Sparse Output Coding for Large-Scale Visual Recognition

Bin Zhao Eric P. Xing
School of Computer Science, Carnegie Mellon University
{binzhao,epxing}@cs.cmu.edu

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.

1. Introduction

A recent trend in visual recognition is the rapid increase of concept space. For example, the SUN [26] database for scene recognition has 899 categories, and ImageNet [10] spans a total number of 21841 image classes. Moreover, the number of categories can even grow further in the future. With such massive number of classes, it is no surprise that classical algorithms such as one-vs-rest, one-vs-one, or kNN, often favored for their simplicity [22, 3], will be brought to their knees not only because of the training time and storage cost they incur [10], but also because of the conceptual awkwardness of such algorithms in massive multi-class paradigms. For example, facing, say, one million classes, should we go ahead and build 1 million classifiers each trained on 1-vs-999999 classes? Just imagine the resultant data imbalance issue at its extreme, let alone the terrible irregularities of the decision boundaries of such classifiers. Clearly, large-scale visual recognition requires new, out-of-box rethinking of classical approaches and more effective yet simple alternatives.

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 decoding scheme to extract the class label.

1.1. Previous Work

The following two lines of research are related to our work.

Large-scale visual recognition: Very recently, we have seen successful attempts in large-scale visual recognition [19, 18, 2, 11]. Both [19] and [18] 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 [19, 18] to yield even better results. On the other hand, [2, 11] 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.

Error Correcting Output Coding: For a K class problem, *error correcting output coding (ECOC)* [1] 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 used to construct bit predictor h_l using training data $\mathbf{Z}_l = \{(\mathbf{x}_1, B_{y_1,l}), \ldots, (\mathbf{x}_m, B_{y_m,l})\}$ with $B_{y_i,l} \neq 0$, for $l = 1, \ldots, L$ (throughout the rest of this paper, we use "bit predictor" to denote the binary classifier associated with a column of the coding matrix). Results in [1] suggest that learning a coding matrix in a problem-dependent way is better than using a pre-defined one. However, strong error-correcting ability alone does not guarantee good classification [8], since the performance of output coding is also highly dependent on the accuracy of the individual bit predictors. Consequently, several approaches [23, 8, 13] op-

timizing 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. Therefore, *SpOC* learns coding matrix and bit predictors separately, but still balances error-correcting ability and bit prediction accuracy in learning the coding matrix. Therefore, *SpOC* is computationally very efficient compared with [23, 8, 13], without sacrificing accuracy.

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 [12] 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. *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.

1.2. 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 hundred or thousand class problems. *SpOC* is robust to errors in bit predictors, simple to parallel, and its computational time scales logarithmically with the number of classes. (2) We propose a new optimization technique: *dual proximal gradient method*, for efficiently solving l_1 regularized optimization with complicated constraints. (3) We propose *probabilistic decoding* to effectively utilize semantic similarity between visual categories for accurate decoding.

2. Coding

For the sake of scalability to large-scale problems, *SpOC* decouples the learning processes of coding matrix and bit predictors. However, our proposed approach still balances the error-correcting ability of the coding matrix and classification accuracy for each resulting bit predictor, by utilizing training data and structure information among classes.

2.1. Formulation

As its most attractive advantage, the coding matrix in output coding is usually chosen for strong error-correcting ability. Besides, since output coding is essentially aggregating discriminative information residing in each bit, learning accurate bit predictors is also crucial for its success. Since the high computational cost associated with methods optimizing coding matrix and bit predictors simultaneously [13] renders them unfavorable in large-scale problems, we propose to use training data and class taxonomy, to provide a measure of separability for each binary partition. Specifically, if some classes are closely related but are given different codes in the l-th 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. Finally, for large-scale multi-class problems, it is crucial to introduce ignored classes, i.e., 0 in the coding matrix. Otherwise, every single class will participate in the training of each bit predictor. As an illustrating example, consider the ImageNet data set with $20K+$ classes. With each of the $20K$ classes participating in learning 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, *sparse output coding* learns optimal coding matrix as follows,

