Large-scale image classification with trace-norm regularization

Zaid Harchaoui, Matthijs Douze, Mattis Paulin, Miro Dudik, Jérôme Malick

Large-scale image classification with trace-norm regularization

Zaid Harchaoui
INRIA & LJK

Matthijs Douze
INRIA

Mattis Paulin
INRIA

Miroslav Dudik
Yahoo! Research

Jérôme Malick
CNRS & LJK

Abstract

With the advent of larger image classification datasets such as ImageNet, designing scalable and efficient multi-class classification algorithms is now an important challenge. We introduce a new scalable learning algorithm for large-scale multi-class image classification, based on the multinomial logistic loss and the trace-norm regularization penalty. Reframing the challenging non-smooth optimization problem into a surrogate infinite-dimensional optimization problem with a regular ℓ_1-regularization penalty, we propose a simple and provably efficient accelerated coordinate descent algorithm. Furthermore, we show how to perform efficient matrix computations in the compressed domain for quantized dense visual features, scaling up to 100,000s examples, 1,000s-dimensional features, and 100s of categories. Promising experimental results on the "Fungus", "Ungulate", and "Vehicles" subsets of ImageNet are presented, where we show that our approach performs significantly better than state-of-the-art approaches for Fisher vectors with 16 Gaussians.

1. Introduction

Large-scale image classification is becoming a central challenge in computer vision, with ever more and ever larger computer vision datasets being available. The ImageNet dataset (www.image-net.org) along with the associated challenges is a representative example of this trend, with more than 14 million images labelled with almost 22,000 concepts [9].

Current state-of-the-art methods for large-scale image classification [18, 25] use high-dimensional image descriptors in combination with linear classifiers. Linear classifiers are attractive due to their computational efficiency. Expressive feature representation is then performed using so-called explicit embeddings [29]. Popular and efficient visual features include the low dimensional bag-of-visual-words (BOV) [5], Fisher vectors [22, 23], local coordinate coding [30] and supervector coding [31].

Most image categorization approaches rely on one-vs-rest strategies for training multi-class classifiers. Such strategies are clearly attractive for the following computational reasons: i) simple underlying learning algorithms, ii) simple way to cross-validate hyper-parameters, iii) scalability to large-scale settings. As an example, the winning approaches of 2010 edition of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[1] were built upon one-vs-rest classifiers [18, 25]. However, one-vs-rest strategies suffer from weak theoretical guarantees. In particular, there exists a joint distribution over the examples and labels such that, even for large number of examples, one-vs-rest is not statistically consistent [3].

On the other hand, multi-class classifiers based on loss functions such as the multinomial logistic loss enjoy attractive theoretical guarantees such as universal consistency [3]. Yet, up to our knowledge, there are still no publicly available packages implementing a scalable approach for multi-class classification up to scales as in the ImageNet dataset. In other words, it is currently computationally impractical to compare the performance of multi-class approaches to one-vs-rest approaches on large datasets such as ImageNet.

We propose a new scalable approach for multi-class classification, based on the multinomial logistic loss and the trace-norm regularization penalty. The main contributions of this work are the following:

- we prove that one can reframe the challenging non-smooth optimization problem into a surrogate infinite-dimensional optimization problem with a regular ℓ_1-regularizer.
- we propose a simple and efficient coordinate descent algorithm with theoretical convergence guarantees on the original objective.
- we show how to perform efficient matrix computations in the compressed domain for quantized dense features, to scale our algorithm to 100,000s examples, 1,000s-dimensional features, and 100s of categories.
- we show that our approach performs significantly better than state-of-the-art approaches for Fisher vectors with 16 Gaussians on the "Fungus", "Ungulate", and "Vehicles" datasets from ImageNet.

[1] http://www.image-net.org/challenges/LSVRC/2010

2. Related work

Most state-of-the-art approaches for large-scale image classification that scale up to datasets as large as ImageNet are based on large margin classifiers, *i.e.*, based on performing a constrained/penalized empirical risk minimization.

The most popular approach for dealing with large number of categories is the one-vs-rest approach [24]. Besides their built-in generalization ability, a major advantage of one-vs-rest (OVR) strategies is their linear scalability in the number of categories, and their high flexibility for tuning hyperparameters. The downside is their lack of theoretical guarantees [27]. They can indeed provably fail for certain distributions of the data even for large samples. Other theoretically-grounded large margin approaches have however been proposed, such as the Crammer and Singer variant of the multi-class SVM [7] or L_2-regularized multi-class logistic regression [14]. Yet, up to our knowledge, we are not aware of any publicly available implementation of such approaches that scale to datasets as large as ImageNet in reasonable time.

Several learning algorithms with different regularization penalties were previously proposed [1, 10, 26]. The authors in [10] develop a boosting approach in an additive logistic regression style, *i.e.*, with a multinomial logistic regression, assuming a group sparsity pattern over features. It is unclear how to use such approaches for state-of-the-art features such as Fisher vectors which are typically dense. In [26], an elegant boosting approach is presented to learn shared features in multi-class classification. Again, group sparsity-patterns are assumed and do not look appropriate for image classification tasks. Finally, in [1], a variant of SVMs with a trace-norm regularization penalty is proposed, with an algorithm based on a gradient descent on the smoothed objective, with applications to a small-scale image dataset (the Mammal dataset). Because of the smoothing of the objective, the algorithm does not explicitly leverage the low-rank structure of the problem and is therefore not directly amenable to large-scale situations.

