Int J Comput Vis (2016) 119:60–75
DOI 10.1007/s11263-015-0839-4

Sparse Output Coding for Scalable Visual Recognition

Bin Zhao[1] · **Eric P. Xing**[1]

Received: 15 May 2013 / Accepted: 16 June 2015 / Published online: 26 June 2015

Abstract Many vision tasks require a multi-class classifier to discriminate multiple categories, on the order of hundreds or thousands. In this paper, we propose sparse output coding, a principled way for large-scale multi-class classification, by turning high-cardinality multi-class categorization into a bit-by-bit decoding problem. Specifically, sparse output coding is composed of two steps: efficient coding matrix learning with scalability to thousands of classes, and probabilistic decoding. Empirical results on object recognition and scene classification demonstrate the effectiveness of our proposed approach.

Keywords Scalable classification · Output coding · Probabilistic decoding · Object recognition · Scene recognition

1 Introduction

Big data has recently attracted a great deal of interest in the vision community. However, previous research has been largely focused on situations involving only large number of data points and/or high-dimensional features, but relatively small task size. For example, many popular benchmark datasets (Fei-Fei et al. 2004; Griffin et al. 2007; Russell et al. 2008) involve only a limited number of class labels. In a modern era when prevalence of social media data and consumer-driven problems are inspiring attention on datasets and tasks mimicking human intelligence in real world, a new dimension of large scale machine learning and computer vision—large task space, merits serious attention due to a lack of scalable and robust new learning framework to meet the present and future challenges that are challenging the decade-old classical approaches still in service, such as kNN or one-vs-all style classification. Indeed, problems involving a large number of possible category labels (i.e., classes), in the order of tens or even hundreds of thousands, in addition to the large volume of data points and features, are easily within our reach. For example, ImageNet (Deng et al. 2009) for object recognition spans a total of 21,841 classes. Similarly, TinyImage (Torralba et al. 2008) contains 80 million 32×32 low resolution images, with each image loosely labeled with one of 75,062 English nouns. Clearly, these are no longer artificial visual categorization problems created for machine learning, but instead more like a human-level cognition problem for real world object recognition with a much bigger set of objects. A natural way to formulate this problem is a *multi-class* or *multi-task* classification, but the seemingly standard formulation on such gigantic dataset poses a completely new challenge both to computer vision and machine learning. Unfortunately, despite the well-known advantages and recent advancements of multi-class classification techniques (Bakker and Heskes 2003; Jacob et al. 2008; Binder et al. 2011) in machine learning, complexity concerns have driven most research on such large-scale datasets back to simple methods such as *nearest neighbor search* (Boiman et al. 2008), *least squares regression* (Fergus et al. 2010) or learning tens of thousands of binary classifiers (Lin et al. 2011).

With such large number of classes, it is no surprise that classical algorithms such as one-vs-rest, one-vs-one, or kNN,

Communicated by Antonio Torralba and Alexei Efros.

✉ Bin Zhao
binzhao@andrew.cmu.edu

Eric P. Xing
epxing@cs.cmu.edu

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, USA

often favored for their simplicity (Rifkin and Klautau 2004; Boiman et al. 2008), will be brought to their knees not only because of the training time and storage cost they incur (Kosmopoulos et al. 2010), but also because of the conceptual awkwardness of such algorithms in massive multi-class paradigms. For example, with 21,841 classes in the ImageNet problem, should we go ahead and build 21,841 classifiers each trained for 1-vs-21,840 classification? Just imagine the resultant data imbalance issue at its extreme, let alone the terrible irregularities of the decision boundaries of such classifiers. Worse still, the number of classes can even grow further in the future. One possible alternative that is attempted but still not popular is hierarchical classification (HC) (Koller and Sahami 1997; Dekel et al. 2004; Cai and Hofmann 2004; Bengio et al. 2010; Deng et al. 2011; Zhou et al. 2011; Beygelzimer et al. 2009; Beygelzimer et al. 2009), which in principle can reduce the number of classification decisions to $O(\log K)$, where K is the number of leaf classes. However, HC faces remarkable difficulty in practice for large scale problems because of a number of undesirable intrinsic properties, such as sensitivity to reliability of near-root classifiers, error propagation along the tree path, over-heterogeneity of training data for near-root super classes, etc. Clearly, massive multi-class classification with the number of classes approaching or even surpassing human cognitive capability is an important yet under-addressed research problem, and requires new, out-of-the-box rethinking of classical approaches and more effective yet simple alternatives (we emphasize simplicity as for massive multi-class problems, any computationally intense methods would immediately fall out of favor by practitioners).

Our goal in this work is to design a multi-class classification method that is both accurate and fast when facing a large number of categories. Specifically, we propose *sparse output coding (SpOC)*, which turns the original large-scale K-class classification into an L-bit code construction problem, where $L = \mathcal{O}(\log(K))$ and each bit can be constructed in parallel through a binary off-the-shelf classifier; followed by a probabilistic decoding scheme to extract the class label.

1.1 Previous Work

The following lines of research are related to our work.

1.1.1 Large-Scale Visual Recognition

Very recently, we have seen successful attempts in large-scale visual recognition (Lin et al. 2011; Sanchez et al. 2013; Le et al. 2012; Bengio et al. 2010; Deng et al. 2011; Gao and Koller 2011). Specifically, (Lin et al. 2011) employs sparse coding to represent each image as a high-dimensional coding vector. Similarly, (Sanchez et al. 2013) designs high-dimensional Fisher vector by describing image patches using their deviation from a "universal" Gaussian mixture model. Then (Le et al. 2012) utilizes deep neural networks to learn nonlinear feature representation for images via heavy parallel computation. However, all three works discussed above focus on designing high-dimensional feature representation for images, where classifier is trained using conventional one-vs-rest approach. *SpOC* serves as an important complement to this line of research, in the sense that we could very easily combine our classification method with feature representations learned in Lin et al. (2011), Le et al. (2012) to yield even better results. On the other hand, (Bengio et al. 2010; Deng et al. 2011) learn tree classifiers, where multiple classifiers are organized in a tree and a test image traverses the tree from root to leaf to obtain its class label. However, tree structured classifiers face the well-known error propagation problem, where errors made close to the root node are propagated through the tree and yield misclassification. On the other hand, *SpOC* is robust to errors in local classifiers, as a result of the error correcting property of output coding. Moreover, Gao and Koller 2011 introduced relaxed tree hierarchy, where a class can appear on both left child and right child of a node, with the ability to at least partially avoid error propagation. However, allowing a class to appear on both child nodes increases the computational complexity of the tree classifier. Moreover, Gao and Koller (2011) learns the relaxed tree structure and classifiers in a unified optimization framework, via alternating optimization. The fact that Gao and Koller (2011) needs to train classifiers multiple times in alternating optimization, renders it rather expensive computationally, especially for large-scale classification with big task space. We will show empirical results in Sect. 4 to demonstrate the efficiency of *SpOC* against relaxed tree classifier in Gao and Koller (2011).

1.1.2 Error Correcting Output Coding

For a K class problem, *error correcting output coding (ECOC)* Allwein et al. (2001) consists of two stages: coding and decoding. An output code $\mathbf{B}$ is a matrix of size $K \times L$ over $\{-1, 0, +1\}$ where each row of $\mathbf{B}$ corresponds to a class $y \in \mathcal{Y} = \{1, \ldots, K\}$. Each column $\boldsymbol{\beta}_l$ of $\mathbf{B}$ defines a partition of $\mathcal{Y}$ into three disjoint sets: positive partition ($+1$ in $\boldsymbol{\beta}_l$), negative partition (-1 in $\boldsymbol{\beta}_l$), and ignored classes (0 in $\boldsymbol{\beta}_l$). Binary learning algorithms are then used to construct bit predictor h_l using training data

$$\mathbf{Z}_l = \{(\mathbf{x}_1, B_{y_1,l}), \ldots, (\mathbf{x}_m, B_{y_m,l})\} \tag{1}$$

with $B_{y_i,l} \neq 0$, for $l = 1, \ldots, L$, where $\{\mathbf{x}_i\}_{i=1}^m$ are feature vectors for training examples, and $\{y_i\}_{i=1}^m$ are corresponding labels (throughout the rest of this paper, we use "bit predictor" to denote the binary classifier associated with a column of the coding matrix). Clearly, classical multi-class

categorization algorithms, such as *one-vs-one* and *one-vs-all* are special cases under the *ECOC* framework, with special choice of coding matrix (Allwein et al. 2001). Moreover, results in Allwein et al. (2001) suggest that learning a coding matrix in a problem-dependent way is better than using a predefined one. However, strong error-correcting ability alone does not guarantee good classification (Crammer and Singer 2002), since the performance of output coding is also highly dependent on the accuracy of the individual bit predictors. Consequently, several approaches (Schapire 1997; Crammer and Singer 2002; Gao and Koller 2011) optimizing coding matrix and bit predictors simultaneously have been proposed. However, the coupling of learning coding matrix and bit predictors in a unified optimization framework is both a blessing and a curse. On the one hand, it could directly assess the accuracy of each bit predictor and hence pick the coding matrix that avoids difficult bit prediction problems; on the other hand, simultaneous optimization often results in expensive computation, hindering these approaches from being applied to large-scale multi-class problems. Consequently, for the sake of scalability to massive number of classes, *SpOC* decouples the learning processes of code matrix and bit predictors. Therefore, the expensive procedure of learning bit predictors only needs to be carried out once, instead of multiple times in aforementioned approaches that learn code matrix and bit predictors simultaneously. However, our proposed approach still balances error-correcting ability of the code matrix and potential accuracy of associated bit predictors. Moreover, we also consider other properties that could affect classification accuracy of output coding based multi-class classifier, such as correlation among bit predictors, and complexity of each bit prediction problem. To the best of our knowledge, we provide the first attempt in learning optimal code matrix that explicitly considers multiple competing factors, and the fact that code matrix is learned without training associated binary classifiers multiple times enables our approach capable of handling massive multi-class classification problems.