$$\max_{\mathbf{B}} \; F_b(\mathbf{B}) - \lambda_r F_r(\mathbf{B}) - \lambda_c \sum_{l=1}^{L} \|\boldsymbol{\beta}_l\|_2^2 \tag{1}$$

$$s.t. \;\; \mathbf{B} \in \{-1, 0, +1\}^{K \times L} \tag{2}$$

$$\sum_{k=1}^{K} \mathcal{I}_{\{B_{kl}=1\}} \geq 1, \; \sum_{k=1}^{K} \mathcal{I}_{\{B_{kl}=-1\}} \geq 1, \; \forall l = 1, \ldots, L \tag{3}$$

$$\sum_{l=1}^{L} \mathcal{I}_{\{B_{kl} \neq 0\}} \geq 1, \; \forall k = 1, \ldots, K \tag{4}$$

where $\mathcal{I}$ is the indicator function. $F_b(\mathbf{B})$ measures the separability of each binary partition problem associated with columns of $\mathbf{B}$, and reflects the expected accuracy of bit predictors. Moreover, $F_r(\mathbf{B})$ measures codeword correlation, and minimizing $F_r(\mathbf{B})$ ensures the strong error-correcting ability of the resulting coding matrix. The l_2 regularization on each column $\boldsymbol{\beta}_l$ of $\mathbf{B}$ controls the complexity of each bit prediction problem. λ_r and λ_c are regularization parameters controlling the relative importance of the 3 competing objectives. Constraints in (2) ensure that each column of the coding matrix defines a binary partition problem, with the freedom of introducing ignored classes. Constraints in (3) ensure that each bit prediction problem has non-empty positive partition and non-empty negative partition. Finally,

constraints in (4) ensure that each class in the original K-way classification appears in at least one bit prediction problem, so that we could effectively decode this class.

Before presenting details of each part in problem (1), we would like to make clear that the goal and contribution of this work is effective multi-class categorization with large number of classes, where complex methods would fail, and simplicity prevails. As a result, motivation for design of each piece in (1) is balance between effectiveness and efficiency. Although there might be more sophisticated formulations for pieces in (1), they will very likely increase computational cost, ultimately rendering the method incapable of handling large-scale multi-class problems.

2.1.1 $F_b(\mathbf{B})$: Separability of Binary Problem

One key issue in designing the coding matrix is to ensure that the resulting bit prediction problems could be effectively solved. The key motivation of our mathematical formulation is to compute the following two measures using semantic relatedness matrix $\mathbf{S}$ (for example, *rose* is more similar to *tulip* than *truck*) 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 different from the former set which composes the positive partition. Specifically, for the l-th binary partition defined by $\boldsymbol{\beta}_l$, its separability is measured as

$$\begin{aligned} F_b(\boldsymbol{\beta}_l) &= \sum_{k=1}^{K}\sum_{k'=1}^{K}\left(\mathcal{I}_{\{B_{kl}B_{k'l}>0\}}S_{kk'} - \mathcal{I}_{\{B_{kl}B_{k'l}<0\}}S_{kk'}\right) \\ &= \sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kl}B_{k'l}S_{kk'} = \mathbf{e}_K^\top\left[\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top \odot \mathbf{S}\right]\mathbf{e}_K \end{aligned} \tag{5}$$

where $\mathbf{e}_K \in \mathbb{R}^K$ is the all-one vector and $\odot$ is element-wise product. $F_b(\boldsymbol{\beta}_l)$ defined above should subtract $\sum_k S_{kk}$. However, since this quantity is constant and will not affect optimization of $\mathbf{B}$, we omit this step. Finally,