Stochastic training algorithm have recently re-gained popularity for large-scale applications. These algorithms rely on stochastic gradient descent [17, 4]. While such strategies are clearly advantageous for large-scale situations, devising SGD schemes for regularization penalties such as the trace-norm is not straightforward (it would require one full SVD per example). The focus of this paper is on a scalable training algorithm for multi-class classification with trace-norm regularization for large-scale image classification. Batch learning offers several advantages [17]: well-understood convergence properties and easier to incorporate acceleration techniques. We propose here a new algorithm with guaranteed convergence properties, with a continuation-style technique allowing to quickly scan the performance along regularization path.

3. Regularization through trace-norm penalty

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ be a set of n training images with visual features of size d and labelled into one of the k classes. Multi-class classification consists in learning a multi-class classifier $g(\mathbf{x}) = \operatorname{Arg\,max}_{\ell=1,\ldots,n} \mathbf{w}_\ell^T \mathbf{x}$ allowing to predict a class-label for a new image $\mathbf{x}$. The classifier is specified by a separate weight vector $\mathbf{w}_\ell \in \mathbb{R}^d$ for each class. Class-wise weight vectors form the weight matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{d \times k}$. Training of classifiers is performed through regularized empirical risk minimization

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \lambda \Omega(\mathbf{W}) + \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{W}; \mathbf{x}_i, y_i) \,, \qquad (1)$$

where $\Omega(\mathbf{W})$ denotes the regularization penalty and $L(\mathbf{W}; \mathbf{x}_i, y_i)$ the loss function. For example, in the so-called "Crammer and Singer" multi-class support vector machine [7], $\Omega(\mathbf{W}) = \sum_{\ell=1}^{k} \|\mathbf{w}_\ell\|^2$ and

$$L(\mathbf{W}; \mathbf{x}_i, y_i) = \max_{\ell=1,\ldots,k} \left\{ \delta(y_i, y_\ell) + (\mathbf{w}_{y_i} - \mathbf{w}_\ell)^T \mathbf{x}_i \right\} ,$$

where the function δ equals 1 when the two arguments are equal, and 0 otherwise. Other formulations of support vector machines were proposed, each with a different loss function. Only some of them, including the above mentioned one, satisfy a *universal consistency* property [27]. For instance, the popular one-vs-rest (OVR) strategy [24] does not enjoy strong theoretical guarantees.

Towards a low-rank enforcing penalty The approaches just mentioned usually lead to satisfactory performance in a variety of situations. Yet, these approaches are not tailored to deal with situations with a large number of categories. Indeed, with the exception of OVR strategies, we are not aware of any publicly available software implementing a "true" multi-class approach that scales to *large number of examples*, *large feature size* and *large number of categories*. Our goal here is to attempt to fill this gap, taking OVR strategies as a baseline for large-scale multi-class classification learning algorithms.

Furthermore, datasets with a large number of categories usually exhibit *low-rank embedding of the classes* behaviour, *i.e.*, classes can be embedded in a lower-dimensional subspace than the ambient space. There are many ways to motivate this assumption [1, 2]. Here, we motivate and empirically check this assumption by looking at the weight vectors *learned from the data* by one-vs-rest classifiers on a real-world image categorization dataset. We consider the "Ungulate" dataset from the ImageNet large dataset, taking 4096-dimensional descriptors. We ran a collection of one-vs-rest classifiers and performed a cross-validation on the different λ parameters. Then we formed

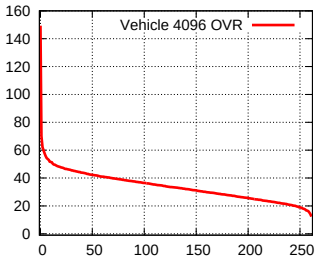


Figure 1. Spectrum of the classification weight matrix $\mathbf{W}$ learned on the "Vehicle" dataset

the weight matrix $\mathbf{W}$ and computed a singular value decomposition (SVD) to get the spectrum of the matrix. Note that the weight matrix $\mathbf{W}$ was *learned from the training data* and we therefore can infer structure about the categories using the information gathered by an algorithm yielding good performance. This way, we are able to check the low-rank assumption *a posteriori*. The spectrum plotted in Figure 1 exhibits an *exponential decay*, *i.e.*, while the first singular values are large the next ones are much smaller and remain at a similar magnitude. Most of the energy is concentrated in the first singular values. One can interpret this phenomenon looking at either dimensions of the weight matrix $\mathbf{W}$: i) the feature dimension, of size d, ii) the class dimension, of size k. The exponential decay implies that the examples could be represented in a subspace of dimension lower than d, and still yield similar performance when fed into a learning algorithm. On the other hand, it also implies that the classes/categories could be represented through linear combinations of a set of underlying prototypes classes of size lower than k.

We propose to leverage such behaviour by tailoring the learning objective so that the weight matrix captures the low-rank embedding of the classes. Hence we propose to add a low-rank enforcing regularization penalty, the trace-norm regularization penalty, to the usual Frobenius-norm $\|\mathbf{W}\|_2^2$ regularization penalty. Along with a universally consistent multi-class loss function, we get a learning algorithm for which we propose scalable algorithm suitable for tackling large datasets such as ImageNet.