Given a test instance $\mathbf{x}$, the decoding procedure finds the class y whose codeword in $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_L(\mathbf{x}))$. For binary output coding scenario, where $\mathbf{B} \in \{-1, +1\}^{K \times L}$, either Hamming distance or Euclidean distance could be adopted to measure distance between two codewords. However, in the ternary case, where $\mathbf{B} \in \{-1, 0, +1\}^{K \times L}$, the special 0 symbol indicating ignored classes could raise problems. Specifically, previous attempts in decoding ternary codes (Escalera et al. 2010) either (1) treat "0" bits the same way as non-zero bits, or (2) ignore those "0" bits entirely and only use non-zero bits for decoding. However, neither of the above approaches would prove sufficient. Specifically, treating "0" bits the same way as non-zero ones would introduce bias in decoding, since the distance increases with the number of positions that contain the zero symbol. On the other hand, ignoring "0" bits entirely would discard great amount of information. In our proposed framework, *probabilistic decoding* utilizes zero bits by propagating labels from non-zero bits to zero ones subject to smoothness constraints, and proves effective especially on large scale problems.

1.1.3 Attributes

This line of research (Farhadi et al. 2009; Kumar et al. 2009; Lampert et al. 2009; Wang et al. 2009; Torresani et al. 2010; Li et al. 2010; Deng et al. 2011; Patterson et al. 2012; Rastegari et al. 2012; Bergamo and Torresani 2012) employs attribute descriptors, mid-level semantic visual concepts such as "short", "furry", "leg", etc., which are shareable across categories, to encode categorical information as image features. Each attribute could be a response map of binary classifiers, and the object recognition task is carried out by utilizing multiple attributes as image features for training classifier. Specifically, the Meta-Class algorithm (Bergamo and Torresani 2012) employs label tree learning (Bengio et al. 2010) to learn meta-classes, which are set of classes that can be easily separated from others. Meta-Class algorithm could be seen as generalization of one-vs-rest, where instead of using only one class as positive data, it selects a set of classes called Meta-Class as positive data in learning binary classifiers.

1.1.4 Label Embedding

Another line of research related to this work is *label embedding* (Weinberger et al. 2008; Hsu et al. 2009; Weston et al. 2011; Zhang and Schneider 2012), where each class is represented by a prototype vector in some subspace, into which all training data points are also projected. The projection is optimized such that data points are mapped close to their corresponding class prototype. Classification is then carried out using nearest neighbor search in the subspace. Different from *label embedding*, our proposed method follows the idea of divide-and-conquer, which breaks a massive problem into a series of bit predictions, and combines all bit predictors for final classification through probabilistic decoding.

1.2 Summary of Contributions

To conclude the introduction, we summarize our main contributions as follows. (1) We propose an approach for large-scale visual recognition, with scalability to problems with tens of thousands of classes. *SpOC* is robust to errors in bit predictors, simple to parallelize, and its computational time scales sub-linearly with the number of classes. (2) We propose efficient optimization based on *alternating direction method of multipliers*, where each sub-problem is solved using gradient descent with Cayley transform to preserve

orthogonality constraint and curvilinear search for optimal step size. (3) We propose *probabilistic decoding* to effectively utilize semantic similarity between visual categories for accurate decoding. (4). We provide promising empirical results, tested on ImageNet with around 16,000 classes.

A shorter version of this work has appeared in Zhao and Xing (2013). The improvements in this work compared with Zhao and Xing (2013) are summarized as follows: (1) In Zhao and Xing (2013), we did not constrain the correlation among bit predictors, while in this work, we have added orthogonally constraints to ensure uncorrelated bit predictors. (2) Optimization is completely different. In Zhao and Xing (2013), we used constrained concave-convex procedure (CCCP) and dual proximal gradient method for optimization. However, the introduction of the orthogonally constraint greatly complicates the optimization, as in each step of gradient descent, we have to ensure the orthogonally constraint is satisfied. Moreover, we adopted a more efficient algorithm (*ADMM*) to handle the L1 norm in objective function. (3) We have added experimental results on the large ImageNet dataset.

2 Coding

In this section, we provide details of the formulation for learning optimal code matrix for large-scale multi-class classification, together with efficient optimization algorithm.

2.1 Formulation

Output coding employs a code matrix to break a potentially massive multi-class problem into a series of binary bit predictions. Clearly, code matrix is crucial for the success of output coding, and its suitability could be measured using several competing factors, such as error-correcting ability, learnability of each bit predictor, and correlation between bit predictors.

As its most attractive advantage, the code matrix in output coding is usually chosen for strong error-correcting ability. That is to say, the optimal code matrix should have maximal separation between codewords for different classes. Besides codeword separation, since output coding is essentially aggregating discriminative information residing in each bit, learning accurate bit predictors is also crucial for its success. However, we usually do not know whether a binary partition can be well handled by the base bit predictor, unless a bit predictor has been learned on the partition. Unfortunately, the high computational cost associated with methods optimizing coding matrix and bit predictors simultaneously (Gao and Koller 2011) renders them unfavorable in large-scale problems. To overcome this difficulty, we propose to use the training data and structure information among classes, to provide a measure of separability for each binary partition problem. Specifically, if some classes are closely related but are given different codes in the lth bit, the bit predictor h_l may not be easily learnable. However, a binary partition is more likely to be well solved if the intra-partition similarity is large while the inter-partition similarity is small. Moreover, as output coding predicts class label by combining information from all bits, an ideal code matrix should have uncorrelated columns. Specifically, uncorrelated columns mean each bit predictor is focusing on a unique sub-problem of the original multi-class classification, while highly correlated columns severely limit the amount of information available at decoding. Finally, enforcing sparsity of the code matrix, i.e., introducing ignored classes in bit predictions, is crucial for massive multi-class classification. In this section, we will provide details for each of the aforementioned pieces, and formulate learning optimal code matrix as an orthogonality constrained optimization problem.

Before presenting the detailed formulation for learning the code matrix, we would like to re-emphasize that the goal and contribution of this work is effective multi-class classification with massive number of classes, where any complex method would fail, and simplicity prevails. As a result, motivation for design of each piece in the optimization problem is balance between effectiveness and efficiency. Although there might be more sophisticated formulations of the optimization problem, they will very likely increase computational cost, ultimately rendering the method incapable of handling massive multi-class classification.

2.1.1 Codeword Separation

Given an example $\mathbf{x}$, an L-dimensional bit predictor $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is computed. We then predict its label y based on which row in $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x})$. To increase tolerance of errors occurred in bit predictions, a crucial design objective of the code matrix is to ensure that the rows in $\mathbf{B}$ are separated as far from each other as possible. Hence, we propose to maximize the distance between rows in $\mathbf{B}$. Equivalently, we could minimize their inner products. Thus, codeword correlation of $\mathbf{B}$ could be computed as following:

$$\sum_{k=1}^{K}\sum_{k'=1}^{K}\mathbf{r}_k^\top\mathbf{r}_{k'} = \mathbf{e}_K^\top(\mathbf{B}\mathbf{B}^\top)\mathbf{e}_K = tr(\mathbf{B}^\top\mathbf{E}\mathbf{B}) \tag{2}$$

where $tr(\cdot)$ is matrix trace, $\mathbf{r}_1^\top, \ldots, \mathbf{r}_K^\top$ are row vectors of code matrix $\mathbf{B}$, $\mathbf{e}_K \in \mathbb{R}^K$ is the all-one vector and $\mathbf{E} = \mathbf{e}_K\mathbf{e}_K^\top$ is the $K \times K$ all-one matrix.

2.1.2 Learnability of Bit Predictors

One key property of optimal code matrix is to ensure that the resulting bit prediction problems could be accurately solved. The key motivation of our mathematical formulation is to compute the following two measures using semantic relatedness matrix $\mathbf{S}$ (defined later in this section) for each binary partition problem: intra-partition similarity, and inter-partition similarity. Specifically, in each binary partition problem, both positive partition and negative partition are composed of data points from multiple classes in the original problem. To encourage better separation, those classes composing the positive partition should be similar to each other. The similar argument goes for those classes composing the negative partition, but they should be dissimilar from the former set which composes the positive partition. Specifically, separability of the lth binary partition problem could be measured as follows:

$$\sum_{B_{kl}B_{k'l}>0} S_{kk'} - \sum_{B_{kl}B_{k'l}<0} S_{kk'} = \sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kl}B_{k'l}S_{kk'} \tag{3}$$

It should be noted that the above defined measure should subtract $\sum_k S_{kk}$. However, as $\sum_k S_{kk}$ is constant and will not affect optimization of $\mathbf{B}$, we omit this step. Finally, learnability of all bit predictions could be measured as

$$\begin{aligned}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kl}B_{k'l}S_{kk'} &= \sum_{l=1}^{L}\mathbf{e}_K^\top\left[\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top\odot\mathbf{S}\right]\mathbf{e}_K \\ &= \mathbf{e}_K^\top\left[\mathbf{B}\mathbf{B}^\top\odot\mathbf{S}\right]\mathbf{e}_K \\ &= tr(\mathbf{B}\mathbf{B}^\top\mathbf{S}) = tr(\mathbf{B}^\top\mathbf{S}\mathbf{B})\end{aligned} \tag{4}$$

where $\odot$ is Hadamard (a.k.a., element-wise) product of two matrices, $\boldsymbol{\beta}_l$ is the lth column of $\mathbf{B}$, and we have used the fact that $\mathbf{B}\mathbf{B}^\top = \sum_l \boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top$ and $\mathbf{e}^\top(\mathbf{A}\odot\mathbf{B})\mathbf{e} = tr(\mathbf{A}\mathbf{B})$.