$$\begin{aligned} F_b(\mathbf{B}) &= \sum_{l=1}^{L} F_b(\boldsymbol{\beta}_l) = \sum_{l=1}^{L}\mathbf{e}_K^\top\left[\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top \odot \mathbf{S}\right]\mathbf{e}_K \\ &= \mathbf{e}_K^\top\left[\sum_{l=1}^{L}\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top \odot \mathbf{S}\right]\mathbf{e}_K = \mathbf{e}_K^\top\left[\mathbf{B}\mathbf{B}^\top \odot \mathbf{S}\right]\mathbf{e}_K = tr(\mathbf{B}\mathbf{B}^\top\mathbf{S}) \end{aligned} \tag{6}$$

where $\mathbf{B}\mathbf{B}^\top = \sum_l \boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top$ and $\mathbf{e}^\top(\mathbf{A}\odot\mathbf{B})\mathbf{e} = tr(\mathbf{A}\mathbf{B})$.

Semantic Relatedness Matrix: $\mathbf{S}$ measures similarity between classes using training data and class taxonomy. 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 categorization problem. Several approaches have been proposed to measure class similarity / distance, such as *Hausdorff distance*, *match kernel* [14, 20], divergence between probability distributions [21]. Here we adopt the sum match kernel [14], and define data similarity $\mathbf{S}^D$ between classes i and j as $S_{ij}^D = \frac{1}{|\mathcal{X}_i|}\frac{1}{|\mathcal{X}_j|}\sum_{p=1}^{|\mathcal{X}_i|}\sum_{q=1}^{|\mathcal{X}_j|} K_D(X_p^{(i)}, X_q^{(j)})$, where K_D is a Mercer kernel. Moreover, classes in large-scale multi-class categorization are rarely organized in a flat fashion, but rather with a taxonomical structure [10, 6], such as a tree. Besides, algorithms for learning structure among classes have also been studied [2], although this is beyond the scope of this work. Following [5] 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 $A_{ij} = \text{intersect}(P_i, P_j)/\max(\text{length}(P_i), \text{length}(P_j))$, where P_i is the path from root node to node i and $\text{intersect}(P_i, P_j)$ counts nodes shared by two paths P_i and P_j. We then construct structural similarity matrix $\mathbf{S}^S = \exp(-\kappa(\mathbf{E} - \mathbf{A}))$, where κ is a constant controlling the decay factor, and $\mathbf{E} \in \mathbb{R}^{K\times K}$ is all-one matrix. Finally, semantic relatedness matrix $\mathbf{S}$ is the weighted sum $\mathbf{S} = \alpha\mathbf{S}^D + (1-\alpha)\mathbf{S}^S$ with $\alpha \in [0, 1]$ being the weight. In problems without class taxonomy, we simply set $\alpha = 1$.

2.1.2 $F_r(\mathbf{B})$: Codeword Correlation

Given an example $\mathbf{x}$, the 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 coding 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 the inner products of the corresponding vectors. Thus, codeword correlation of $\mathbf{B}$ could be computed as following:

$$F_r(\mathbf{B}) = \sum_{k=1}^{K}\sum_{k'=1}^{K}\mathbf{r}_k^\top\mathbf{r}_{k'} = \mathbf{e}_K^\top(\mathbf{B}\mathbf{B}^\top)\mathbf{e}_K \tag{7}$$

where $\mathbf{r}_1^\top, \ldots, \mathbf{r}_K^\top$ are row vectors of coding matrix $\mathbf{B}$.

2.1.3 Relaxing the Integer Constraints

In problem (1), the integer constraint (2) on $\mathbf{B}$ makes the problem NP-hard to solve. To enable efficient solution, we follow [8] and relax it to $\mathbf{B} \in [-1, +1]^{K\times L}$. To introduce ignored classes in bit predictors, we further regularize l_1 norm of $\mathbf{B}$. Putting everything together, we get

