Optimization of Robust Loss Functions for Weakly-Labeled Image Taxonomies: An ImageNet Case Study

Julian J. McAuley[1], Arnau Ramisa[2], and Tibério S. Caetano[1]

[1] Statistical Machine Learning Group, NICTA, and the Australian National University
{julian.mcauley, tiberio.caetano}@nicta.com.au
[2] Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Spain, aramisa@iri.upc.edu

Abstract. The recently proposed *ImageNet* dataset consists of several million images, each annotated with a single object category. However, these annotations may be imperfect, in the sense that many images contain *multiple* objects belonging to the label vocabulary. In other words, we have a multi-label problem but the annotations include only a single label (and not necessarily the most prominent). Such a setting motivates the use of a *robust* evaluation measure, which allows for a limited number of labels to be predicted and, as long as one of the predicted labels is correct, the overall prediction should be considered correct. This is indeed the type of evaluation measure used to assess algorithm performance in a recent competition on ImageNet data. Optimizing such types of performance measures presents several hurdles even with existing structured output learning methods. Indeed, many of the current state-of-the-art methods optimize the prediction of only a single output label, ignoring this 'structure' altogether. In this paper, we show how to directly optimize continuous surrogates of such performance measures using structured output learning techniques with latent variables. We use the output of existing binary classifiers as input features in a new learning stage which optimizes the structured loss corresponding to the robust performance measure. We present empirical evidence that this allows us to 'boost' the performance of existing binary classifiers which are the state-of-the-art for the task of object classification in ImageNet.

1 Introduction

The recently proposed ImageNet project consists of building a growing dataset using an image taxonomy based on the WordNet hierarchy (Deng et al., 2009). Each node in this taxonomy includes a large set of images (in the hundreds or thousands). From an object recognition point of view, this dataset is interesting because it naturally suggests the possibility of leveraging the image taxonomy in order to improve recognition beyond what can be achieved independently for each image. Indeed this question has been the subject of much interest recently, culminating in a competition in this context using ImageNet data (Berg et al., 2010; Lin et al., 2011; Sánchez and Perronnin, 2011).

Although in ImageNet each image may have several objects from the label vocabulary, the annotation only includes a single label per image, and this label is not necessarily the most prominent. This imperfect annotation suggests that a meaningful performance measure in this dataset should somehow not penalize predictions that contain

legitimate objects that are missing in the annotation. One way to deal with this issue is to enforce a *robust* performance measure based on the following idea: an algorithm is allowed to predict more than one label per image (up to a maximum of K labels), and as long as one of those labels agrees with the ground-truth label, no penalty is incurred. This is precisely the type of performance measure used to evaluate algorithm performance in the aforementioned competition (Berg et al., 2010).

In this paper, we present an approach for directly optimizing a continuous surrogate of this robust performance measure. In other words, we try to optimize the very measure that is used to assess recognition quality in ImageNet. We show empirically that by using binary classifiers as a starting point, which are state-of-the-art for this task, we can boost their performance by means of optimizing the structured loss.

1.1 Literature Review

The success of visual object classification obtained in recent years is pushing computer vision research towards more difficult goals in terms of the number of object classes and the size of the training sets used. For example, Perronnin et al. (2010) used increasingly large training sets of Flickr images together with online learning algorithms to improve the performance of linear SVM classifiers trained to recognize the 20 Pascal Visual Object Challenge 2007 objects; or Torralba et al. (2008), who defined a gigantic dataset of 75,062 classes (using all the nouns in WordNet) populated with 80 million tiny images of only 32×32 pixels. The WordNet nouns were used in seven search engines, but without any manual or automatic validation of the downloaded images. Despite its low resolution, the images were shown to still be useful for classification.

Similarly, Deng et al. (2009) created ImageNet: a vast dataset with thousands of classes and millions of images, also constructed by taking nouns from the WordNet taxonomy. These were translated into different languages, and used as query terms in multiple image search engines to collect a large amount of pictures. However, as opposed to the case of the previously mentioned 80M Tiny Images dataset, in this case the images were kept at full resolution and manually verified using Amazon Mechanical Turk. Currently, the full ImageNet dataset consists of over 17,000 classes and 12 million images. Figure 1 shows a few example images from various classes.

Deng et al. (2010) performed classification experiments using a substantial subset of ImageNet, more than ten thousand classes and nine million images. Their experiments highlighted the importance of algorithm design when dealing with such quantities of data, and showed that methods believed to be better in small scale experiments turned out to under-perform when brought to larger scales. Also a cost function for classification taking into account the hierarchy was proposed. In contrast with Deng et al. (2010), most of the works using ImageNet for large scale classification made no use of its hierarchical structure.