Semantic Relatedness Matrix S measures similarity between classes, using training data and structure information among classes. Let $\mathcal{X}_i = \{X_1^{(i)}, \ldots, X_{|\mathcal{X}_i|}^{(i)}\}$ and $\mathcal{X}_j = \{X_1^{(j)}, \ldots, X_{|\mathcal{X}_j|}^{(j)}\}$ be two classes from the multi-class problem. Several approaches have been proposed to measure similarity/distance between them, such as *Hausdorff distance*, *match kernel* (Haussler 1999; Parsana et al. 2007), divergence between probability distributions estimated from $\mathcal{X}_i$ and $\mathcal{X}_j$ (Póczos et al. 2011), or even classification accuracy of binary classifiers trained to separate the classes (Bengio et al. 2010). In this work, we use sum match kernel (Haussler 1999), and define data similarity between $\mathcal{X}_i$ and $\mathcal{X}_j$ as

$$S_{ij}^D = \frac{1}{|\mathcal{X}_i|}\frac{1}{|\mathcal{X}_j|}\sum_{p=1}^{|\mathcal{X}_i|}\sum_{q=1}^{|\mathcal{X}_j|} K_D(X_p^{(i)}, X_q^{(j)}) \tag{5}$$

where superscript D indicates that the similarity is estimated from data (in comparison to the similarity estimated from class structure discussed later), K_D is a Mercer kernel and in this work we use linear kernel.

Moreover, classes in massive multi-class problems are rarely organized in a flat fashion, but instead with a taxonomical structure (Deng et al. 2009; Cai and Hofmann 2004; Deng et al. 2011), such as a tree. Besides, algorithms for learning class structure have also been proposed (Bengio et al. 2010; Zhou et al. 2011), although this is beyond the scope of this work. Following Budanitsky and Hirst (2006) we define structural affinity A_{ij} between class i and class j as the number of nodes shared by their two parent branches, divided by the length of the longest of the two branches

$$A_{ij} = \text{intersect}(P_i, P_j)/\max(\text{length}(P_i), \text{length}(P_j)) \tag{6}$$

where P_i is the path from root node to node i and intersect(P_i, P_j) counts nodes shared by two paths P_i and P_j. We then construct structural similarity matrix

$$\mathbf{S}^S = \exp(-\kappa(\mathbf{E}-\mathbf{A})) \tag{7}$$

where κ is a constant controlling the decay factor. It should be noted that although we use class structure to define similarity, the goal of this work is not to propose yet another hierarchical classifier. In cases without such hierarchy, other ways of defining similarity between classes suffice as well (for example, we could simply use $\mathbf{S}^D$ only). Finally, semantic relatedness matrix $\mathbf{S}$ is the weighted sum

$$\mathbf{S} = \alpha\mathbf{S}^D + (1-\alpha)\mathbf{S}^S \tag{8}$$

with $\alpha \in [0, 1]$ being the weight.

2.1.3 Relaxation and Bit Correlation

Theoretical work Crammer and Singer (2002) shows that learning discrete code matrix directly is NP-complete. Thus, we follow Crammer and Singer (2002) and allow code matrix to take real values, followed by post-processing (taking the sign) to get the discrete code matrix.

Moreover, the power of output coding for multi-class classification stems from the fact that the final prediction is obtained by combining information from multiple bit predictors. Consequently, the more uncorrelated the bit predictors are, the more information we have at decoding time, and hence the better classification accuracy can be expected. As an extreme example, if all columns of the code matrix are

solving the same binary separation problem, the amount of information available at decoding time is one single bit, and it is obviously not sufficient for accurate multi-class classification. Therefore, to ensure maximal amount of information at decoding, an ideal code matrix should have uncorrelated columns, such that each bit predictor is tackling a unique sub-problem. To minimize bit correlation, we constrain the columns in code matrix to be orthogonal to each other, i.e.,

$$\mathbf{B}^\top \mathbf{B} = \mathbf{I} \tag{9}$$

where $\mathbf{I}$ is the identity matrix.

2.1.4 Sparse Code Matrix

For massive multi-class problems, it is crucial to introduce ignored classes, i.e., 0 in the code matrix (Allwein et al. 2001; Schapire and Freund 2012). Otherwise, every bit predictor needs to consider the entire data. As an illustrating example, consider the ImageNet problem. With each of the 21,841 classes participating in training a bit predictor, we will likely be facing a binary partition problem where both the positive and negative partitions are populated with data points coming from thousands of different classes. Clearly, learning bit predictor for such binary partition will be extremely difficult, due to the huge intra-partition dissimilarity. Therefore, to introduce ignored classes in bit predictors, we further regularize the l_1 norm of $\mathbf{B}$.

2.1.5 Final Formulation

Combining all pieces together, optimal code matrix should have minimal codeword correlation, maximal learnability of bit predictors, sufficient sparsity to reduce complexity of learning bit predictors, and orthogonal columns for uncorrelated bits. Weighing the above objectives, we propose the following formulation for learning optimal code matrix $\mathbf{B} \in \mathbb{R}^{K\times L}$

$$\min_{\mathbf{B}} \ \frac{1}{2} tr[\mathbf{B}^\top(\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \lambda_1 \|\mathbf{B}\|_1 \tag{10}$$

$$s.t. \ \ \mathbf{B}^\top \mathbf{B} = \mathbf{I} \tag{11}$$

where $\|\mathbf{B}\|_1 = \sum_{i,j} |B_{ij}|$ is its l_1 vector norm, λ_r and λ_1 are regularization parameters.

Selecting optimal parameters It should be noted that unlike multi-class classification problems with small task space, the sheer size of the problem *SpOC* is designed to handle, makes it impossible to perform cross-validation or leave-one-out procedures to select optimal values for the parameters, such as α in (8), λ_r and λ_1 in (10). Thus, our approach for parameter selection is based on grid search, where we try several different values for each of $\{\alpha, \lambda_r, \lambda_1\}$ and solve problem (10) for optimal coding matrix. Then we compute the relative ratio among the three components in objective function of (10). Finally, the optimal parameters are selected as the group resulting in the relative ratio closest to 1. The motivation for such strategy is to ensure that each piece in the objective function has comparable value, such that all pieces could contribute and compete for optimal code matrix.

2.2 Optimization

The difficulty of solving problem (10) lies in the non-smoothness of the l_1 regularization on $\mathbf{B}$, and the orthogonality constraint (11). In this work, we employ *alternating direction method of multipliers (ADMM)* (Boyd et al. 2011) to effectively reduce the l_1 regularized problem into a series of l_2 regularized problems, where each problem is solved using gradient descent with Cayley transform to preserve orthogonality constraint on $\mathbf{B}$ and curvilinear search for optimal step size (Wen and Yin 2012).

2.2.1 Alternating Direction Method of Multipliers

ADMM is a simple yet powerful algorithm, which takes the form of a decomposition-coordination procedure (Boyd et al. 2011), where the solutions to small local subproblems are coordinated to find a solution to a large global problem. *ADMM* was first introduced in the 1970s (Gabay and Mercier 1976), with most of the theoretical results established in the 1990s (Eckstein and Bertsekas 1992). Moreover, it is shown in Boyd et al. (2011) that *ADMM* converges to local optimal point for non-convex optimization problems. However, until very recently, *ADMM* was not widely known in the computer vision or machine learning community. For completeness, we provide a brief review of the algorithm [for more details, see Boyd et al. (2011)]. *ADMM* solves problems in the form

$$\min_{\mathbf{x},\mathbf{z}} \ f(\mathbf{x}) + g(\mathbf{z}), \ \ s.t. \ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \tag{12}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times m}$, and $\mathbf{c} \in \mathbb{R}^p$. For problem (12), the augmented Lagrangian is formed as follows:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \tag{13}$$

where $\rho > 0$ is called the penalty parameter. *ADMM* consists of the following iterations (Boyd et al. 2011)

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \tag{14}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \tag{15}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}) \tag{16}$$

To solve problem (10), we first reformulate it as follows

$$\min_{\mathbf{B},\mathbf{Z}} \ \frac{1}{2} tr[\mathbf{B}^\top(\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) + \lambda_1 \|\mathbf{Z}\|_1 \tag{17}$$

$$s.t. \ \ \mathbf{B} - \mathbf{Z} = \mathbf{0} \tag{18}$$

where $\mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) = 0$ if constraint $\mathbf{B}^\top\mathbf{B} = \mathbf{I}$ is satisfied, and $\mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) = +\infty$ otherwise. Then *ADMM* solves problem (17) using the following iterations:

$$\mathbf{B}^{k+1} = \arg\min_{\mathbf{B}} \left(f(\mathbf{B}) + \frac{\rho}{2} \|\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k\|_2^2 \right) \tag{19}$$

$$\mathbf{Z}^{k+1} = \mathcal{S}_{\lambda_1/\rho}(\mathbf{B}^{k+1} + \mathbf{U}^k) \tag{20}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{B}^{k+1} - \mathbf{Z}^{k+1} \tag{21}$$

where $f(\mathbf{B})$ is defined as

$$f(\mathbf{B}) = tr[\mathbf{B}^\top(\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \mathcal{I}(\mathbf{B}^\top\mathbf{B} = \mathbf{I}) \tag{22}$$

and $\mathcal{S}$ is the soft-thresholding operator defined as

$$\mathcal{S}_\kappa(a) = \max\{(1 - \kappa/|a|)a, 0\} \tag{23}$$

In the above *ADMM* iterations, both $\mathbf{Z}$ update (20) and $\mathbf{U}$ update (21) are trivial to compute. The $\mathbf{B}$ update in (19) is equivalent to the following constrained optimization

$$\min_{\mathbf{B}} \ \frac{1}{2} tr[\mathbf{B}^\top(\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B}] + \frac{\rho}{2} \|\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k\|_2^2 \tag{24}$$

$$s.t. \ \ \mathbf{B}^\top\mathbf{B} = \mathbf{I} \tag{25}$$

Comparing the above problem with (10), we can see that *ADMM* effectively reduces an l_1 regularized problem into a series of l_2 regularized problems.