$$\min_{\mathbf{B}} \; -tr(\mathbf{B}\mathbf{B}^\top\mathbf{S}) + \lambda_r\mathbf{e}_K^\top\mathbf{B}\mathbf{B}^\top\mathbf{e}_K + \lambda_c\sum_{l=1}^{L}\|\boldsymbol{\beta}_l\|_2^2 + \lambda_1\|\mathbf{B}\|_1 \tag{8}$$

$$s.t. \; -1 \le B_{kl} \le 1, \; \forall k = 1, \ldots, K; \forall l = 1, \ldots, L \tag{9}$$

$$\sum_{k=1}^{K}(|B_{kl}| + B_{kl}) \ge 2, \sum_{k=1}^{K}(|B_{kl}| - B_{kl}) \ge 2, \forall l = 1, \ldots, L \tag{10}$$

$$\sum_{l=1}^{L}|B_{kl}| \ge 1, \; \forall k = 1, \ldots, K \tag{11}$$

where $\|\mathbf{B}\|_1 = \sum_{k,l} |B_{kl}|$, and we equivalently reformulate constraints (3) and (4) to (10) and (11).

2.2. Optimization

The difficulty of solving problem (8) lies in two facts: non-smoothness of l_1 regularization on $\mathbf{B}$, and non-convexity of objective and constraints. However, problem (8) has the special structure that the objective is the difference of two convex functions. Specifically, both $f(\mathbf{B}) = \lambda_r \mathbf{e}_K^\top \mathbf{B}\mathbf{B}^\top \mathbf{e}_K + v\lambda_c \sum_{l=1}^{L} \|\boldsymbol{\beta}_l\|_2^2 + \lambda_1 \|\mathbf{B}\|_1$ and $g(\mathbf{B}) = tr\left(\mathbf{B}\mathbf{B}^\top \mathbf{S}\right)$ are convex. Constraints (10) and (11) could also be formulated similarly. Therefore, we propose a *concave-convex procedure* based algorithm, where the non-convexity is handled by *constrained concave-convex procedure (CCCP)* [24, 7], and the non-smoothness is handled using *dual proximal gradient method*.

2.2.1 Constrained Concave-Convex Procedure

Given an initial point $\mathbf{B}^0$, the *CCCP* computes $\mathbf{B}^{t+1}$ from $\mathbf{B}^t$ by replacing $g(\mathbf{B})$ with its first-order Taylor expansion at $\mathbf{B}^t$, i.e., $g(\mathbf{B}^t) + < \nabla g(\mathbf{B}^t), \mathbf{B} - \mathbf{B}^t >$. For non-smooth functions, it has been shown that the gradient should then be replaced by sub-gradient [7, 28]. Hence, the $|B_{kl}|$ term appearing in constraints could be replaced with its first-order Taylor expansion at $\mathbf{B}^t$, i.e., $\text{sign}(B_{kl}^t) B_{kl}$. The resulting optimization problem is as follows,

$$\min_{\mathbf{B}} \quad -2tr(\mathbf{S}\mathbf{B}^t\mathbf{B}^\top) + \lambda_r \mathbf{e}_K^\top \mathbf{B}\mathbf{B}^\top \mathbf{e}_K + \lambda_c \sum_{l=1}^{L} \|\boldsymbol{\beta}_l\|_2^2 + \lambda_1 \|\mathbf{B}\|_1 \tag{12}$$

$$s.t. \quad -1 \le B_{kl} \le 1, \ \forall k = 1, \ldots, K; \forall l = 1, \ldots, L \tag{13}$$

$$2 + \sum_{k=1}^{K} [-1 - \text{sign}(B_{kl}^t)] B_{kl} \le 0, \ \forall l = 1, \ldots, L \tag{14}$$

$$2 + \sum_{k=1}^{K} [1 - \text{sign}(B_{kl}^t)] B_{kl} \le 0, \ \forall l = 1, \ldots, L \tag{15}$$

$$1 - \sum_{l=1}^{L} \text{sign}(B_{kl}^t) B_{kl} \le 0, \ \forall k = 1, \ldots, K \tag{16}$$