Learning with a low-rank enforcing regularizer We describe the core idea of our large-scale algorithm for multi-class classification with a low-rank enforcing regularization penalty. For the sake of clarity, we shall focus in this paper on the multinomial logistic loss [14] as a multi-class loss

$$L(\mathbf{W}; \mathbf{x}, y) = \log\Big(1 + \sum_{\ell \in \mathcal{Y} \setminus \{y\}} \exp\left\{\mathbf{w}_\ell^T \mathbf{x} - \mathbf{w}_y^T \mathbf{x}\right\}\Big).$$

Indeed, this loss yields "for free" probability estimates for the prediction. As opposed to support vector machines, working with multi-class hinge loss, there is no need to fit *a posteriori* a logistic function to the outputs of the classifier to get probability estimates.

Therefore, we focus here on learning objectives for the purpose of multi-class classification which write as

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \lambda_1 \operatorname{rank}(\mathbf{W}) + \lambda_2 \|\mathbf{W}\|_2^2 + R_n(\mathbf{W}) \,. \quad (2)$$

where $R_n(\mathbf{W})$ denotes the empirical risk

$$R_n(\mathbf{W}) := \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{W}; \mathbf{x}_i, y_i) \,. \quad (3)$$

Such an objective yields a *non-smooth non-convex* optimization problem. Solving it directly would lead to unstable algorithms requiring a large number of multiple restarts to slowly achieve good performance. Similar to [1, 2], we propose to relax the objective by replacing $\operatorname{rank}(\mathbf{W})$ by its tightest convex surrogate [13]: the trace-norm.

From the rank to the trace-norm For $\mathbf{W} \in \mathbb{R}^{d \times k}$, denote by $\sigma(\mathbf{W})$ the singular spectrum of the matrix, viewed as a vector of its singular values, and define the norms

$$\|\mathbf{W}\|_{\sigma,p} := \|\sigma(\mathbf{W})\|_p \,.$$

Taking $p = \infty$ leads to the so-called spectral norm. Now, taking $p = 1$, we obtain the so-called trace-norm of the matrix, also called nuclear norm. Note that for a positive-definite matrix $\mathbf{W}$, $\|\mathbf{W}\|_{\sigma,1}$ equals the trace of $\mathbf{W}$, hence the name trace-norm. Feeding the trace-norm $\|\mathbf{W}\|_{\sigma,1}$ back into the learning objective in place of $\operatorname{rank}(\mathbf{W})$, we get the following objective which we seek to minimize:

$$J(\mathbf{W}) := \lambda_1 \|\mathbf{W}\|_{\sigma,1} + \lambda_2 \|\mathbf{W}\|_2^2 + R_n(\mathbf{W}) \quad (4)$$

The learning objective in Eq. 4 is now convex. Our regularization penalty is a combination of a trace-norm penalty and a regular Frobenius-norm penalty. It can be seen as matrix-counterpart of the so-called "elastic net" penalty for vectors [32]. Note that, in contrast to [1, 2], our objective is *strongly convex*, owing to the Frobenius-norm part of the regularization penalty. Strong convexity yields more stable optimization. Furthermore, in our experiments, it proved helpful for very high-dimensional problems to get competitive performance; see Section 5.

Although now convex, the learning objective in Eq. 4 is still challenging, since it yields a non-differentiable optimization problem. The low-rank enforcing property of the trace-norm is an advantage in large-scale settings, so we want to keep it in our objective as is (instead of smoothing it, which would cancel its low-rank enforcing property).

The learning objective is strikingly similar to sparse logistic regression for binary classification problems [14], where sparsity is enforced on the weight vector with a (vector) ℓ_1-norm. Here, sparsity is enforced in the spectral

domain of the weight matrix $\mathbf{W}$. The trace-norm can be thought as a natural generalization of the ℓ_1-norm to matrices with the coordinates replaced by the rank-one matrices made by singular vectors of $\mathbf{W}$.

Coordinate descent algorithms are remarkably efficient for sparse logistic regression, and to optimize ℓ_1-norm penalized objectives in large-scale settings in general. Hence, a coordinate descent-like approach appears to be the method of choice here, if one could apply a coordinate descent algorithm in the spectral domain of $\mathbf{W}$. However, the challenge is determining *which* rank-one matrices should come into play. Ideally, we would use the rank-one matrices obtained from the SVD of the minimizer $\mathbf{W}^\star$ of Eq. 4 as a "basis", and the weights of the decomposition of the matrix onto this basis as *coordinates*. However, except for simple least-squares regression problems, it is not clear how to determine these matrices without first finding the solution.

Reframing as an ℓ_1-regularization The intuition behind the above remarks can be formalized to get a convergent and efficient optimization algorithm. The idea is to look at $\mathbf{W}$ through its "atomic decomposition" onto a particular dictionary of matrices. Atomic decompositions and pursuit algorithms have been popular in signal processing and harmonic analysis [6, 20]. As we show here, the trace-norm corresponds the infimal ℓ_1-norm on the weights of atomic decompositions of $\mathbf{W}$ onto a particular dictionary of atoms.