2.2.2 Solving Problem (24) Using Cayley Transform and Curvilinear Search

Problem (24) is difficult to optimize due to the orthogonality constraint (25) on $\mathbf{B}$. In this work, we follow state-of-the-art technique (Wen and Yin 2012), and solve problem (24) using gradient descent, with Cayley transform to preserve the orthogonality constraint and curvilinear search for optimal step size. In each iteration of the algorithm, given current feasible solution $\mathbf{B}$, the gradient of the objective function w.r.t. $\mathbf{B}$ could be computed as

$$\mathbf{G} = (\lambda_r \mathbf{E} - \mathbf{S})\mathbf{B} + \rho(\mathbf{B} - \mathbf{Z}^k + \mathbf{U}^k) \tag{26}$$

Then a skew-symmetric matrix $\mathbf{A}$ is computed as

$$\mathbf{A} = \mathbf{G}\mathbf{B}^\top - \mathbf{B}\mathbf{G}^\top \tag{27}$$

The next new trial point $\mathbf{B}(\tau)$ is determined by the Crank–Nicolson like scheme Wen and Yin (2012)

$$\mathbf{B}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1} \left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right) \mathbf{B} \tag{28}$$

It is easy to verify that

$$\mathbf{B}(\tau)^\top\mathbf{B}(\tau) = \mathbf{B}^\top\mathbf{B} \tag{29}$$

i.e., every intermediate result is feasible, as long as initial point $\mathbf{B}$ satisfies the orthogonality constraint. For fast convergence, we adopt the Barzilai–Borwein step size in curvilinear search (Wen and Yin 2012) to find optimal τ. Moreover, since $(\mathbf{I} + \frac{\tau}{2}\mathbf{A})^{-1}$ dominates the computation in (28), we apply the Sherman–Morrison–Woodbury theorem for efficient computation of matrix inverse. Theoretical results in Wen and Yin (2012) show the above algorithm converges to local optimal point.

Finally, we present in Algorithm 1 the method for learning optimal code matrix.

Algorithm 1 Sparse Output Coding: Optimal Code Matrix Learning

```
Initialize B with randomly generated orthogonal matrix, Z = U = 0
repeat
  repeat
    Compute skew-symmetric matrix A
    Curvilinear search for optimal step size τ
    Update new trial point B(τ) as in Eq. (28)
  until stopping criterion satisfied Wen and Yin (2012)
  Z update using Eq. (20)
  U update using Eq. (21)
until stopping criterion satisfied Boyd et al. (2011)
```

3 Probabilistic Decoding

For large-scale multi-class categorization, a sparse output coding matrix is necessary to ensure the learnability of each bit predictor. However, the zero bits in coding matrix also bring difficulty in decoding. Specifically, given an instance $\mathbf{x}$, we denote the vector of predictions generated by learned bit predictors as $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_L(\mathbf{x}))$. The decoding procedure in *output coding* is to find the class y for which codeword of $\mathbf{B}$ is "closest" to $\mathbf{h}(\mathbf{x})$. In the simple case where a binary coding matrix $\mathbf{B} \in \{-1, +1\}^{K \times L}$ is adopted, the most frequently applied decoding approaches include Hamming decoding (Nilsson 1965; Dietterich and Bakiri 1995) and

Table 1 Decoding strategies for *error correcting output coding*

Decoding algorithm	Optimal label
Hamming decoding (Nilsson 1965; Dietterich and Bakiri 1995)	$y^* = \arg\min_{y\in\mathcal{Y}} \sum_{l=1}^{L} \frac{1}{2}(1 - \text{sgn}(h_l(\mathbf{x}) \cdot B_{y,l}))$
Euclidean decoding (Pujol et al. 2006)	$y^* = \arg\min_{y\in\mathcal{Y}} \sqrt{\sum_{l=1}^{L}(h_l(\mathbf{x}) - B_{y,l})^2}$
Attenuated Euclidean decoding Escalera et al. (2010)	$y^* = \arg\min_{y\in\mathcal{Y}} \sqrt{\sum_{l=1}^{L} \lvert B_{y,l}\rvert (h_l(\mathbf{x}) - B_{y,l})^2}$
Loss-based decoding (Allwein et al. 2001)	$y^* = \arg\min_{y\in\mathcal{Y}} \sum_{l=1}^{L} loss(h_l(\mathbf{x}) B_{y,l})$
Probability-based decoding (Passerini et al. 2004)	$y^* = \arg\min_{y\in\mathcal{Y}} -\log\left(\prod_{l:B_{y,l}\neq 0} \mathbb{P}(h_l(\mathbf{x}) = B_{y,l}) + K\right)$

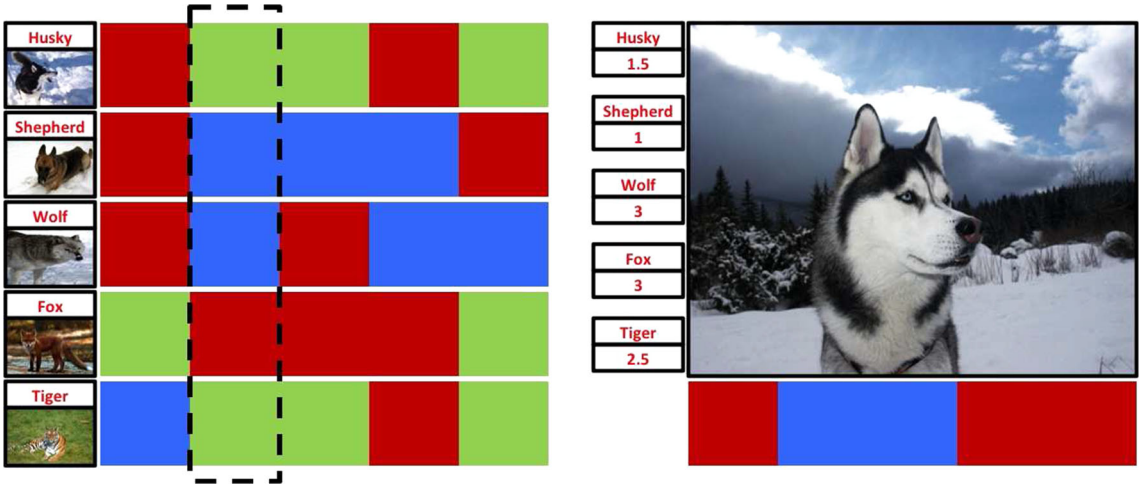

Fig. 1 (Best viewed in color) Motivation for probabilistic decoding: (*Left*). one possible coding matrix for 5-class categorization, with $red = +1$, $blue = -1$, and $green = 0$; (*Right*). one test image from class *Husky*, with its codeword shown in the bottom and Hamming distance with codewords for the five classes shown to the left. For the second bit (highlighted in *dash box*), although the first node (class *Husky*) is ignored during learning the bit predictor, it has a preference of being colored blue, rather than red than red (Color figure online)

Euclidean decoding Pujol et al. (2006), defined in Table 1. For ternary decoding with $\mathbf{B} \in \{-1, 0, +1\}^{K\times L}$, we could still apply those binary decoding strategies, treating 0 bits equally as non-zero ones, although we will encounter decoding bias as illustrated later in this section. One alternative strategy proposed in the literature ignores all zero bits in the coding matrix during decoding, and only counts the matching with non-zero bits. One example is attenuated Euclidean decoding (Escalera et al. 2010), an adaptation of the Euclidean decoding strategy, which makes the measure unaffected by the zero bits of the codeword. Moreover, Allwein et al. (2001) further improves the ternary decoding strategy by replacing the Euclidean distance with loss function, as defined in Table 1, where $h_l(\mathbf{x})B_{y,l}$ corresponds to the margin and $loss(\cdot)$ is a loss function that depends on the nature of the binary bit predictor. Finally, the authors of Passerini et al. (2004) propose a probability-based decoding strategy based on the continuous output of binary classifiers to deal with the ternary decoding. However, the probabilistic formulation in Passerini et al. (2004) only uses non-zero bits in the code matrix, and is thus equivalent to the loss-based decoding strategy in Allwein et al. (2001) with a logistic loss function. To sum up, although this latter group of approaches would avoid the problem of decoding bias on 0 bits of the coding matrix, it also discards significant amount of information, as only those non-zero bits are used for decoding.

3.1 Motivating Example

As a motivating example, consider a five-class problem in Fig. 1. Given a test image from class *Husky*, if we treat zero bits the same way as non-zero ones, both Hamming decoding and Euclidean decoding would prefer *Shepherd* over *Husky*. However, *Husky* is only worse than *Shepherd* as its codeword has more zero. This effect occurs because the decoding value increases with the number of positions that contain the zero symbol and hence introduces bias. This problem might not seem severe in the example shown in Fig. 1, however, for massive multi-class problems with large number of class labels, the bias introduced through zero bits would significantly affect classification accuracy. On the other hand, ignoring zero bits entirely would discard great amount of

information that could potentially help in decoding the correct class label. This is especially true when K is large, where we expect a very sparse coding matrix to maximize learnability of each binary classifier. For example, in Fig. 1, both classes *Husky* and *Tiger* have only two non-zero bits in their codewords. Since we cannot always have perfect bit predictors, classification errors on bit 1 and bit 4 would severely impair the overall accuracy. Therefore, it is our goal in this work to effectively utilize information residing in zero bits to effectively decode ternary output codes.