As mentioned before, in order to encourage large scale image classification using ImageNet, a competition using a subset of 1,000 classes and 1.2 million images, called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC; Berg et al., 2010), was conducted together with the Pascal Visual Object Challenge 2010 competition. Notoriously, the better classified participants of the competition used a traditional one-versus-all approach and completely disregarded the WordNet taxonomy.

Fig. 1. Example images from ImageNet. Classes range from very general to very specific, and since there is only one label per image, it is not rare to find images with unannotated instances of other classes from the dataset.

Lin et al. (2011) obtained the best score in the ILSVRC'10 competition using a conventional one-vs-all approach. However, in order to make their method efficient enough to deal with large amounts of training data, they used Local Coordinate Coding and Super-Vector Coding to reduce the size of the image descriptor vectors, and averaged stochastic gradient descent (ASGD) to efficiently train a thousand linear SVM classifiers.

Sánchez and Perronnin (2011) got the second best score in the ILSVRC'10 competition (and a posteriori reported better results than those of Lin et al. (2011)). In their approach, they used high-dimensional Fisher Kernels for image representation with lossy compression techniques: first, dimensionality reduction using Hash Kernels (Shi et al., 2009) was attempted and secondly, since the results degraded rapidly with smaller descriptor dimensionality, coding with Product Quantizers (Jégou et al., 2010) was used to retain the advantages of a high-dimensional representation without paying an expensive price in terms of memory and I/O usage. For learning the standard binary one-vs-all linear classifiers, they also used Stochastic Gradient Descent.

The difficulty of using the hierarchical information for improving classification may be explained by the findings of Russakovsky and Fei-Fei (2010). In their work ImageNet is used to show that the relationships endowed by the WordNet taxonomy do not necessarily translate in visual similarity, and that in fact new relations based only on visual appearance information can be established between classes, often far away in the hierarchy.

2 Problem Statement

We are given the dataset $\mathcal{S} = \left\{(x^1, y^1), \ldots, (x^N, y^N)\right\}$, where $x^n \in \mathcal{X}$ denotes a feature vector representing an image with label y^n. Our goal is to learn a classifier $\bar{Y}(x;\theta)$ that for an image x outputs a set of K distinct object categories. The vector θ 'parametrizes' the classifier $\bar{Y}$; we wish to learn θ so that the labels produced by $\bar{Y}(x^n;\theta)$ are 'similar to' the training labels y^n under some loss function $\Delta(\bar{Y}(x^n;\theta), y^n)$. Our specific choice of classifier and loss function shall be given in Section 2.1.

We assume an estimator based on the principle of regularized risk minimization, i.e. we aim to find θ^* such that

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \Bigg[\underbrace{\frac{1}{N} \sum_{n=1}^{N} \Delta(\bar{Y}(x^n;\theta), y^n)}_{\text{empirical risk}} + \underbrace{\frac{\lambda}{2} \|\theta\|^2}_{\text{regularizer}} \Bigg]. \tag{1}$$

Our notation is summarized in Table 1. Note specifically that each image is annotated with a *single* label, while the output space consists of a *set* of K labels (we use y to denote a single label, Y to denote a set of K labels, and $\mathcal{Y}$ to denote the space of sets of K labels). This setting presents several issues when trying to express (eq. 1) in the framework of structured prediction (Tsochantaridis et al., 2005). Apparently for this reason, many of the state-of-the-art methods in the ImageNet Large Scale Visual Recognition Challenge (Berg et al., 2010, or just 'the ImageNet Challenge' from now on) consisted of binary classifiers, such as multiclass SVMs, that merely optimized the score of a *single* prediction (Lin et al., 2011; Sánchez and Perronnin, 2011).

Motivated by the surprisingly good performance of these binary classifiers, in the following sections we shall propose a learning scheme that will 'boost' their performance by re-weighting them so as to take into account the structured nature of the loss function from the ImageNet Challenge.

2.1 The Loss Function

Images in the ImageNet dataset are annotated with a single label y^n. Each image may contain multiple objects that are not labeled, and the labeled object need not necessarily be the most salient, so the method should not be penalized for choosing 'incorrect' labels in the event that those objects actually appear in the scene. Note that this is not an issue in some similar datasets, such as the Caltech datasets (Griffin et al., 2007), where the images have been selected to avoid such ambiguity in the labeling, or all instances of objects covered in the dataset are annotated in every image, as in the Pascal Visual Object Challenge (Everingham et al., 2010).

