kernel is used on output for the joint SVM (plain joint SVM), while Gaussian kernel and 2-degree polynomial kernel are tried for inputs in both learners. In the learning phase, the optimization problems (Eq.(7)) and (Eq.(12)) were solved with the same coordinate descent method [20]. In addition, the same cross-validation procedure is used for both many independent SVMs and the joint SVM to find the best hyper-parameters C, d, c, σ. To measure the efficiency, training and testing time were recorded as well. All experiments were run on the same simulation and hardware conditions (Python 3, Intel Core i7). The comparison of accuracy and efficiency between independent SVMs and joint SVM is presented in Table 4. While the learning and testing time of independent SVMs scale with the number of tags, the computation time of joint SVM approximately equals a SVM for single-tag classification. At the same time, in terms of accuracy, joint SVM also worked much better than independent SVMs. We can also see that 2-degree polynomial input kernel worked better than Gaussian input kernel for both learners.

7.7. Comparison with state-of-the-art

More intensive experiments of joint SVM were conducted with different pre-designed, explicit output kernels: odds-ratio based kernel (JSVM+Odd), 2-degree polynomial (JSVM+Pol(2)), 3-degree polynomial (JSVM+Pol(3)). Also, online learning algorithm of joint SVM (JSVM-Per) was also implemented. All configurations were run on all three datasets, with optimal feature combination and 2-degree polynomial kernel on inputs. The experimental results, together with the reported results from other related work, are presented in Table 5. We can see that plain joint SVM (JSVM) outperforms all other results on Corel5k and Espgame datasets, yielding the best results so far. JSVM is also the second best result on Iaprtc12 dataset. The results of JSVM+Odd and JSVM+Pol(2) are similar on all three datasets. It is worth noting that JSVM+Odd and JSVM+Pol(2) also worked better than previous methods by a large margin. Meanwhile, JSVM+Pol(3) is worse than JSVM+Pol(2). JSVM-Per's performance is inferior to other JSVM versions, although it is still better than two classic methods [9, 3].

7.8. Discussions

Based on our experiments, it seems that plain joint SVM (JSVM) works more robustly than the joint SVMs with explicit output kernels. In order to dig deeper to find an explanation, we can study the correlation matrices of output tag-sets in three datasets. In Figure 3, for each dataset, we plot the histograms (in log scale) of all correlation values in both training sets and testing sets. We can see that most entries in correlation matrices are 0, which means that the pairwise correlation (or roughly speaking, dependencies) is rather sparse. Although JSVM, JSVM+Odd both encode linear pairwise dependencies, it should be reminded that the implicit output kernel in JSVM is in regularization term, which implies that simpler output kernels (dependencies) are encouraged. However, JSVM+Odd does not have this preference. Therefore, JSVM can implicitly learned most simple output kernels when no more complex ones are needed. Analogously, the same principle can explain why JSVM+Pol(2), or even JSVM+Pol(3) led to worse results. If we look closer, we can observe that in Corel5k datasets, stronger correlations are displayed in its testing set, and correspondingly, the performance gaps between JSVM, JSVM+Odd and JSVM+Pol(2) are also rather small.

As for JSVM-Per, one reason of its inferiority is that the regularization is computed instance-wisely, which might conflict the global effect it is supposed to have. However, we gain tractability, for extremely large datasets, with acceptable accuracy cost. As a future direction work, we will investigate some alternative online regularization strategies.

8. Conclusions

A novel joint SVM was presented for automatic image tagging. It is superior to conventional SVMs based on our empirical results. In particular, it compares favorably with state-of-the-art methods. As possible future work directions, we would like to apply and improve joint SVM in other multi-label learning domains.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

Reference

[1] S. Szedmak, J. Shawe-taylor, Learning via linear operators: Maximum margin regression, Tech. rep., University of Southampton, UK (2005).

[2] I. Tsochantaridis, T. Hofmann, T. Joachims, Y. Altun, Support vector machine learning for interdependent and structured output spaces, in: ICML, 2004.