Fortunately, the semantic class similarity $\mathbf{S}$ computed using training data and class taxonomy, provides venue for effectively propagating information from non-zero bits to zero ones. For the example in Fig. 1, class *Husky* is more similar to *(Shepherd, Wolf)* than *(Fox, Tiger)*. The second bit predictor in Fig. 1 solves a binary partition of *(Shepherd, Wolf)* against *Fox*. Even though class *Husky* is ignored in training for this bit, the binary partition on images from this class will have a higher probability of being $+1$, due to the fact that the two positive classes in this binary problem are closely related to class *Husky*. Therefore, those classes with non-zero bits in the coding matrix, should effectively propagate their label to those initially ignored classes. In this section, we propose probabilistic decoding, to effectively utilize semantic class similarity for better decoding. Specifically, we treat each bit prediction (without loss of generality, say, the lth bit) as a label propagation (Zhu et al. 2003) problem, where the labeled data corresponds to those classes whose codeword's lth bit is non-zero, and unlabeled data corresponds to those whose lth bit is zero. The goal of label propagation is to define a prior distribution indicating the probability of one class being classified as positive in the lth binary partition. Combining this prior with the available training data, we formulate the decoding problem in *sparse output coding* as *maximum a posteriori* estimation.

3.2 Formulation

Given code matrix $\mathbf{B} \in \{-1, 0, +1\}^{K\times L}$, our decoding method estimates conditional probability of each class k given input $\mathbf{x}$ and L bit predictors $\{h_1, \ldots, h_L\}$. Without loss of generality, we assume the bit predictors constructed in the coding stage are linear classifiers, each parameterized by a vector $\mathbf{w}$ as $h_l(\mathbf{x}) = \text{sign}(\mathbf{w}_l^\top \mathbf{x})$. Define $(c_1, \ldots, c_L) \in \{-1, +1\}^L$ as a random vector of binary values, representing one possible codeword for instance $\mathbf{x}$. The decoding problem is then to find the class k, which maximizes the following conditional probability:

$$
\begin{aligned}
&\mathbb{P}(y = k|\mathbf{w}_1, \ldots, \mathbf{w}_L, \mathbf{x}, \boldsymbol{\mu}) \\
&= \sum_{\{c_l\}} \mathbb{P}(y = k|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}, \{c_l\}) \cdot \mathbb{P}(\{c_l\}|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}) \\
&= \sum_{\{c_l\}} \mathbb{P}(y = k|\boldsymbol{\mu}, \{c_l\}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&\propto \sum_{\{c_l\}} \prod_l \mathbb{P}(c_l|y = k, \mu_{kl}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \sum_{\{c_l\}} \prod_l \mu_{kl}^{c_l}(1 - \mu_{kl})^{1-c_l} \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \prod_l \{\mu_{kl}\mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}) + (1 - \mu_{kl})(1 - \mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}))\}
\end{aligned}
\tag{30}
$$

where $\{c_l\} = \{c_1, \ldots, c_L\}$, $\{\mathbf{w}_l\} = \{\mathbf{w}_1, \ldots, \mathbf{w}_L\}$, and $\mu_{kl} \in [0, 1]$ is the parameter in Bernoulli distribution $\mathbb{P}(c_l = 1|y = k) = \mu_{kl}$. Moreover, given the learned bit predictors, $\mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x})$ could be computed using a logistic link function as follows

$$
\mathbb{P}(c_l = 1|\mathbf{w}_l, \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}_l^\top \mathbf{x})} \tag{31}
$$

Therefore, in order to employ conditional probability in decoding for ternary output codes, we need to learn the values of Bernoulli parameters $\{\mu_{kl}\}_{k=1,\ldots,K}^{l=1,\ldots,L}$, which measures the probability of the lth bit being $+1$ given the true class as $y = k$. Specifically, for the lth column of the coding matrix, those classes corresponding to $+1$ in the lth bit, i.e., $B_{kl} = 1$, will have $\mu_{kl} = 1$, and similarly those classes corresponding to -1, i.e., $B_{kl} = -1$, will have $\mu_{kl} = 0$. However, originally ignored classes (those corresponding to 0 in the coding matrix) will also be likely to have a preference on the value of the lth bit. For the example in Fig. 1, the second bit predictor separates *(Shepherd, Wolf)* from *Fox*. Clearly, $\mathbb{P}(c_l = 1|Shepherd) = \mathbb{P}(c_l = 1|Wolf) = 0$ and $\mathbb{P}(c_l = 1|Fox) = 1$. Since class *Husky* is not directly involved in this binary classification problem, a non-informative prior would put $\mathbb{P}(c_l = 1|Husky) = 0.5$. However, if the true class for an instance $\mathbf{x}$ is *Husky*, this bit clearly has a much higher probability of being -1 than $+1$, due to the fact that *Husky* is much closer to *Shepherd* and *Wolf* semantically, than *Fox*. Therefore, we should have $\mathbb{P}(c_l = 1|Husky) < 0.5$, and in such way those classes with non-zero values in the lth bit effectively propagate their label through the semantic class similarity $\mathbf{S}$ to those initially ignored classes.

3.2.1 Prior Distribution via Label Propagation

Following the motivating example in Fig. 1, we define a prior distribution over Bernoulli parameters $\boldsymbol{\mu}$ such that labeled nodes effectively propagate their labels to those unlabeled nodes following the class hierarchy. Since each column $\boldsymbol{\beta}_l$ in the coding matrix will have its own labeling (different composition of positive classes, negative classes, and ignored classes) and label propagation for each column is indepen-

dent of others, without loss of generality, we will focus on the lth column. Suppose we have $\hat{l}$ classes participating in learning the lth binary classifier, corresponding to the $\hat{l}$ non-zero terms in the lth column $\boldsymbol{\beta}_l$ of coding matrix $\mathbf{B}$. Moreover, we also have $\hat{u} = K - \hat{l}$ ignored classes, corresponding to zeros in $\boldsymbol{\beta}_l$. Without loss of generality, we assume the first $\hat{l}$ classes are labeled as $(y_1, \ldots, y_{\hat{l}}) \in \{-1, +1\}^{\hat{l}}$. Consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with K nodes $\mathcal{V}$ corresponding to the K classes, where nodes $\hat{L} = \{1, \ldots, \hat{l}\}$ correspond to the labeled classes, and nodes $\hat{U} = \{\hat{l}+1, \ldots, K\}$ correspond to ignored classes. Our task is to assign probabilities to nodes $\hat{U}$ being labeled as positive, using label information on nodes $\hat{L}$ and the graph structure of $\mathcal{G}$. Define $\boldsymbol{\mu}_l = (\mu_{1l}, \ldots, \mu_{Kl})$ as labels on nodes $\mathcal{V}$, where $\mu_{kl} = 1$ for those classes labeled as $+1$ and $\mu_{kl} = 0$ for those classes labeled as -1 in the lth column of coding matrix $\mathbf{B}$. Consequently, the value of $\boldsymbol{\mu}_l$ on those unlabeled nodes represents our belief of it being labeled as $+1$. Equivalently, the distribution of $\boldsymbol{\mu}_l$ defines a prior on our Bernoulli parameters, in the sense that $\mu_{kl} = \mathbb{P}(c_l = 1 | y = k)$. Intuitively, we want unlabeled nodes that are nearby in the graph to have similar labels, and this motivates the choice of the following quadratic energy function (Zhu et al. 2003):

$$E(\boldsymbol{\mu}_l) = \frac{1}{2} \sum_{i,j} S_{ij} (\mu_{il} - \mu_{jl})^2 \tag{32}$$

where $\mathbf{S}$ is the semantic similarity matrix. To assign probability distribution on $\boldsymbol{\mu}_l$, we form Gaussian field (Zhu et al. 2003)

$$p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C} \exp(-C E(\boldsymbol{\mu}_l)) \tag{33}$$

where C is inverse temperature parameter, and

$$Z_C = \int_{\boldsymbol{\mu}_l | \forall k \in \hat{L}: \mu_{kl} = \frac{1}{2}(B_{kl}+1)} \exp(-C E(\boldsymbol{\mu}_l)) d\boldsymbol{\mu}_l \tag{34}$$

is a normalizing constant over all possible $\boldsymbol{\mu}_l$ constrained to $\boldsymbol{\beta}_l$ on non-zero terms, i.e., the labeled nodes in the graph corresponding to $\boldsymbol{\beta}_l$. Define diagonal degree matrix $\mathbf{D}$ with $D_{ii} = \sum_j S_{ij}$ and graph Laplacian $\Delta = \mathbf{D} - \mathbf{S}$, the Gaussian field defined on $\boldsymbol{\mu}$ could be equivalently formulated as follows:

$$p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C} \exp(-C \boldsymbol{\mu}_l^\top \Delta \boldsymbol{\mu}_l) \tag{35}$$

with $\mu_{kl} = 1$ on classes labeled as $+1$ in $\boldsymbol{\beta}_l$ and $\mu_{kl} = 0$ on classes labeled as -1 in $\boldsymbol{\beta}_l$. Consequently, $p_C(\boldsymbol{\mu}_l)$ defines a prior distribution on $\boldsymbol{\mu}_l$, following the semantic class similarity, by clamping the labels on non-zero terms, and forcing smoothness of labels for zero terms, i.e., closely related classes should receive similar labels.