To address this issue, a loss is given over a *set* of output labels Y, that only penalizes the method if *none* of those labels is similar to the annotated object. For a training image with label y^n, the loss incurred by choosing the set of labels Y is given by

$$\Delta(Y, y^n) = \min_{y \in Y} d(y, y^n). \tag{2}$$

Table 1. Notation

Notation	Description
x	the feature vector for an image (or just 'an image' for simplicity)
x^n	the feature vector for the n^{th} training image
$\mathcal{X}$	the features space, i.e., $x^n \in \mathcal{X}$
F	the feature dimensionality, i.e., $F = \lvert x^n \rvert$
N	the total number of training images
y	an image label, consisting of a single object class
y^n	the training label for the image x^n
$\mathcal{C}$	the set of classes, i.e., $y^n \in \mathcal{C}$
C	the total number of classes, i.e., $C = \lvert\mathcal{C}\rvert$
$\bar{Y}(x;\theta)$	the *set* of output labels produced by the classifier
$\hat{Y}(x;\theta)$	the output labels resulting in the most violated constraints during column-generation
$\bar{Y}^n$	shorthand for $\bar{Y}(x^n;\theta)$
$\hat{Y}^n$	shorthand for $\hat{Y}(x^n;\theta)$
K	the number of output labels produced by the classifier, i.e., $K = \lvert\bar{Y}^n\rvert = \lvert\hat{Y}^n\rvert$
$\mathcal{Y}$	the space of all possible sets of K labels
θ	a vector parameterizing our classifier
θ^y_{binary}	a binary classifier for the class y
λ	a constant that balances the importance of the empirical risk versus the regularizer
$\phi(x,y)$	the joint parametrization of the image x with the label y
$\Phi(x,Y)$	the joint parametrization of the image x with a *set of labels* Y
$\Delta(Y,y^n)$	the error induced by the set of labels Y when the correct label is y^n
$d(y,y^n)$	a distance measure between the two classes y and y^n in our image taxonomy
Z^n	latent annotation of the image x^n, consisting of $K-1$ object classes distinct from y^n
Y^n	the 'complete annotation' of the image x^n, i.e., $Z^n \cup \{y^n\}$

In principle, $d(y,y^n)$ could be any difference measure between the classes y and y^n. If $d(y,y^n) = 1 - \delta(y = y^n)$ (i.e., 0 if $y = y^n$, 1 otherwise), this recovers the ImageNet Challenge's 'flat' error measure. If $d(y,y^n)$ is the shortest-path distance from y to the nearest common ancestor of y and y^n in a certain taxonomic tree, this recovers the 'hierarchical' error measure (which we shall use in our experiments).

For images with multiple labels we could use the loss $\Delta(Y,Y^n) = \frac{1}{|Y^n|}\sum_{y^n \in Y^n} \Delta(Y,y^n)$, though when using the ImageNet data we always have a single label.

2.2 'Boosting' of Binary Classifiers

Many of the state-of-the-art methods for image classification consist of learning a series of binary 'one vs. all' classifiers that distinguish a single class from all others. That is, for each class $y \in \mathcal{C}$, one learns a separate parameter vector θ^y_{binary}, and then performs classification by choosing the class with the highest score, according to

$$\bar{y}_{\text{binary}}(x) = \operatorname*{argmax}_{y \in \mathcal{C}} \left\langle x, \theta^y_{\text{binary}} \right\rangle. \tag{3}$$

In order to output a set of K labels, such methods simply return the labels with the highest scores,

$$\bar{Y}_{\text{binary}}(x) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \sum_{y \in Y} \left\langle x, \theta^y_{\text{binary}} \right\rangle, \tag{4}$$

where $\mathcal{Y}$ is the space of sets of K distinct labels. The above equations describe many of the competitive methods from the ImageNet Challenge, such as Lin et al. (2011) or Sánchez and Perronnin (2011).

One obvious improvement is simply to learn a new set of classifiers $\{\theta^y\}_{y \in \mathcal{C}}$ that optimize the structured error measure of (eq. 1). However, given the large number of classes in the ImageNet Challenge ($|\mathcal{C}| = 1000$), and the high-dimensionality of standard image features, this would mean simultaneously optimizing several million parameters, which is not practical using existing structured learning techniques.