Algorithm 1 Learning output coding matrix

Initialize $\mathbf{B}^0$
repeat
  Find B^t as the solution to problem (12);
  Set $t = t + 1$ and get the new problem (12)
until stopping criterion satisfied

2.2.2 Dual Proximal Gradient Method

Denote the objective function in problem (12) as $F(\mathbf{B})$. Although $F(\mathbf{B})$ is convex, it is a non-smooth function of $\mathbf{B}$ due to the l_1 regularization imposed on $\mathbf{B}$, making the problem difficult to solve. Traditional algorithms for non-smooth convex optimization include smoothing techniques where the non-smooth term is approximated by a smooth function, and sub-gradient method. However, smoothing technique will lose the sparsity inducing property of the l_1 regularization, and sub-gradient method is known for slow convergence and difficulty in picking step size. On the other hand, proximal gradient method [15] has been the major workhorse for solving un-constrained non-smooth convex optimization problems, due to its fast convergence and low complexity. Unfortunately, the constraints in (12) renders the problem unsuitable for proximal gradient methods, as we cannot easily project the solution to constraints in problem (12). Therefore, to solve problem (12), we first get its dual, then apply proximal gradient method on the obtained dual, as the constraints in the dual problem are much easier for projection. Specifically, define $\boldsymbol{\beta} = \text{vec}(\mathbf{B}) \in \mathbb{R}^{KL}$ as the vector obtained by stacking columns of $\mathbf{B}$. Problem (12) could be equivalently formulated as

$$\min_{\boldsymbol{\beta}} \ F_s(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1, \ \ s.t. \ \mathbf{A}\boldsymbol{\beta} \le \mathbf{b} \tag{17}$$

with $F_s(\boldsymbol{\beta}) = -2tr(\mathbf{S}\mathbf{B}^t\mathbf{B}^\top) + \lambda_r \mathbf{e}_K^\top \mathbf{B}\mathbf{B}^\top \mathbf{e}_K + \lambda_c \sum_{l=1}^{L} \|\boldsymbol{\beta}_l\|_2^2$, $\mathbf{A} \in \mathbb{R}^{(2KL+2L+K) \times KL}$ and $\mathbf{b} \in \mathbb{R}^{2KL+2L+K}$ are obtained by organizing the constraints in problem (12) according to $\boldsymbol{\beta}$. Due to the difficulty of projection onto constraints $\mathbf{A}\boldsymbol{\beta} \le \mathbf{b}$, traditional proximal gradient method cannot be applied here. To solve this problem, we first split $F_s(\boldsymbol{\beta})$ and $\|\boldsymbol{\beta}\|_1$ into two parts by introducing an additional variable $\mathbf{z}$

$$\min_{\boldsymbol{\beta}, \mathbf{z}} \ F_s(\boldsymbol{\beta}) + \lambda_1 \|\mathbf{z}\|_1, \ \ s.t. \ \mathbf{A}\boldsymbol{\beta} \le \mathbf{b}, \ \mathbf{z} - \boldsymbol{\beta} = 0 \tag{18}$$

The Lagrange for the above problem is:

$$\begin{aligned} g(\boldsymbol{\gamma}, \boldsymbol{\mu}) &= \inf_{\boldsymbol{\beta}, \mathbf{z}} \left\{ F_s(\boldsymbol{\beta}) + \lambda_1 \|\mathbf{z}\|_1 + \boldsymbol{\gamma}^\top (\mathbf{A}\boldsymbol{\beta} - \mathbf{b}) + \boldsymbol{\mu}^\top (\mathbf{z} - \boldsymbol{\beta}) \right\} \\ &= \inf_{\boldsymbol{\beta}} \left\{ F_s(\boldsymbol{\beta}) + (\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})^\top \boldsymbol{\beta} \right\} + \inf_{\mathbf{z}} \left\{ \lambda_1 \|\mathbf{z}\|_1 + \boldsymbol{\mu}^\top \mathbf{z} \right\} - \boldsymbol{\gamma}^\top \mathbf{b} \\ &= -\sup_{\boldsymbol{\beta}} \left\{ -F_s(\boldsymbol{\beta}) - (\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})^\top \boldsymbol{\beta} \right\} + \inf_{\mathbf{z}} \left\{ \lambda_1 \|\mathbf{z}\|_1 + \boldsymbol{\mu}^\top \mathbf{z} \right\} - \boldsymbol{\gamma}^\top \mathbf{b} \\ &= -F_s^*(-(\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})) + \inf_{\mathbf{z}} \left\{ \lambda_1 \|\mathbf{z}\|_1 + \boldsymbol{\mu}^\top \mathbf{z} \right\} - \boldsymbol{\gamma}^\top \mathbf{b} \end{aligned} \tag{19}$$