3.2.2 Parameter Learning

Given the L bit predictors learned in the coding stage of *sparse output coding*, and m training data points $\mathbf{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, we could calculate the conditional log-likelihood as follows:

$$\begin{aligned} &\log \mathbb{P}(Y | \{\mathbf{w}_l\}, \mathbf{X}, \boldsymbol{\mu}) \\ &= \sum_{i=1}^{m} \sum_{l=1}^{L} \log \left\{ \mu_{y_i l} \mathbb{P}_{li} + (1 - \mu_{y_i l})(1 - \mathbb{P}_{li}) \right\} \end{aligned} \tag{36}$$

where $\mathbb{P}_{li} = \mathbb{P}(c_l = 1 | \mathbf{w}_l, \mathbf{x}_i)$. Combining the above defined data likelihood with prior distribution over $\boldsymbol{\mu}$, we get the following optimization problem for learning parameters $\boldsymbol{\mu}$ using MAP estimation

$$\begin{aligned} \min_{\boldsymbol{\mu}} & - \sum_{i=1}^{m} \sum_{l=1}^{L} \log \left\{ \mu_{y_i l} \mathbb{P}_{li} + (1 - \mu_{y_i l})(1 - \mathbb{P}_{li}) \right\} \\ & + C \sum_{l=1}^{L} \boldsymbol{\mu}_l^\top \Delta \boldsymbol{\mu}_l \end{aligned} \tag{37}$$

$$s.t. 0 \le \mu_{kl} \le 1, \ k = 1, \ldots, K, \ l = 1, \ldots, L \tag{38}$$

$$\mu_{kl} = 1, \text{ if } B_{kl} = +1 \tag{39}$$

$$\mu_{kl} = 0, \text{ if } B_{kl} = -1 \tag{40}$$

where $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L]$. Clearly, $\boldsymbol{\mu}_l$ in the above optimization problem is independent of each other, and could therefore be optimized separately. We use projected gradient descent to solve the above optimization problem.

3.3 Decoding

Given the learned Bernoulli parameters $\boldsymbol{\mu}$, the inference targets to find the label k^* that maximizes the conditional probability:

$$k^* = \arg\max_k \mathbb{P}(y = k | \mathbf{w}_1, \ldots, \mathbf{w}_L, \mathbf{x}, \boldsymbol{\mu}) \tag{41}$$

Clearly, once the Bernoulli parameters $\boldsymbol{\mu}$ are obtained, decoding should take time that scales linearly with the number of columns in the coding matrix, which could be as small as $L = O(\log K)$. It should be noted that in order to enforce sparsity in the coding matrix, we usually pick $L = C \log K$, where C is a constant usually picked around 10 (Allwein et al. 2001). Still, our proposed probabilistic decoding is very efficient, especially when K is large, making it promising for large-scale multi-way classification. Moreover, besides decoding the most probable class label for an instance $\mathbf{x}$, the

Table 2 Datasets details

Dataset	#Class	#Train	#Test	#Feature
Flower	462	170 K	170 K	170,006
Food	1308	467 K	467 K	170,006
SUN	397	19,850	19,850	170,006
ImageNet	15,952	2.5M	0.8M	338,163

probabilistic decoding approach also naturally assigns confidence to each class, which will prove important when we not only want to generate a single label for instance $\mathbf{x}$ with potentially high risk, especially when the confidence gap between several classes is small.

4 Experiments

In this section, we test the performance of *sparse output coding* on two datasets: ImageNet (Deng et al. 2009) for object recognition, and SUN database (Xiao et al. 2010) for scene recognition.

4.1 Datasets and Feature Representations

Details of the datasets used in our experiments are provided in Table 2.

4.1.1 Object Recognition on ImageNet

We start with two subtrees in ImageNet, with the root node being *flower* and *food*, respectively. The *flower* image collection contains a total of 0.34 million images covering 462 categories, and the *food* dataset contains a total of 0.93 million images covering 1308 categories. For both datasets, we randomly pick 50 % of images from each class as training data, and test on the remaining 50 % images. Besides, we also carry out experiments on the entire ImageNet data. Specifically, we follow the same experimental protocols specified by Bengio et al. (2010), Weston et al. (2011), where the dataset is randomly split into 2.5 million images for training, 0.8 million for validation, and 0.8 million for testing, removing duplicates between training, validation and test sets by throwing away test examples which had too close a nearest neighbor in training or validation set (Bengio et al. 2010).

4.1.2 Scene Recognition on SUN Database

The SUN database is by far the largest scene recognition dataset, with 899 scene categories. We use 397 well-sampled categories to run the experiment (Xiao et al. 2010). For each class, 50 images are used for training and the other 50 for test.

4.1.3 Feature Representations

For *flower*, *food* and *SUN* datasets, we employ the same feature representation for images as in Lin et al. (2011). Specifically, we compute SIFT (Lowe 2004) descriptors for each image, and then run *k-means* clustering on a random subset of 1 million SIFT descriptors to form a visual vocabulary of 8192 visual words. Using the learned vocabulary, we employ *Locality-constrained Linear Coding (LLC)* (Lin et al. 2011) for feature coding. Finally, a single feature vector is computed for each image using max pooling on a spatial pyramid (Lin et al. 2011). Similar feature engineering is performed on the large *ImageNet* dataset, but with a dictionary containing 16,384 visual words, resulting in 338,163 dimensional feature representation for each image.

4.1.4 Experiment Design and Evaluation

We use *one-vs-rest (OVR)*, one of the most widely applied frameworks for multi-class classification, to serve as a baseline. It is interesting to compare against *OVR* since it is adopted as the major workhorse in several winning systems for multi-class object recognition competitions (Rifkin and Klautau 2004; Lin et al. 2011; Le et al. (2012)). Moreover, we also compare with three output coding based multi-class classification methods: (1) *random dense code output coding (RDOC)* proposed in Allwein et al. (2001) where each element in the code matrix is chosen at random from $\{-1, +1\}$, with probability $1/2$ for -1 and $+1$ each; (2) *random sparse code output coding (RSOC)* in Allwein et al. (2001), where each element in the code matrix is chosen at random from $\{-1, 0, +1\}$, with probability $1/2$ for 0, and probability $1/4$ for -1 and $+1$ each; (3) *spectral output coding (SpecOC)* proposed in Zhang et al. (2009), which builds dense output codes for multi-class classification, using spectral decomposition of the graph Laplacian constructed to measure class similarities. Moreover, to test the impact of *probabilistic decoding* on *SpOC*, we report results of *SpOC* using a simple Hamming distance based decoding strategy, denoted as *SpOC-H*. The third group of methods we compare with are hierarchical classifiers, a popular alternative to large-scale multi-class problems. Specifically, the first hierarchical classifier *(HieSVM-1)* follows a top-down approach, and trains a multi-class SVM at each node in the class hierarchy (Kosmopoulos et al. 2010). The second one *(HieSVM-2)* adopts the strategy in Dekel et al. (2004). Finally, we also compare against the *relaxed tree classifier (relaxTree)* (Gao and Koller 2011), which learns relaxed tree hierarchy and corresponding classifiers in a unified framework via alternating optimization. Furthermore, we report results of two multi-class classification methods designed for problems with a large number of classes: (1) *error-correcting tournaments (ECT)* (Beygelzimer et al. 2009) which also reduces multi-

Table 3 Flat error comparison on *flower*, *food* and *SUN* datasets

Algorithm	Flower (%)			Food (%)			SUN (%)		
	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
OVR	72.77	39.95	27.43	75.02	46.59	34.23	83.38	69.72	61.83
RDOC	86.91	54.78	40.87	88.93	67.47	56.86	88.24	75.09	70.45
RSOC	87.12	53.69	39.04	86.52	66.88	57.45	88.11	75.12	71.83
SpecOC	78.63	47.75	35.91	81.94	58.12	45.70	85.91	72.63	64.38
SpOC-H	72.92	38.77	26.82	72.91	43.56	30.80	83.06	70.54	64.05
HieSVM-1	76.19	–	–	82.76	–	–	87.60	–	–
relaxTree	72.57	–	–	73.86	–	–	**81.09**	–	–
ECT	71.84	–	–	73.52	–	–	82.75	–	–
LET	73.48	40.21	28.13	74.18	45.03	32.46	83.69	69.78	61.97
SpOC	**69.73**	**34.35**	**24.08**	**70.98**	**43.28**	**29.00**	81.78	**68.65**	**61.26**

Bold values correspond to best performance among competing algorithms

class problem into a series of binary classifications; (2) *label embedding tree (LET)* (Bengio et al. 2010) which learns a tree structure of classes by optimizing the overall tree loss, and performs multi-class classification via label embedding. Finally, for the largest dataset in our empirical study, the ImageNet dataset, we also report published accuracies to put our results into context (ideally, we would re-run those algorithms on our machines, however, due to the sheer size of the data, such computation is very expensive). To ensure a fair comparison on the large *ImageNet* data, we also report the results of *SpOC* using the same features as adopted in Bengio et al. (2010), known as visual terms, a high-dimensional sparse vector of color and texture features. We denote the results of *SpOC* using visual terms for image features, as *SpOC-VT*.

For all algorithms except *label embedding tree* and *relaxed tree classifier*, we train *linear SVM* using *averaged stochastic gradient descent* (Bottou 2010). For output coding based methods, we set code length $L = 200$ for *flower* and *SUN*, $L = 300$ for *food*, and $L = 1000$ for ImageNet. Data similarity matrix $\mathbf{S}^D$ is pre-computed with linear kernel and $\alpha = 0.5$. For *RDOC* and *RSOC*, 1000 random coding matrices are generated for each scenario. The coding matrix with the largest minimum pair-wise Hamming distances between codewords, and without identical columns, is chosen. To decode the label for *OVR* using learned binary classifiers, we pick the class with the largest decision value. For *RDOC*, *RSOC*, *SpecOC* and *SpOC-H*, we pick the class whose codeword has minimum Hamming distance with the codeword of test data point. Specifically, for decoding in *RSOC* and *SpOC-H*, we test both strategies of treating zero bits the same way as non-zero ones and ignoring zero bits entirely, and report the best result of these two methods. Finally, for *error-correcting tournaments (a.k.a. filter trees)*, we use the label tree structure learning method in Bengio et al. (2010) to learn a binary tree among classes, because we cannot use the given tree structure associated with datasets as *ECT* requires a binary tree.