Instead, we would like to leverage the already good classification performance of existing binary classifiers, simply by re-weighting them to account for the structured nature of (eq. 2). Hence we will learn a *single* parameter vector θ that re-weights the features of every class. Our proposed learning framework is designed to extend any classifier of the form given in (eq. 4). Given a set of binary classifiers $\{\theta^y_{\text{binary}}\}_{y \in \mathcal{C}}$, we propose a new classifier of the from

$$\bar{Y}(x; \theta) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \sum_{y \in Y} \left\langle x \odot \theta^y_{\text{binary}}, \theta \right\rangle, \tag{5}$$

where $x \odot \theta^y_{\text{binary}}$ is simply the Hadamard product of x and θ^y_{binary}. Note that when $\theta = \mathbf{1}$ this recovers precisely the original model of (eq. 4).

To use the standard notation of structured prediction, we define the joint feature vector $\Phi(x, Y)$ as

$$\Phi(x, Y) = \sum_{y \in Y} \phi(x, y) = \sum_{y \in Y} x \odot \theta^y_{\text{binary}}, \tag{6}$$

so that (eq. 4) can be expressed as

$$\bar{Y}(x; \theta) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \left\langle \Phi(x, Y), \theta \right\rangle. \tag{7}$$

We will use the shorthand $\bar{Y}^n \coloneqq \bar{Y}(x^n; \theta)$ to avoid excessive notation. In the following sections we shall discuss how structured prediction methods can be used to optimize models of this form.

2.3 The Latent Setting

The joint parametrization of (eq. 6) is problematic, since the energy of the 'true' label y^n, $\langle \phi(x^n, y^n), \theta \rangle$, is not readily comparable with the energy of a *set* of predicted outputs Y, $\langle \Phi(x^n, Y), \theta \rangle$.

To address this, we propose the introduction of a latent variable, $Z = \{Z_1 \dots Z_N\}$, which for each image x^n encodes *the set of objects that appear in x^n that were not annotated*. The full set of labels for the image x^n is now $Y^n = Z_n \cup \{y^n\}$. If our method outputs K objects, then we fix $|Z_n| = K - 1$, so that $|Y^n| = K$. It is now possible to

meaningfully compute the difference between $\Phi(x^n, Y)$ and $\Phi(x^n, Y^n)$, where the latter is defined as

$$\Phi(x^n, Y^n) = \phi(x^n, y^n) + \sum_{y \in Z_n} \phi(x^n, y). \tag{8}$$

The importance of this step shall become clear in Section 3.1, (eq. 13). Note that we still define $\Delta(Y, y^n)$ in terms of the single training label y^n, as in (eq. 2).

Following the programme of Yu and Joachims (2009), learning proceeds by alternately optimizing the latent variables and the parameter vector. Optimizing the parameter vector θ^i given the latent variables Z^i is addressed in Section 3.1; optimizing the latent variables Z^i given the parameter vector θ^{i-1} is addressed in Section 3.2.

3 The Optimization Problem

The optimization problem of (eq. 1) is non-convex. More critically, the loss is a piecewise constant function of θ.[3] A similar problem occurs when one aims to optimize a $0/1$ loss in binary classification; in that case, a typical workaround consists of minimizing a surrogate convex loss function that upper-bounds the $0/1$ loss, such as the hinge loss, which gives rise to support vector machines. We will now see that we can construct a suitable convex relaxation for the problem defined in (eq. 1).

3.1 Convex Relaxation

Here we use an analogous approach to that of SVMs, notably popularized in Tsochantaridis et al. (2005), which optimizes a convex upper bound on the structured loss of (eq. 1). The resulting optimization problem is

$$[\theta^*, \xi^*] = \underset{\theta, \xi}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \|\theta\|^2 \right] \tag{9a}$$

$$\text{s.t. } \langle \Phi(x^n, Y^n), \theta \rangle - \langle \Phi(x^n, Y), \theta \rangle \geq \Delta(Y, Y^n) - \xi_n \tag{9b}$$

$$\forall n, Y \in \mathcal{Y}.$$

It is easy to see that ξ_n^* upper-bounds $\Delta(\bar{Y}^n, y^n)$ (and therefore the objective in (eq. 9) upper bounds that of (eq. 1) for the optimal solution). First note that since the constraints (eq. 9b) hold for all Y, they also hold for $\bar{Y}^n$. Second, the left hand side of the inequality for $Y = \bar{Y}^n$ must be non-positive since $\bar{Y}(x; \theta) = \operatorname{argmax}_Y \langle \Phi(x, Y), \theta \rangle$. It then follows that $\xi_n^* \geq \Delta(\bar{Y}^n, y^n)$. This implies that a solution of the relaxation is an upper bound on the solution of the original problem, and therefore the relaxation is well-motivated.