We first consider the *overcomplete and uncountable infinite dictionary* of *all possible* (normalized) "atoms" corresponding to rank-one matrices. Denote $\mathcal{M}$ the set of rank-one matrices

$$\mathcal{M} = \{\mathbf{u}\mathbf{v}^T \mid \mathbf{u} \in \mathbb{R}^d,\, \mathbf{v} \in \mathbb{R}^k,\, \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1\} \,.$$

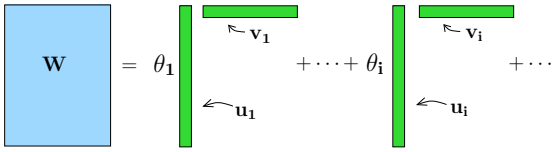
Note that we do not enforce that the set $\mathcal{M}$ forms a basis. Let us consider $\mathscr{I}$, the index set spanning the "atoms" in $\mathcal{M}$. We can then write $\mathcal{M} = \{\mathbf{M}_i \in \mathbb{R}^{d\times k} \mid i \in \mathscr{I}\} = \{\mathbf{u}_i\mathbf{v}_i^T \mid i \in \mathscr{I}\}$.

For $\boldsymbol{\theta} \in \mathbb{R}^{\mathscr{I}}$, denote $\mathrm{supp}(\boldsymbol{\theta}) := \{j,\, \theta_j \neq 0\}$ the support of $\boldsymbol{\theta}$. We consider Θ defined as $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^{\mathscr{I}} \mid \mathrm{supp}(\boldsymbol{\theta})$ is finite$\}$, equipped with the natural ℓ_1-norm defined by $\|\boldsymbol{\theta}\|_1 = \sum_{i\in\mathrm{supp}(\boldsymbol{\theta})} |\theta_i|$. We can now decompose the matrix $\mathbf{W}$ onto atoms in $\mathcal{M}$ as follows:

$$\mathbf{W} = \sum_{i\in\mathrm{supp}(\boldsymbol{\theta})} \theta_i \mathbf{M}_i \,. \tag{5}$$

The above equation defines *an atomic decomposition* of $\mathbf{W}$ onto a series of "atoms" (rank-one matrices). Such a decomposition is not unique, since $\mathcal{M}$ is an overcomplete dictionary and not a basis.

One can look at the trace-norm as a regular ℓ_1-norm on the weights of the atomic decomposition of $\mathbf{W}$, as shown by the next proposition.



Proposition 1. *The trace-norm of* $\mathbf{W}$ *can be expressed as*

$$\|\mathbf{W}\|_{\sigma,1} = \min\left\{ \|\boldsymbol{\theta}\|_1 \mid \exists \mathbf{M}_i,\ s.t.\ \mathbf{W} = \sum_{i=1}^N \theta_i \mathbf{M}_i \right\}.$$

Proof. Consider the singular value decomposition of $\mathbf{W}$, written as $\mathbf{W} = \sum_{i=1}^{\mathrm{rank}(\mathbf{W})} \theta_i \mathbf{u}_i \mathbf{v}_i^T$. Note that $\mathbf{M}_i = \mathbf{u}_i\mathbf{v}_i^T \in \mathcal{M}$, hence

$$\begin{aligned}\|\mathbf{W}\|_{\sigma,1} &= \sum_{i=1}^{\mathrm{rank}(\mathbf{W})} \theta_i \\ &\geq \min\left\{ \|\boldsymbol{\theta}\|_1 \mid \exists \mathbf{M}_i,\ \text{s.t. } \mathbf{W} = \sum_{i=1}^N \theta_i \mathbf{M}_i \right\} \,,\end{aligned}$$

i.e., the left-hand side is upper-bounded by the right-hand side.

To prove the opposite inequality, consider an arbitrary decomposition of the form $\mathbf{W} = \sum_{i=1}^N \theta_i \mathbf{M}_i$ with $\mathbf{M}_i \in \mathcal{M}$. Now, by triangular inequality and positive homogeneity of the norm $\|\cdot\|_{\sigma,1}$, we have :

$$\begin{aligned}\|\mathbf{W}\|_{\sigma,1} &= \left\| \sum_{i=1}^N \theta_i \mathbf{M}_i \right\|_{\sigma,1} \\ &\leq \sum_{i=1}^N |\theta_i|\ \|\mathbf{M}_i\|_{\sigma,1} = \sum_{i=1}^N |\theta_i| = \|\boldsymbol{\theta}\|_1 \,,\end{aligned}$$

completing the proof. □

Let us rephrase our problem in terms of the new variable $\boldsymbol{\theta}$, corresponding to the weights of the atomic decomposition of $\mathbf{W}$ as $\mathbf{W}_{\boldsymbol{\theta}} = \sum_{i\in\mathrm{supp}(\boldsymbol{\theta})} \theta_i \mathbf{M}_i$. Consider

$$\tilde{R}_n(\boldsymbol{\theta}) := \lambda_2 \|\mathbf{W}_{\boldsymbol{\theta}}\|_2^2 + R_n(\mathbf{W}_{\boldsymbol{\theta}}) \,, \tag{6}$$

i.e., the infinite dimensional version of the empirical risk R_n with the Frobenius-norm regularization penalty. We get a reframed optimization problem

$$\min_{\boldsymbol{\theta}\in\Theta^+} I(\boldsymbol{\theta}) := \lambda \sum_{j\in\mathrm{supp}(\boldsymbol{\theta})} \theta_j + \tilde{R}_n(\mathbf{W}_{\boldsymbol{\theta}}) \,. \tag{7}$$

We can prove that this optimization problem shares the same minimum as the original one [11]. We now work on the new smooth objective, instead of working on the non-smooth original objective.