Performance is measured using flat error and hierarchical error. For every data point, each algorithm will produce a list of 10 classes in descending order of confidence (except *HieSVM-1*, *relaxTree* and *ECT*, which only provide the most confident class label), based on which the top-n flat error is computed, $n = 1, 5, 10$ in our case. Specifically, flat error equals 1 if the true class is not within the n most confident predictions, and 0 otherwise. On the other hand, hierarchical error reports the minimum height of the lowest common ancestors between true and predicted classes, using the given class hierarchical structure associated with the datasets. For each of the above two error measures, the overall error score for an algorithm is the average error over all test data points.

4.2 Results

Classification results for various algorithms are shown in Tables 3, 4, 5. For the ImageNet data, as the sheer size of the data renders it very expensive to run competing algorithms, we only provide accuracies of our method and *relaxTree* in Table 5. However, we also compare with the accuracies reported in Bengio et al. (2010) using *label embedding tree* and (Weston et al. 2011) using *learning to rank (L2R)*. Moreover, as feature engineering is crucial for image classification, Weston et al. (2011) proposed an *ensemble* approach (in Table 5), combining multiple feature representations, such as spatial and multiscale color and texton histograms, and generating the final classification as an average of 10 separate models.

From results in Tables 3 and 4, we have the following observations. (1) *SpOC* systematically outperforms *OVR*. More interestingly, *OVR* classifier consists of more than 1300 binary classifiers on *food* dataset, while *SpOC* only involves 300 bit predictors. Although previous work has shown better results with *OVR* than output coding on small scale problems (Rifkin and Klautau 2004), with the number of classes increasing to thousands or tens of thousands, and

Table 4 Hierarchical error comparison on *flower*, *food* and *SUN* datasets

	Flower	Food	SUN
OVR	1.95	3.26	1.15
RDOC	2.35	4.39	1.23
RSOC	2.38	4.10	1.24
SpecOC	2.27	4.02	1.21
SpOC-H	1.99	2.97	1.15
HieSVM-1	2.34	3.42	1.28
relaxTree	1.89	3.04	**1.02**
ECT	1.87	**2.89**	1.12
LET	1.98	3.09	1.16
SpOC	**1.69**	2.92	1.06

Bold values correspond to best performance among competing algorithms

hierarchical structure among classes, output coding with a carefully designed code matrix could outperform *OVR*, while maintaining cheaper computational cost, due to the error-correcting property introduced in the code matrix. (2) Both *SpOC* and *OVR* beat *RDOC* and *RSOC*, revealing the importance of enforcing learnability of each bit predictor, since randomly generated code matrix could very likely generate difficult binary separation problems. (3) *SpOC* performs better than *SpecOC*, which employs a dense code matrix. The margin between *SpOC* and *SpecOC* is even more severe on *food*, revealing the importance of having ignored classes in each bit predictor. (4) *SpOC-H* generates inferior results than *SpOC* across the board, indicating the necessity of *probabilistic decoding* to handle zero bits in the code matrix. (5) *SpOC* and *OVR* both outperform *HieSVM-1*, where errors made in the higher level of the class hierarchy get propagated into the lower levels, with no mechanism to correct those early errors. However, the error-correcting property in *SpOC* introduces robustness to errors made in bit predictors. Results for *HieSVM-2* are not available as it runs into out of memory problems on all three datasets. (6) *SpOC* is comparable with *relaxTree* on *SUN* dataset, but outperforms it on the other two datasets. Though *relaxTree* defers the decision on some difficult classes to lower level nodes, hence achieving better accuracy than conventional hierarchical classifier such as *HieSVM-1*, it still lacks the robustness or error correcting ability introduced by the coding matrix in *SpOC*. More interestingly, *relaxTree* involves multiple iterations of learning base classifiers due to the adoption of alternating optimization, and we will compare its time complexity against *SpOC* later this section. (7) *SpOC* beats both *ECT* and *LET*, both of which involve expensive procedure of learning optimal semantic class structure from data, while *SpOC* simply utilizes the class structure associated with the dataset.

According to results in Table 5 for the large *ImageNet* dataset, we see that *SpOC* clearly outperforms *OVR*, consistent with results reported on other datasets. It is interesting

Table 5 Classification accuracy on ImageNet [accuracy of *approximate kNN* is reported by Weston et al. (2011)]

	Accuracy (%)
OVR	2.27
Approximate KNN	1.55
LET (Bengio et al. 2010)	2.54
L2R (Weston et al. 2011)	6.14
relaxTree	5.28
SpOC-VT	9.15
SpOC	**9.46**
Ensemble (Weston et al. 2011)	10.03

Bold value corresponds to best performance among competing algorithms

that *SpOC* also clearly beats *relaxTree*, revealing the importance of error correcting ability in real large-scale multi-class problems. Moreover, comparison with numbers reported in Bengio et al. (2010) and Weston et al. (2011) shows that *SpOC* also outperforms *label embedding tree* and *learning to rank* on the large ImageNet task. Also, using the same features as (Bengio et al. 2010), *SpOC-VT* clearly beats label embedding tree (Bengio et al. 2010) with a significant margin, revealing the effectiveness of our proposed sparse output coding approach. Finally, *SpOC* result is comparable with the accuracy of *ensemble* method, where classification is obtained by combining various kinds of features and averaging over 10 separate models.

Finally, we visualize the coding matrix learned for the large *ImageNet* dataset using our proposed algorithm. Due to the sheer size of the task space, we cannot show the entire coding matrix or the entire bit predictor, as each bit predictor usually involves thousands of original classes. Consequently, we randomly selected 6 bit predictors and show randomly picked classes from both positive and negative partitions. Specifically, Fig. 2 shows the composition of positive partition and negative partition for several bit predictors, and we could already see the similarity within each partition, and distinction between the two partitions.

4.2.1 Remarks on How to Interpret Our ImageNet Result

It is widely acknowledged in vision community Lin et al. (2011), Le et al. (2012) that feature engineering is an important alternative source for boosting image classification accuracy. Recently, with a standard *OVR* classifier system but sophisticated feature engineering that learns feature representations using deep network with 1 billion trainable parameters, strong results surpassing what we report here have been published Le et al. (2012). However, it should be noted that our work focuses on techniques of training superior massive multi-class classifiers under any given feature, and the work of designing state-of-the-art feature representation

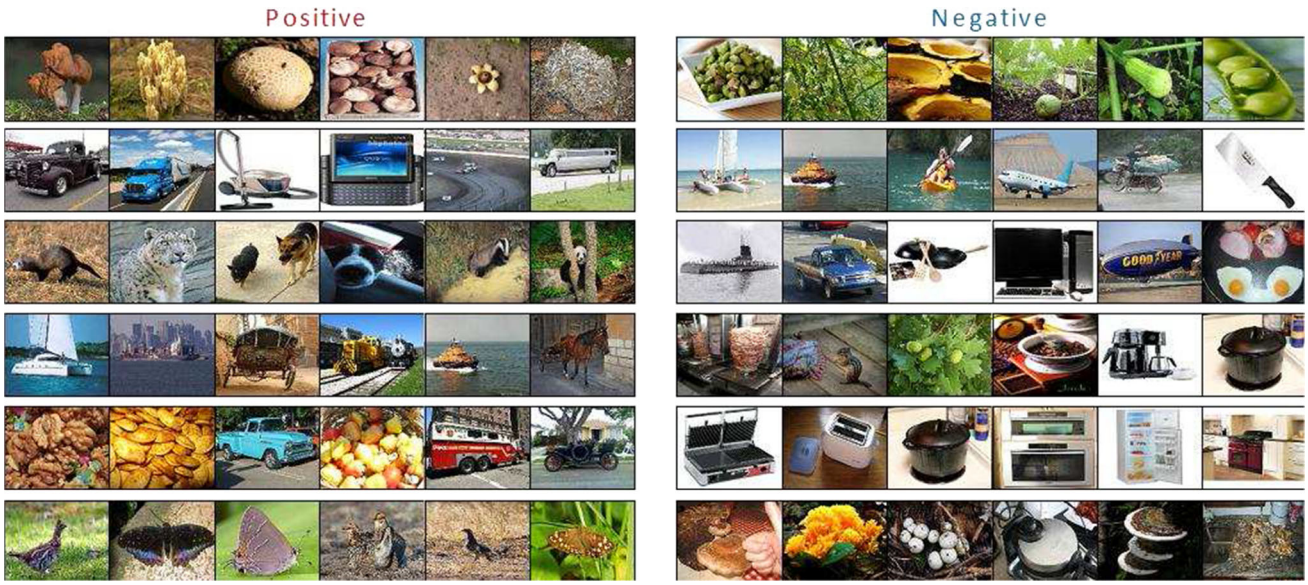

Fig. 2 Visualization of 6 bit prediction problems generated by the learned coding matrix for *ImageNet* with $L = 1000$. Each row corresponds to a binary problem, with the left panel showing categories composing the positive partition, and right panel showing categories composing the negative partition

for images, should not be be viewed as competing efforts, but rather, complementary. Indeed we expect that one can directly apply *SpOC* on top of a superior feature representations learned in previous works to yield even better results. In fact, applying *SpOC* on feature representations learned in Lin et al. (2011) already beats most state-of-the-art results on ImageNet dataset. The superior result in Le et al. (2012) using deep learning for feature design should not discredit *SpOC*'s value as a principled way for massive multi-class classification, as the two approaches focus on different stages in the image classification pipeline. We consider combining *SpOC* with feature representations learned in Le et al. (2012) as an interesting future work.