The constraints (eq. 9b) basically enforce a loss-sensitive margin: θ is learned so that mispredictions Y that incur some loss end up with a score $\langle \Phi(x^n, Y), \theta \rangle$ that is smaller than the score $\langle \Phi(x^n, Y^n), \theta \rangle$ of the correct prediction Y^n by a margin equal to that loss (minus the slack ξ_n). The formulation is a generalization of support vector machines for the multi-class case.

[3] There are countably many values for the loss but uncountably many values for the parameters, so there are large equivalence classes of parameters that correspond to precisely the same loss.

There are two options for solving the convex relaxation of (eq. 9). One is to explicitly include all $N \times |\mathcal{Y}|$ constraints and then solve the resulting quadratic program using one of several existing methods. This may not be feasible if $N \times |\mathcal{Y}|$ is too large. In this case, we can use a constraint generation strategy. This consists of iteratively solving the quadratic program by adding at each iteration the constraint corresponding to the most violated Y for the current model θ and training instance n. This is done by maximizing the violation gap ξ_n, i.e., solving at each iteration the problem

$$\hat{Y}(x^n; \theta) = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \left\{ \Delta(Y, y^n) + \langle \Phi(x^n, Y), \theta \rangle \right\}, \tag{10}$$

(as before we define $\hat{Y}^n := \hat{Y}(x^n; \theta)$ for brevity). The solution to this optimization problem (known as 'column generation') is somewhat involved, though it turns out to be tractable as we shall see in Section 3.3.

Several publicly available tools implement precisely this constraint generation strategy. A popular example is SvmStruct (Tsochantaridis et al., 2005), though we use BMRM ('Bundle Methods for Risk Minimization'; Teo et al., 2007) in light of its faster convergence properties. Algorithm 1 describes pseudocode for solving the optimization problem (eq. 9) with BMRM. In order to use BMRM, one needs to compute, at the optimal solution ξ_n^* for the most violated constraint $\hat{Y}^n$, both the value of the objective function (eq. 9) and its gradient. At the optimal solution for ξ_n^* with fixed θ we have

$$\langle \Phi(x^n, Y^n), \theta \rangle - \left\langle \Phi(x^n, \hat{Y}^n), \theta \right\rangle = \Delta(\hat{Y}^n, y^n) - \xi_n^*. \tag{11}$$

By expressing (eq. 11) as a function of ξ_n^* and substituting into the objective function we obtain the following lower bound on the objective of (eq. 9a):

$$o_i = \frac{1}{N} \sum_n \Delta(\hat{Y}^n, y^n) - \langle \Phi(x^n, Y^n), \theta \rangle + \left\langle \Phi(x^n, \hat{Y}^n), \theta \right\rangle + \lambda \|\theta\|^2, \tag{12}$$

whose gradient with respect to θ is

$$g_i = \lambda\theta + \frac{1}{N} \sum_n (\Phi(x^n, \hat{Y}^n) - \Phi(x^n, Y^n)). \tag{13}$$

3.2 Learning the Latent Variables

To learn the optimal value of θ, we alternate between optimizing the parameter vector θ^i given the latent variables Z^i, and optimizing the latent variables Z^i given the parameter vector θ^{i-1}. Given a fixed parameter vector θ, optimizing the latent variables Z^n can be done greedily, and is in fact equivalent to performing inference, with the restriction that the true label y^n cannot be part of the latent variable Z^n (see Algorithm 2, Line 5). See Yu and Joachims (2009) for further discussion of this type of approach.

Algorithm 1 Taxonomy Learning

1: **Input:** training set $\{(x^n, Y^n)\}_{n=1}^N$
2: **Output:** θ
3: $\theta := \mathbf{0}$ {in the setting of Algorithm 2, θ can be 'hot-started' with its previous value}
4: **repeat**
5: **for** $n \in \{1 \dots N\}$ **do**
6: $\hat{Y}^n := \operatorname{argmax}_{Y \in \mathcal{Y}} \{\Delta(Y, y^n) + \langle \phi(x^n, Y), \theta \rangle\}$
7: **end for**
8: Compute gradient g_i (equation (eq. 13))
9: Compute objective o_i (equation (eq. 12))
10: $\theta := \operatorname{argmin}_\theta \frac{\lambda}{2} \|\theta\|^2 + \max(0, \max_{j \le i} \langle g_j, \theta \rangle + o_j)$
11: **until** converged (see Teo et al. (2007))
12: **return** θ