[3] A. Makadia, V. Pavlovic, S. Kumar, Baselines for image annotation, International Journal of Computer Vision 90 (2010) 88–105.

[4] B. Hariharan, S. V. N. Vishwanathan, M. Varma, Efficient max-margin multi-label classification with applications to zero-shot learning, Machine Learning 88 (1-2) (2012) 127–155.

[5] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Machine Learning 73 (3) (2008) 243–272.

[6] B. Taskar, V. Chatalbashev, D. Koller, C. Guestrin, Learning structured prediction models: A large margin approach, in: ICML, 2005.

[7] V. Lavrenko, R. Manmatha, J. Jeon, A model for learning the semantics of pictures, in: NIPs, 2004.

[8] D. M. Blei, M. I. Jordan, Modeling annotated data, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, 2003.

[9] S. L. Feng, R. Manmatha, V. Lavrenko, Multiple bernoulli relevance models for image and video annotation, in: CVPR, 2004.

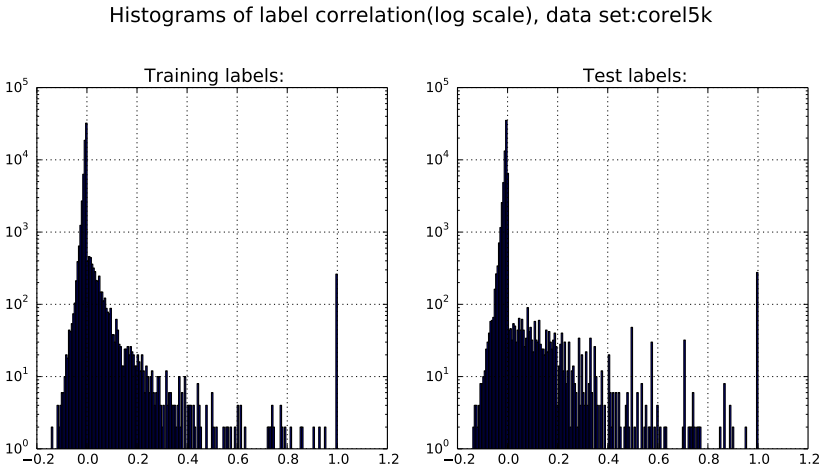
[10] M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: ICCV, 2009.

[11] M. Chen, A. Zheng, K. Q. Weinberger, Fast image tagging, in: ICML, 2013.

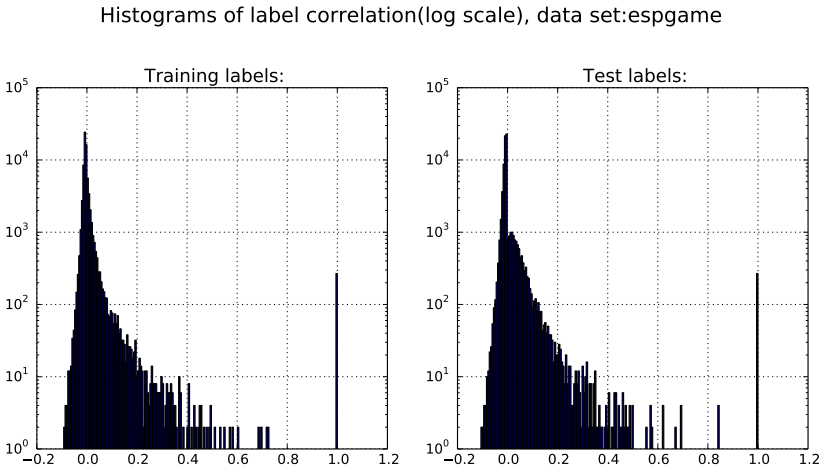
[12] D. R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Computation 16 (2004) 2639–2664.

[13] X. Qi, Y. Han, Incorporating multiple svms for automatic image annotation, Pattern Recogn. 40 (2) (2007) 728–741.