4.3 Effect of Code Length

In this section, we investigate the effect of code length on classification accuracy of *SpOC*. Specifically, we test *SpOC* on the *flower* dataset with various code lengths and report classification errors in Table 6. According to Table 6, classification error of *SpOC* decreases as the code length increases, as stronger error-correcting ability is accompanied with longer codes. However, the fact that $L = 200$ performs almost as well as $L = 400$ demonstrates that *SpOC* usually requires much less bit predictors compared to the number of classes in the multi-class classification problem.

4.4 Time Complexity

We also report computational time of *SpOC* on the *flower* dataset with various code lengths in Table 6. Specifically, computational time for *SpOC* consists of three parts: (1) time for learning output coding matrix, (2) time for training bit predictors, and (3) time for probabilistic decoding. We implement *SpOC* using MATLAB 7.12 on a 3.40 GHZ Intel i7 PC with 16.0 GB main memory. Bit predictors are trained in parallel on a cluster composed of 200 nodes. Time for training bit predictors is the summation of time spent on each node of the cluster. According to Table 6, time for learning

Table 6 Classification error (flat error) and time complexity (seconds) of SpOC with various code lengths

	$L = 100$	$L = 200$	$L = 300$	$L = 400$
Top 1 (%)	74.71	69.73	68.82	68.79
Top 5 (%)	41.28	34.35	33.70	33.82
Top 10 (%)	30.12	24.08	22.83	22.76
T_c (seconds)	106.4	175.3	237.1	398.6
T_b (seconds)	1.72E7	3.24E7	4.89E7	6.49E7

T_c is the time for learning coding matrix and decoding, and T_b is the time for learning bit predictors. ($1E7 = 1 \times 10^7$)

Table 7 Time complexity comparison (seconds)

	Flower	Food	SUN
OVR	$1.68E8$	$2.63E8$	$1.71E7$
ECT	$5.60E7$	$8.07E7$	$4.28E6$
LET	$5.26E7$	$8.02E7$	$4.15E6$
relaxTree	$1.30E8$	$2.41E8$	$1.09E7$
SpOC	$3.24E7$	$5.02E7$	$3.72E6$

code matrix and probabilistic decoding is almost negligible compared to the time spent on training bit predictors.

Moreover, we also compare the time complexity of *SpOC* with that of *OVR*, *ECT*, *LET* and *relaxTree*. According to Table 7, the total CPU time of *SpOC* is systematically shorter than *OVR*, which is expected as *SpOC* requires training much less binary classifiers than *OVR*, and each bit predictor in *SpOC* only involves a subset of classes from the original problem, while each binary classifier in *OVR* is trained using data from all classes in the multi-class problem. This again reveals the advantage of *SpOC* over the widely popular *OVR* on massive multi-class classification. Moreover, *SpOC* is also more efficient than *ECT* and *LET*. One possible reason could be the expensive tree learning procedure involved in both *ECT* and *LET*, while the time spent on learning optimal code matrix in *SpOC* is negligible compared to the time of training bit predictors. Finally, *SpOC* clearly takes less computational time than *relaxTree*, which requires multiple iterations of learning base classifiers due to the alternating optimization approach in learning relaxed tree classifier.

5 Conclusions and Future Works

Sparse output coding provides an initial foray into real-world scale massive multi-class problem, by turning high-cardinality multi-class classification into a bit-by-bit decoding problem. Effectiveness of *SpOC* is demonstrated on both large scale text classification and image categorization. The fact that *SpOC* takes less bit predictors than *one-vs-rest* multi-class classification while achieving better accuracy, renders our proposed approach especially promising when scaling up to human cognition level multi-class classification.

For future works, one particularly interesting topic is to apply *SpOC* on top of feature representations learned using deep network (Le et al. 2012). Moreover, we would like to study the selection of optimal code word length L, both empirically and theoretically.

References

Allwein, E., Schapire, R., & Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, *1*, 113–141.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for bayesian multitask learning. *The Journal of Machine Learning Research*, *4*, 83–99.

Bengio, S., Weston, J., & Grangier, D. (2010). Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems*, pp. 163–171.

Bergamo, A., & Torresani, L. (2012). Meta-class features for large-scale object categorization on a budget. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR '12)*.

Beygelzimer, A., Langford, J., Lifshits, Y., Sorkin, G., & Strehl, A. (2009). Conditional probability tree estimation analysis and algorithms. In *Conference in Uncertainty in Artificial Intelligence (UAI)*.

Beygelzimer, A., Langford, J., & Ravikumar, P. (2009). Error-correcting tournaments. In *International conference on algorithmic learning theory (ALT)*.

Binder, A., Mller, K. -R., & Kawanabe, M. (2011). On taxonomies for multi-class image categorization. *International Journal of Computer Vision*, 1–21.

Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of nearest-neighbor based image classification. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundation and Trends in Machine Learning*, *3*(1), 1–122.

Budanitsky, A., & Hirst, G. (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, *32*, 13–47.

Cai, L., & Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *CIKM*.

Crammer, K., & Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine Learning*, *2*, 265–292.

Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. In *ICML*.

Deng, J., Berg, A., & Fei-Fei, L. (2011). Hierarchical semantic indexing for large scale image retrieval. In *CVPR*.

Deng, J., Dong, W., Socher, R., Li, L. -J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

Deng, J., Satheesh, S., Berg, A., & Fei-Fei, L. (2011). Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*.

Dietterich, T., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Eckstein, J., & Bertsekas, D. (1992). On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, *55*(1), 293–318.

Escalera, S., Pujol, O., & Radeva, P. (2010). On the decoding process in ternary error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(1), 120–134.

Farhadi, A., Endres, I., Hoiem, D., & Forsyth, D. (2009). Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*.

Fergus, R., Bernal, H., Weiss, Y., & Torralba, A. (2010). Semantic label sharing for learning with many categories. In *ECCV*. Berlin: Springer.

Gabay, D., & Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, *2*(1), 17–40.

Gao, T., & Koller, D. (2011). Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *International Conference on Computer Vision (ICCV)*.

Gao, T., & Koller, D. (2011). Multiclass boosting with hinge loss based on output coding. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* .

Griffin, G., Holub, A., & Perona, P. (2007). Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical report.

Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Proceedings of NIPS*.

Jacob, L., Bach, F., & Vert, J. -P. (2008). Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems NIPS*.

Koller, D., & Sahami, M. (1997). Hierarchically classifying docuemnts using very few words. In *ICML*.

Kosmopoulos, A., Gaussier, E., Paliouras, G., & Aseervatham, S. (2010). The ecir 2010 large scale hierarchical classification workshop. *SIGIR Forum*, *44*(1), 23–32.

Kumar, N., Berg, A., Belhumeur, P., & Nayar, S. (2009). Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*.

Lampert, C., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., & Ng, A. (2012). Building high-level features using large scale unsupervised learning. In *ICML*.

Li, L., Su, H., Xing, E., & Fei-Fei, L. (2010). Object bank: A highlevel image representation for scene classification and semantic feature sparsification. In *Proceedings of NIPS*.

Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., & Huang. T. (2011). Large-scale image classification: fast feature extraction and svm training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1689–1696.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Nilsson, N. (1965). *Learning Machines*. New York: McGraw-Hill.

Parsana, M., Bhattacharya, S., Bhattacharyya, C., & Ramakrishnan, K. (2007). Kernels on attributed pointsets with applications. In *Advances in Neural Information Processing Systems (NIPS)*.

Passerini, A., Pontil, M., & Frasconi, P. (2004). New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, *15*(1), 45–54.

Patterson, G., & Hays, J. (2012). Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Póczos, B., Xiong, L., & Schneider, J. (2011). Nonparametric divergence estimation with applications to machine learning on distributions. In *UAI*.

Pujol, O., Radeva, P., & Vitria, J. (2006). Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(6), 1001–1007.

Rastegari, M., Farhadi, A., & Forsyth, D. (2012). Attribute discovery via predictable discriminative binary codes. In *Computer Vision (ECCV)*. Berlin: Springer.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research*, *5*, 101–141.

Russell, B., Torralba, A., Murphy, K., & Freeman, W. (2008). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*, 157–173.

Sanchez, Jorge, Perronnin, Florent, Mensink, Thomas, & Verbeek, Jakob. (2013). Image classification with the Fisher vector: Theory and practice. *International Journal of Computer Vision*, *105*(3), 222–245.

Schapire, R. (1997). Using output codes to boost multiclass learing problems. In *ICML* .

Schapire, R., & Freund, Y. (2012). *Boosting: Foundations and algorithms*., Adaptive computation and machine learning series Cambridge: MIT Press.

Torralba, A., Fergus, R., & Freeman, W. (2008). 80 Million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*, 1958–1970.

Torresani, L., Szummer, M., & Fitzgibbon, A. (2010). Efficient object category recognition using classemes. In *Computer Vision (ECCV)*.

Wang, G., Hoiem, D., & Forsyth, D. (2009). Learning image similarity from flickr using stochastic intersection kernel machines. In *IEEE 12th International Conference on Computer Vision (ICCV)*.

Weinberger, K., & Chapelle, O. (2008). Large margin taxonomy embedding for document categorization. In *Advances in Neural Information Processing Systems (NIPS)*.

Wen, Z., & Yin, W. (2012). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, pp. 1–38.

Weston, J., Bengio, S., & Usunier, N. (2011). Wsabie: scaling up to large vocabulary image annotation. In *IJCAI*.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., & Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, X., Liang, L., & Shum, H. (2009). Spectral error correcting output codes for efficient multiclass recognition. In *12th International Conference on Computer Vision (ICCV)*.

Zhang, Y., & Schneider, J. (2012). Maximum margin output coding. In *ICML*.

Zhao, B., & Xing, E. (2013). Sparse output coding for large-scale visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, D., Xiao, L., & Wu, M. (2011). Hierarchical classification via orthogonal transfer. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.