4. Large-scale algorithm

We can now devise a coordinate descent algorithm to minimize the objective in Eq. 7. The algorithm belongs to the general family of AtomDescent algorithms [11, Supplement]. Therefore, our algorithm comes with theoretical guarantees; in particular, it is provably convergent. For any $\varepsilon > 0$, the stopping criterion is given by ε-approximate optimality defined by

$$\text{(C1)} \quad \forall i \in \mathscr{I} : \frac{\partial \tilde{R}_n(\boldsymbol{\theta})}{\partial \theta_i} \geq -\lambda_1 - \varepsilon\,,$$

$$\text{(C2)} \quad \forall i \in \operatorname{supp}(\boldsymbol{\theta}) : \left| \frac{\partial \tilde{R}_n(\boldsymbol{\theta})}{\partial \theta_i} + \lambda_1 \right| \leq \varepsilon\,.$$

We describe below the main blocks of our algorithm **R1D** (Algorithm 4.1), which stands for rank-one descent as it will become clear below.

4.1. Coordinate descent scheme

At the current iterate $\boldsymbol{\theta}_t$, we pick the coordinate along which we can achieve the steepest descent while remaining in Θ^+. We pick $i \in \mathscr{I}$ with the largest $-\partial I(\boldsymbol{\theta}_t)/\partial \theta_i$, that is the smallest partial derivative of I with respect to the i-th coordinate in the $\boldsymbol{\theta}$ representation.

Descent direction In fact, the coordinate corresponding to largest $-\partial I(\boldsymbol{\theta}_t)/\partial \theta_i$, is held by top singular-vector pair of the matrix $-\nabla \tilde{R}_n(\mathbf{W}_{\boldsymbol{\theta}_t})$:

$$\begin{aligned} \operatorname*{Arg\,min}_{i \in \mathscr{I}} \frac{\partial I(\boldsymbol{\theta}_t)}{\partial \theta_i} &= \operatorname*{Arg\,max}_{\substack{i \in \mathscr{I} \\ \|\mathbf{u}_i\|_2 = \|\mathbf{v}_i\|_2 = 1}} \mathbf{u}_i^T \left(-\nabla \tilde{R}_n(\mathbf{W}_{\boldsymbol{\theta}_t})\right) \mathbf{v}_i \\ &= \operatorname*{Arg\,max}_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1} \mathbf{u}^T \left(-\nabla \tilde{R}_n(\mathbf{W}_{\boldsymbol{\theta}_t})\right) \mathbf{v}\,. \end{aligned}$$

Recall here that $\tilde{R}_n(\mathbf{W})$ is the L_2^2-norm regularized empirical risk defined in Eq. 6. For coordinates $i \notin \operatorname{supp}(\boldsymbol{\theta}_t)$, we can only move in the positive direction. For coordinates $i \in \operatorname{supp}(\boldsymbol{\theta})$, it is also possible to move in the negative direction. In our algorithm, we do not compute the steepest direction $-\partial I(\boldsymbol{\theta}_t)/\partial \theta_i$, but only use a steep-enough direction (steepest up to $\varepsilon/2$). The algorithm works by iteratively making rank-one updates along these directions and performing second-order subspace optimization on a regular basis.

Stepsize selection At each iteration of our algorithm, we need to perform a descent step along the direction given by the rank-one subspace $\mathbf{u}_t \mathbf{v}_t^T$. Note that the direction $\mathbf{u}_t \mathbf{v}_t^T$ is not a steepest-descent direction. We need to compute a stepsize so that we are guaranteed to decrease the value of the objective $I(\boldsymbol{\theta})$ at each iteration. Hence, at iteration t, we perform an Armijo-style line-search along the direction $\mathbf{u}_t \mathbf{v}_t^T$ to find the stepsize δ. For this, we used a similar rule as the one proposed by [28] for ℓ_1-regularized objectives. We make the update $\mathbf{W}_{t+1} = \mathbf{W}_t + \delta\, \mathbf{u}_t \mathbf{v}_t^T$ and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta\, \mathbf{e}_t$, where $\{\boldsymbol{\theta}_t\}_{t \geq 1}$ should be understood as a list that keeps track of the weights of the atomic decomposition along the iterations and $\mathbf{e}_t = (0, \ldots, 0, 1, 0, \ldots)$ is an infinite-length vector with a 1 at the t-th position.

Input: regularization parameters λ_1 and λ_2
initial point $\mathbf{W}_{\boldsymbol{\theta}_0}$, convergence threshold ε

Output: ε-optimal $\mathbf{W}_{\boldsymbol{\theta}}$

Algorithm:

For $t = 0, 1, 2, \ldots$:

1. Compute (approximately) top singular vector pair $\{\mathbf{u}_t, \mathbf{v}_t\}$ of $-\nabla \tilde{R}_n(\mathbf{W}_t)$

   $$\mathbf{u}_t^T \left(-\nabla \tilde{R}_n(\mathbf{W}_t)\right) \mathbf{v}_t \geq \left\| \nabla \tilde{R}_n(\mathbf{W}_t) \right\|_{\sigma,\infty} - \varepsilon/2$$

2. Let $g_t = \lambda_1 + \langle \nabla \tilde{R}_n(\mathbf{W}_t), \mathbf{u}_t \mathbf{v}_t^T \rangle$
3. If $g_t \leq -\varepsilon/2$