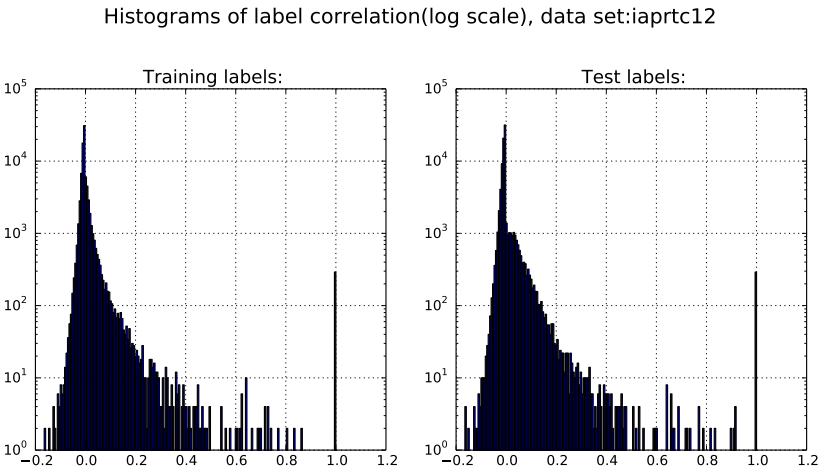
[14] S. Szedmák, T. D. Bie, D. R. Hardoon, A metamorphosis of canonical correlation analysis into multivariate maximum margin learning, in: ESANN, 2007.
[15] J. Rousu, C. Saunders, S. Szedmák, J. Shawe-Taylor, Kernel-based learning of hierarchical multilabel classification models, Journal of Machine Learning Research 7 (2006) 1601–1626.
[16] Y. Zhang, D.-Y. Yeung, Multilabel relationship learning, ACM Trans. Knowl. Discov. Data 7 (2) (2013) 1–30.
[17] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: NIPs, 2001.
[18] Y. Guo, D. Schuurmans, Multi-label classification with output kernels, in: ECML/PKDD, 2013.
[19] V. Vapnik, Statistical learning theory, Wiley, 1998.
[20] F. Dinuzzo, C. S. Ong, P. V. Gehler, G. Pillonetto, Learning output kernels with block coordinate descent, in: ICML, 2011.
[21] S. M. Hailpern, P. F. Visintainer, Odds ratios and logistic regression: further examples of their use and interpretation, Stata Journal 3 (3) (2003) 213–225.
[22] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, J. Rousu, Towards structured output prediction of enzyme function, in: BMC Proceedings, 2(4):S2, 2008.
[23] P. D. Grünwald, The Minimum Description Length Principle, MIT Press, 2007.
[24] D. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scienctific, 1999.
[25] K. Kiwiel, Convergence of approximate and incremental subgradient methods for convex optimization, Journal of Optimization 14, 3 (2004) 807–840.
[26] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
[27] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning and Games, Cambridge University Press, 2006.
[28] C. Gentile, A new approximate maximal margin classification algorithm, Journal of Machine Learning Research 2 (2001) 213–242.
[29] Y. Li, H. Zaragoza, R. Herbich, J. Shawe-Taylor, J. Kandola, The perceptron algorithm with uneven margins, in: Proceedings of the International Conference of Machine Learning (ICML'2002), 2002.
[30] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods, Cambridge University Press, New York, NY, USA, 2000.
[31] T. Graepel, R. Herbrich, J. Shawe-Taylor, Generalisation error bounds for sparse linear classifiers, in: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, Morgan Kaufmann Publishers Inc., 2000, pp. 298–303.
[32] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, Journal of Machine Learning Research 6(Sep) (2005) 1579–1619.
[33] S. Shalev-Shwartz, T. Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, Journal of Machine Learning Research 14(Feb) (2013) 567–599.



(a)



(b)



(c)

Figure 3: The histograms (in log scale) of all correlation values in both training sets and testing sets: (a) Corel5k, (b) Espgame (c) Iartc12.