Algorithm 2 Taxonomy Learning with Latent Variables

1: **Input:** training set $\{(x^n, y^n)\}_{n=1}^N$
2: **Output:** θ
3: $\theta^0 := \mathbf{1}$
4: **for** $i = 1 \dots I$ **do**
5: $Z_i^n := \left\{\operatorname{argmax}_{Y \in \mathcal{Y}} \left\langle \Phi(x^n, Y), \theta^{i-1} \right\rangle\right\} \setminus \{y^n\}$ {choose only $K - 1$ distinct labels}
6: $\theta^i := \text{Algorithm1}\left(\left\{\left(x^n, Z_i^n \cup \{y^n\}\right)\right\}_{n=1}^N\right)$
7: **end for**
8: **return** θ^I

3.3 Column Generation

Given the loss function of (eq. 2), obtaining the most violated constraints (Algorithm 1, Line 6) takes the form

$$\hat{Y}^n = \operatorname*{argmax}_{Y \in \mathcal{Y}} \left\{ \min_{y \in Y} d(y, y^n) + \sum_{y \in Y} \langle \phi(x^n, y), \theta \rangle \right\}, \tag{14}$$

which appears to require enumerating through all $Y \in \mathcal{Y}$, which if there are $C = |\mathcal{C}|$ classes amounts to $\binom{C}{K}$ possibilities. However, if we know that $\operatorname{argmin}_{y \in \hat{Y}^n} d(y, y^n) = c$, then (eq. 14) becomes

$$\hat{Y}^n = \operatorname*{argmax}_{Y \in \mathcal{Y}'} \left\{ d(c, y^n) + \sum_{y \in Y} \langle \phi(x^n, y), \theta \rangle \right\}, \tag{15}$$

where $\mathcal{Y}'$ is just $\mathcal{Y}$ restricted to those y for which $d(y, y^n) \ge d(c, y^n)$. This can be solved greedily by sorting $\langle \phi(x^n, y), \theta \rangle$ for each class $y \in \mathcal{C}$ such that $d(y, y^n) \ge d(c, y^n)$ and simply choosing the top K classes. Since we don't know the optimal value of c in advance, we must consider all $c \in \mathcal{C}$, which means solving (eq. 15) a total of C times. Solving (eq. 15) greedily takes $O(C \log C)$ (sorting C values), so that solving (eq. 14) takes $O(C^2 \log C)$.

Although this method works for any loss of the form given in (eq. 2), for the specific distance function $d(y, y^n)$ used for the ImageNet Challenge, further improvements are possible. As mentioned, for the ImageNet Challenge's hierarchical error measure, $d(y, y^n)$ is the shortest-path distance from y to the nearest common ancestor of y and y^n in a taxonomic tree. One would expect the depth of such a tree to grow logarithmically in the number of classes, and indeed we find that we always have $d(y, y^n) \in \{0 \dots 18\}$. If the number of discrete possibilities for $\Delta(Y, y_n)$ is small, instead of enumerating each possible value of $c = \text{argmin}_{y \in \hat{Y}^n} d(y, y^n)$, we can directly enumerate each value of $\delta = \min_{y \in \hat{Y}^n} d(y, y^n)$. If there are $|L|$ distinct values of the loss, (eq. 14) can now be solved in $O(|L|C \log C)$. In ImageNet we have $|L| = 19$ whereas $C = 1000$, so this is clearly a significant improvement.

Several further improvements can be made (e.g. we do not need to sort all C values in order to compute the top K, and we do not need to re-sort all of them for each value of the loss, etc.). We omit these details for brevity, though our implementation shall be made available at the time of publication.[4]

4 Experiments

4.1 Binary Classifiers

As previously described, our approach needs, for each class, one binary classifier able to provide some reasonable score as a starting point for the proposed method. Since the objective of this paper is not beating the state-of-the-art, but rather demonstrating the advantage of our structured learning approach to improve the overall classification, we used a standard, simple image classification setup. As mentioned, should the one-vs-all classifiers of Lin et al. (2011) or Sánchez and Perronnin (2011) become available in the future, they should be immediately compatible with the proposed method.

First, images have to be transformed into descriptor vectors sensible for classification using machine learning techniques. For this we have chosen the very popular Bag of Features model (Csurka et al., 2004): dense SIFT features are extracted from each image x^n and quantized using a visual vocabulary of F visual words. Next, the visual words are pooled in a histogram that represents the image. This representation is widely used in state-of-the-art image classification methods, and despite its simplicity achieves very good results.