   $\mathbf{W}_{t+1} = \mathbf{W}_t + \delta\, \mathbf{u}_t \mathbf{v}_t^T$ with δ found by line-search
   $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta\, \mathbf{e}_t$

4. Else (i.e., $g_t > -\varepsilon/2$)

   If $\boldsymbol{\theta}_t$ satisfies stopping cond., stop and return $\boldsymbol{\theta}_t$

   Otherwise, compute $\boldsymbol{\theta}_{t+1}$ an ε-solution of the restricted problem (8)

Subspace acceleration On a regular basis, we perform a second-order subspace minimization on the weights obtained so far to accelerate the convergence speed of our algorithm. Let $s = \operatorname{supp}(\boldsymbol{\theta})$ be the size of the support of $\boldsymbol{\theta}$ at iteration t, and let us reorder the coordinate of $\boldsymbol{\theta}$ using $j = 1, \ldots, s$. We solve

$$\begin{cases} \min\limits_{\theta_1, \ldots, \theta_s} & \lambda_1 \sum_{j=1}^s \theta_j + \tilde{R}_n \left(\sum_{j=1}^s \theta_j \mathbf{u}_j \mathbf{v}_j^\top \right) \\ \text{subject to} & \theta_j \geq 0\,, \quad j = 1, \ldots, s \end{cases} \tag{8}$$

This is a convex and smooth objective with simple box constraints. We propose to use a state-of-the-art optimization algorithm to solve it, namely a Quasi-Newton algorithm with box constraints such as L-BFGS-B [21]. This algorithm only requires objective and gradient evaluations with respect to the finite-dimensional $\boldsymbol{\theta}$.

Regularization path It has been often noted that solving ℓ_1-regularized problems is faster when λ_1 is large, see eg. [14]. We use this observation for our algorithm. We take a sequence of problems with decreasing values of λ_1, and use the intermediate solution as a warm start for the next problem. We follow a similar strategy for the parameter λ_2. In addition to the benefit from the warm-starting,

we obtain a sequence of models optimizing the same empirical risk with different values of regularization. We therefore approximately follow the two-dimensional *regularization path* [14] hence quickly perform cross-validation of the parameters λ_1 and λ_2. In the end, we get regularization path for geometrically spaced sequence of λ_1-s and λ_2-s of the form $\lambda_\ell = \lambda_0 \alpha^\ell$ where $\alpha \in (0, 1)$.

As any convex optimization algorithm, our algorihm involves matrix-vector multiplications as elementary operations. In large-scale settings, these operations also need to be performed quickly. We show how in the next section.

4.2. Computation in the compressed domain

Training a multi-class classifier with a large number of classes does not raise any major issue in terms of matrix computations when the full data matrix fits into RAM memory. For our experiments, the data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ is about $d \times n = 65536 \times 250000$ elements, occupying 65 GB.

Many software packages, such as Vowpal Wabbit [16], liblinear [12], etc. efficiently deal with large datasets by leveraging the sparsity of the feature vectors. However, the feature vectors such as Fisher vectors are definitely not sparse, with less than 45% zeroes on average. Therefore, matrix computations must be optimized for our training algorithm to efficiently handle dense data matrices.

Training in the compressed domain, testing without compression We propose to work on compressed data matrices. In [25], feature vectors were compressed through *product quantization* [15], for the purpose of nearest-neighbor image retrieval. One can also use this scheme for decompression, *i.e.*, for reconstructing the original vectors from the compressed ones, in constrast to, *e.g.*, Locality Sensitive Hashing.

We propose an efficient strategy for our approach, which consists in *training with compressed data, testing without compression*. As also observed in [25] and confirmed in Section 5, the $\mathbf{W}$ matrices learned on compressed feature vectors are competitive in terms of performance with their counterparts learned on uncompressed feature vectors. Furthermore, since at test time the feature vectors of each test example can be loaded sequentially, we use the exact feature vectors at test time.

Product quantization The product quantization (PQ) is a lossy encoding scheme. It splits a vector of size d into n_{sq} subvectors of size $d_{\text{sq}} = d/n_{\text{sq}}$. Each subvector is quantized with a nearest neighbor vector quantizer, producing an index between 1 and 256, that can be stored in a single byte. The whole descriptor matrix $\mathbf{X}$ is encoded as integer matrix $\mathcal{I} \in \{1 \cdots 256\}^{n_{\text{sq}} \times n}$. The n_{sq} quantizers are learned by k-means on the descriptor matrix, producing centroid matrices $\mathcal{C}_1 \cdots \mathcal{C}_{n_{\text{sq}}} \in \mathbb{R}^{d_{\text{sq}} \times 256}$. In our experiments we set $n_{\text{sq}} = 4096$ for $d = 65536$. Our largest $\mathcal{I}$, at below 1 GB, easily fits in RAM. In the following, we will call $\mathbf{X}$ the matrix of reconstructed descriptors. This means that we include the compression noise in the descriptor.

Algorithm 1 Matrix multiplication $A \times \mathbf{X}$. We use notations familiar to Matlab users: $M(i, j)$ represents element (i, j) of matrix M and : represents index slices.