where F_s^* is the conjugate function of F_s. Moreover, since the dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$

$$\inf_{\mathbf{z}} \left(\lambda_1 \|\mathbf{z}\|_1 + \boldsymbol{\mu}^\top \mathbf{z} \right) = \begin{cases} 0 & \|\boldsymbol{\mu}\|_\infty \le \lambda_1 \\ -\infty & \text{otherwise} \end{cases} \tag{20}$$

Therefore, we get the following dual problem for (17):

$$\min_{\boldsymbol{\gamma} \ge 0, \|\boldsymbol{\mu}\|_\infty \le \lambda_1} \ h(\boldsymbol{\gamma}, \boldsymbol{\mu}) = F_s^*(-(\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})) + \boldsymbol{\gamma}^\top \mathbf{b} \tag{21}$$

Clearly, the constraints in problem (21) are much easier to project than those in (17). In order to utilize projected gradient method to solve problem (21), we need to compute the gradient of the objective function $h(\boldsymbol{\gamma}, \boldsymbol{\mu})$ w.r.t. $\boldsymbol{\gamma}$ and $\boldsymbol{\mu}$:

$$\frac{\partial h(\boldsymbol{\gamma}, \boldsymbol{\mu})}{\partial \boldsymbol{\gamma}} = -\mathbf{A} \nabla F_s^*(-(\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})) + \mathbf{b} \tag{22}$$

$$\frac{\partial h(\boldsymbol{\gamma}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \nabla F_s^*(-(\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})) \tag{23}$$

where $\nabla F_s^*(-(\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})) = \text{argmin}_{\boldsymbol{\beta}} \left\{ F_s(\boldsymbol{\beta}) + (\mathbf{A}^\top \boldsymbol{\gamma} - \boldsymbol{\mu})^\top \boldsymbol{\beta} \right\}$. Moreover, we could reformulate $F_s(\boldsymbol{\beta})$ as follows:

$$F_s(\boldsymbol{\beta}) = \sum_{l=1}^{L} \left\{ -2(\mathbf{S}\mathbf{B}^t)_l^\top \boldsymbol{\beta}_l + \lambda_r \boldsymbol{\beta}_l^\top \mathbf{e}_K \mathbf{e}_K^\top \boldsymbol{\beta}_l + \lambda_c \boldsymbol{\beta}_l^\top \boldsymbol{\beta}_l \right\} \tag{24}$$

where $(\mathbf{SB}^t)_l$ denotes the l-th column of matrix $\mathbf{SB}^t$. Therefore, $\hat{\boldsymbol{\beta}} = \nabla F_s^*(-(\mathbf{A}^\top\boldsymbol{\gamma}-\boldsymbol{\mu}))$ could be calculated as $\hat{\boldsymbol{\beta}} = [\hat{\boldsymbol{\beta}}_1^\top \dots \hat{\boldsymbol{\beta}}_L^\top]^\top$, with

$$\hat{\boldsymbol{\beta}}_l = \frac{1}{2}(\lambda_r \mathbf{e}_K \mathbf{e}_K^\top + \lambda_c I)^{-1}[2(\mathbf{SB}^t)_l - (\mathbf{A}^\top\boldsymbol{\gamma}-\boldsymbol{\mu})_l] \quad (25)$$

where $(\mathbf{A}^\top\boldsymbol{\gamma}-\boldsymbol{\mu})_l$ is the l-th column of the matrix formulated by resizing $\mathbf{A}^\top\boldsymbol{\gamma}-\boldsymbol{\mu}$ into $K \times L$ matrix. Finally, the projected gradient algorithm for solving problem (12) is shown in Algorithm 2, where P represents projection onto the corresponding constraints.