Regarding the basic classifiers, a rational first choice would be to use a Linear SVM for every class. However, since our objective is to predict the correct class of a new image, we would need to compare the raw scores attained by the classifier, which would not be theoretically satisfying. Although it is possible to obtain probabilities from SVM scores using a sigmoid trained with the Platt algorithm, we opted for training Logistic Regressors instead, which directly give probabilities as output and do not depend on a separate validation set.

In order to deal with the computational and memory requirements derived from the large number of training images, we used Stochastic Gradient Descent (SGD) from

[4] see http://users.cecs.anu.edu.au/~julianm/

Bottou and Bousquet (2008) to train the classifiers. SGD is a good choice for our problem, since it has been shown to achieve a performance similar to that of batch training methods in a fraction of the time (Perronnin et al., 2010). Furthermore, we validated its performance against that of LibLinear in a small-scale experiment using part of the ImageNet hierarchy with satisfactory results. One limitation of online learning methods is that the optimization process iterations are limited by the amount of training data available. In order to add more training data, we cycled over all the training data for 10 epochs.

With this approach, the θ^y_{binary} parameters for each class used in the structured learning method proposed in this work were generated.

4.2 Structured Classifiers

For every image x^n and every class y we must compute $\langle \phi(x^n, y), \theta \rangle$. Earlier we defined $\phi(x, y) = x \odot \theta^y_{\text{binary}}$. If we have C classes and F features, then this computation can be made efficient by first computing the $C \times F$ matrix A whose y^{th} row is given by $\theta^y_{\text{binary}} \odot \theta$. Similarly, if we have N images then the set of image features can be thought of as an $N \times F$ matrix X. Now the energy of a particular labeling y of x^n under θ is given by the matrix product

$$\langle \phi(x^n, y), \theta \rangle = \left(X \times A^T \right)_{n,y}. \tag{16}$$

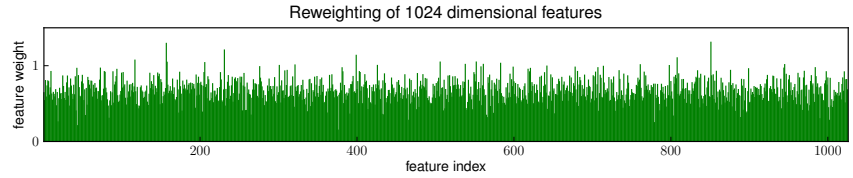
This observation is critical if we wish to handle a large number of images and high-dimensional feature vectors. In our experiments, we performed this computation using Nvidia's high-performance BLAS library CUBLAS. Although GPU performance is often limited by a memory bottleneck, this particular application is ideally suited as the matrix X is far larger than either the matrix A, or the resulting product, and X needs to be copied to the GPU only once, after which it is repeatedly reused. After this matrix product is computed, we must sort every row, which can be naïvely parallelized.

In light of these observations, our method is no longer prohibitively constrained by its running time (running ten iterations of Algorithm 2 takes around one day for a single regularization parameter λ). Instead we are constrained by the size of the GPU's onboard memory, meaning that we only used 25% of the training data (half for training, half for validation). In principle the method could be further parallelized across multiple machines, using a parallel implementation of the BMRM library.

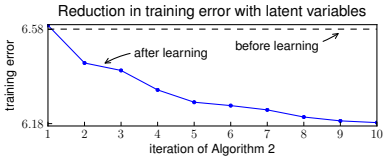
The results of our algorithm using features of dimension $F = 1024$ and $F = 4096$ are shown in Figures 2 and 3, respectively. Here we ran Algorithm 2 for ten iterations, 'hot-starting' θ^i using the optimal result from the previous iteration. The reduction in training error is also shown during subsequent iterations of Algorithm 2, showing that minimal benefits are gained after ten iterations. We used regularization parameters $\lambda \in \{10^{-1}, 10^{-2} \ldots 10^{-8}\}$, and as usual we report the test error for the value of λ that resulted in the best performance on the validation set. We show the test error for different numbers of nearest-neighbors K, though the method was trained to minimize the error for $K = 5$.