Input: $A \in \mathbb{R}^{m \times d}$
Output: $C = A\mathbf{X}$
Algorithm:

1. precompute tables of dot products
   for $q = 1$ to n_{sq}
   $D_q = A(:, (q-1)d_{\text{sq}} + 1 : qd_{\text{sq}})\mathcal{C}_q$
2. accumulate result matrix
   for $i = 1$ to m
   for $j = 1$ to n
   for $q = 1$ to n_{sq}
   $C(i, j) = C(i, j) + D_q(i, \mathcal{I}(q, j))$

Algorithm 2 Matrix multiplication $\mathbf{X} \times B$

Input: $B \in \mathbb{R}^{n \times m}$
Output: $C = \mathbf{X}B \in \mathbb{R}^{d \times m}$
Algorithm:

1. accumulate intermediate matrix
   for $j = 1$ to m
   for $q = 1$ to n_{sq}
   for $j = 1$ to n
   $D_q(\mathcal{I}(q, i), j) = D_q(\mathcal{I}(q, i), j) + B(i, j)$
2. produce result
   for $q = 1$ to n_{sq}
   $C((q-1)d_{\text{sq}} + 1 : qd_{\text{sq}}, :) = \mathcal{C}_q D_q$

Matrix multiplications in the compressed domain. Our algorithm involves expensive matrix computations, such as matrix-vector or matrix-matrix multiplications between $\mathbf{X}$ and a dense floating point matrix. These operations can be implemented in the compressed domain. For both flavors of multiplication, the computation uses the same principle: we perform a matrix multiplication with the table of centroids instead of the full matrix $\mathbf{X}$. The results are combined by a pass over the $\mathcal{I}$ matrix, that simply does table lookups.

Implementation of $A\mathbf{X}$ and $\mathbf{X}B$. Our approach is inspired from [15]. Multiplying $\mathbf{X}$ with an arbitrary $A \in \mathbb{R}^{m \times d}$ consists in precomputing a table of dot products with all subvectors of columns of $\mathcal{C}_q$, see Algorithm 1. Multiply an arbitrary matrix $B \in \mathbb{R}^{n \times m}$ with $\mathbf{X}$, we gather the components of B that correspond to the same quantization index, see Algorithm 2).

Performance analysis The complexity of the exact full matrix-matrix multiplication is mnd multiply-adds:

operation	$A\mathbf{X}$	$\mathbf{X}B$
multiply-adds	$256dn_{\text{sq}}$	$256dm$
random accesses	mnn_{sq}	mnn_{sq}

The limiting factor of our algorithm are the random accesses, that tend to be slower than arithmetic operations. The speedup is therefore in the order of d/n_{sq}: reducing the number of quantizers speeds up the computation (but degrades the precision). By carefully organizing the loops in blocks, we optimize the cache accesses and obtain the following timings (for $m = 128$, on a 16-core Xeon):

operation	$A\mathbf{X}$	$\mathbf{X}B$
full matrix	52.4 s	43.8 s
compressed matrix	42.3 s	9.2 s

The 4-fold speed increase on $\mathbf{X}B$ is particularly useful in the gradient computations of Algorithm 4.1.

5. Experiments

We ran experiments on the Fungus, Ungulate and Vehicle datasets, as defined in [8]. It consists of three groups of leaf nodes that descend from particular parent nodes in the ImageNet hierarchy. They have 134, 182 and 262 classes respectively, each of which contains 193 to 2295 images. The three datasets together amount for a total of 512,282 images. Each dataset is randomly split 50%-50% into a set of training and test images, following the convention of the PASCAL VOC Challenge. To extract smaller subsets of images or classes, we adopt the alphabetical order given by the ImageNet names. For example, we write "V10" for the subset of Vehicle containing its 10 first classes.

Fisher Vectors We chose the Fisher image descriptors as feature vectors, as they have shown excellent results in the last ILSRVC Challenges. Similarly to [22], we use SIFT [19] and local color descriptors reduced to 128 D to train a Gaussian mixture model of 16 or 256 centroids. The Fisher vectors (derivatives w.r.t. μ and Σ) are 4096 and 65536 D, respectively.

Comparison We compared our approach to a classical one-vs-rest SVMs strategy. For high speed and memory efficiency, we use Liblinear [12], modified to use dense descriptors and parallelized without data duplication. In addition to dividing the required memory by four, we report a speed increase of approximately three times. To determine the C parameter of the SVM, and optimal rebalancing, we used three-fold cross-validation for five candidates in logarithmic range between 0.1 and 10^4. For our algorithm, we took $\varepsilon = 0.035$ and used 10 steps for λ_1 and λ_2 along the regularization path. We set the maximum number of iterations per (λ_1, λ_2) to 200 iterations.

Results We compare the two methods with 16-Gaussian Fisher vectors. As shown in Figure 2, the relative performance of our algorithm improves as the number of classes increases. This confirms that our approach scales up more gracefully to large number of classes compared to one-versus-rest classifiers. We observe the accuracy in top k, which is the number of correct labels that appear in the k first predicted scores. Our results outperform the one-versus-rest SVM. For comparison, [8] reports accuracies of around 12, 15 and 25 for Fungus, Ungulate and Vehicle respectively, far below our method.

Effect of compression To evaluate the effect of compression, we observe the classification results of V10. We take a compression factor of 64, which means codes use 256 (resp 4096) bytes for Fisher vectors of 4096 D (resp 65536 D). As shown on Figure 3, our proposed stategy gets almost half the compression error back. We conjecture that the quantization noise is almost *averaged out* in the empirical risk during training.

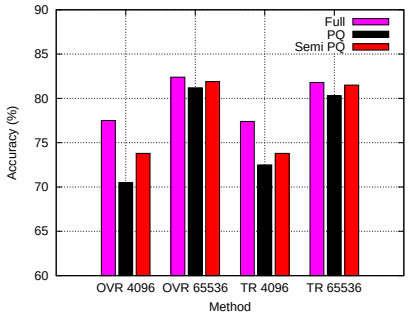


Figure 3. Accuracy performances on V10 (OVR = One vs Rest, TR = our trace regularisation, Full = No compression, PQ = fully compressed, Semi PQ = Compressed only for training).

Acknowledgements

We would like to warmly thank Florent Perronnin for kindly sharing his visual features with us, and Hervé Jégou for insightful discussions. This work was funded by a MathSTIC project from Grenoble University, the Quaero project (funded by OSEO, the French State agency for innovation), and the PASCAL 2 Network of Excellence.

References

[1] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *ICML*, 2007.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3), 2008.

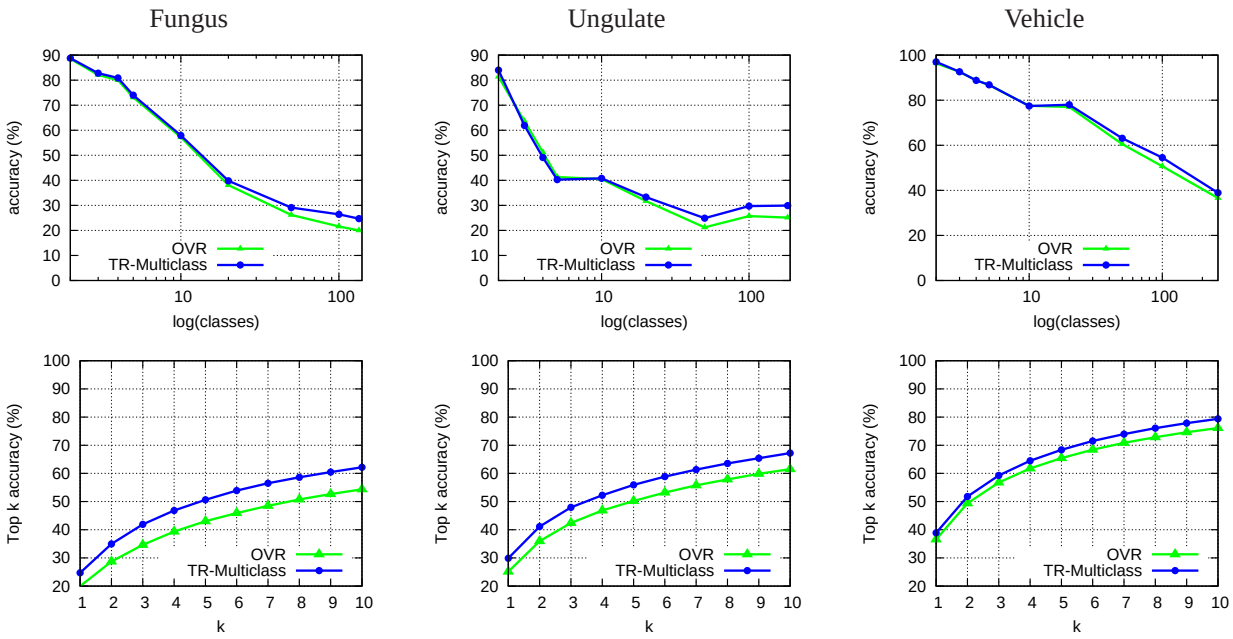


Figure 2. Top: Results of one-vs-rest and trace regularization, on the three groups of classes independently, for varying numbers of classes per group. Fisher vectors use 16 Gaussians. Bottom: Top-k against k for the three datasets in 4096-dim.

[3] P. L. Bartlett, M. . Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

[4] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

[5] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[6] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43.

[7] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Research*, 3, 2003.

[8] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10000 image categories tell us? In *ECCV*, 2010.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] J. Duchi and Y. Singer. Boosting with structural sparsity. ICML, 2009.

[11] M. Dudik, Z. Harchaoui, and J. Malick. Lifted coordinate descent for learning with trace-norm regularization. In *AISTATS*, 2012.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Research*, 9:1871–1874, June 2008.

[13] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford, 2002.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 2008.

[15] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1), 2011.

[16] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Research*, 10, 2009.

[17] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[18] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and SVM training. In *CVPR*, 2011.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.

[20] S. Mallat. *A Wavelet Tour of Signal Processing: the sparse way (3rd ed.)*. Academic Press, 2009.

[21] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[22] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2006.

[23] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, pages 143–156, 2010.

[24] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Research*, 5:101–141, 2004.

[25] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[26] S. Shalev-Shwartz, Y. Wexler, and A. Shashua. Shareboost: Efficient multiclass learning with feature sharing. 2011.

[27] A. Tewari and P. Bartlett. On the consistency of multiclass classification methods. *J. Mach. Learn. Research*, 8:1007–1025, 2007.

[28] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Math. Program.*, 117, 2009.

[29] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[30] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[31] Z. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.

[32] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc., Ser. B, Stat. Methodol.*, 67, 2005.