In both Figures 2 and 3, we find that the optimal θ is non-uniform, indicating that there are interesting relationships that can be learned between the features when a structured setting is used. As hoped, a reduction in test error is obtained over already good

(a)



(b)



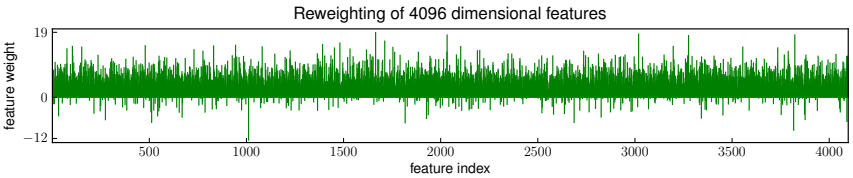
(c)

Test error:

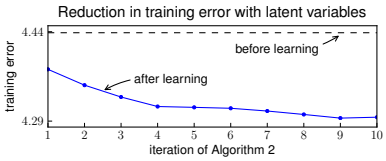
	1nn	2	3	4	5
Before learning	11.35	9.29	8.08	7.25	6.64
After learning	10.88	8.85	7.71	6.93	6.36

Fig. 2. Results for training with 1024 dimensional features. (a) feature weights; (b) reduction in training error during each iteration of Algorithm 2; (c) error for different numbers of nearest-neighbors K (the method was trained to optimize the error for $K = 5$). Results are reported for the best value of λ on the validation set (here $\lambda = 10^{-4}$).

(a)



(b)



(c)

Test error:

	1nn	2	3	4	5
Before learning	9.27	7.29	6.23	5.53	5.03
After learning	9.02	7.08	6.05	5.38	4.91

Fig. 3. Results for training with 4096 dimensional features. (a) feature weights; (b) reduction in training error during each iteration of Algorithm 2; (c) error for different numbers of nearest-neighbors K (the method was trained to optimize the error for $K = 5$). Results are reported for the best value of λ on the validation set (here $\lambda = 10^{-6}$).

classifiers, though the improvement is indeed less significant for the better-performing high-dimensional classifiers.

In the future we hope to apply our method to state-of-the-art features and classifiers like those of Lin et al. (2011) or Sánchez and Perronnin (2011). It remains to be seen whether the setting we have described could yield additional benefits over their already excellent classifiers.

5 Conclusion

Large scale, collaboratively labeled image datasets embedded in a taxonomy naturally invite the use of *both* structured *and* robust losses, to account for the inconsistencies in the labeling process and the hierarchical structure of the taxonomy. However, on datasets such as ImageNet, the state-of-the-art methods still use one-vs-all classifiers,

which do not account for the structured nature of such losses, nor for the imperfect nature of the annotation. We have outlined the computational challenges involved in using structured methods, which sheds some light on why they have not been used before in this task. However, by exploiting a number of computational tricks, and by using recent advances on structured learning with latent variables, we have been able to formulate learning in this task as the optimization of a loss that is both structured and robust to weak labeling. Better yet, our method leverages existing one-vs-all classifiers, essentially by re-weighting, or 'boosting' them to directly account for the structured loss. In practice this leads to improvements in the hierarchical loss of already good one-vs-all classifiers.

Acknowledgements

Part of this work was carried out when both AR and TC were at INRIA Grenoble, Rhône-Alpes. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. This work was partially funded by the QUAERO project supported by OSEO, French State agency for innovation and by MICINN under project MIPRCV Consolider Ingenio CSD2007-00018.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1, 2

Alex Berg, Jia Deng, and Fei-Fei Li. Imagenet large scale visual recognition challenge 2010. http://www.image-net.org/challenges/LSVRC/2010/index, 2010. 1, 2, 4

Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, and Kai Yu. Large-scale image classification: fast feature extraction and SVM training. In *IEEE Conference on Computer Vision and Pattern Recognition*, page (to appear), 2011. 1, 2, 3, 4, 6, 10, 12

Jorge Sánchez and Florent Perronnin. High-Dimensional Signature Compression for Large-Scale Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, page (to appear), 2011. 1, 3, 4, 6, 10, 12

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. *European Conference on Computer Vision*, pages 143–156, 2010. 2, 11

Antonio Torralba, Rob Fergus, and William T Freeman. 80 Million Tiny Images: a Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–70, 2008. 2

Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, 2010. 2

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and Vishy Vishwanathan. Hash Kernels. In *Artificial Intelligence and Statistics*, 2009. 3

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010. 3

Olga Russakovsky and L. Fei-Fei. Attribute learning in large-scale datasets. In *ECCV Workshop on Parts and Attributes*, 2010. 3

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. 4, 7, 8

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 4

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 4

Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning*, 2009. 7, 8

Choon Hui Teo, Alex Smola, S. V.N. Vishwanathan, and Quoc Viet Le. A scalable modular convex solver for regularized risk minimization. In *Knowledge Discovery and Data Mining*, 2007. 8, 9

Gabriela Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on statistical learning in computer vision*, 2004. 10

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, 2008. 11