Algorithm 2 Dual proximal gradient for problem (12)

Choose step size $t > 0$, choose initial $\boldsymbol{\gamma}$ and $\boldsymbol{\mu}$
repeat
  Compute $\hat{\boldsymbol{\beta}}$ using Eq. (25)
  $\boldsymbol{\gamma} = P_{\boldsymbol{\gamma}\geq 0}(\boldsymbol{\gamma} - t(\mathbf{b} - \mathbf{A}\hat{\boldsymbol{\beta}}))$; $\boldsymbol{\mu} = P_{\|\boldsymbol{\mu}\|_\infty \leq \lambda_1}(\boldsymbol{\mu} - t\hat{\boldsymbol{\beta}})$
until convergence

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. For example, consider a 5-class problem in Figure 1. Given a test image from class *Husky*, if we treat zero bits the same way as non-zero ones, Hamming 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. On the other hand, ignoring zero bits entirely would discard precious information for decoding. This is especially important when K is large, where we expect a very sparse coding matrix. For example, in Figure 1, 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.

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 Figure 1, class *Husky* is more similar to *(Shepherd, Wolf)* than *(Fox, Tiger)*. The second bit predictor in Figure 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 l-th bit) as a label propagation [29] problem, where the labeled data corresponds to those classes whose codeword's l-th bit is non-zero, and unlabeled data corresponds to those whose l-th 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 l-th binary partition. Combining this prior with the available training data, we formulate the decoding problem in ternary *ECOC* as *maximum a posteriori* estimation.

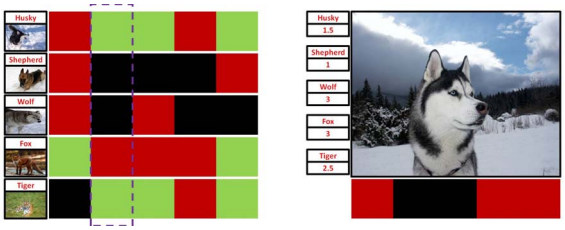

Figure 1. Motivation for probabilistic decoding: (Left). one possible coding matrix for 5-class categorization, with $red = +1$, $black = -1$, and $green = 0$; (Right). one test image from class *Husky*, with its codeword shown in the bottom and Hamming distance with codewords for the 5 classes shown 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 black, rather than red. Best viewed in color.

3.1. Formulation

Given output coding 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 $\{h_1, \dots, h_L\}$. 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 $(c_1, \dots, c_L) \in \{-1, +1\}^L$ 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:

$$\begin{aligned}
&\mathbb{P}(y=k|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}) = \sum_{\{c_l\}} \mathbb{P}(y=k|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}, \{c_l\}) \cdot \mathbb{P}(\{c_l\}|\{\mathbf{w}_l\}, \mathbf{x}, \boldsymbol{\mu}) \\
&= \sum_{\{c_l\}} \mathbb{P}(y=k|\boldsymbol{\mu}, \{c_l\}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&\propto \sum_{\{c_l\}} \prod_l \mathbb{P}(c_l|y=k, \mu_{kl}) \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \sum_{\{c_l\}} \prod_l \mu_{kl}^{c_l}(1-\mu_{kl})^{1-c_l} \prod_l \mathbb{P}(c_l|\mathbf{w}_l, \mathbf{x}) \\
&= \prod_l \{\mu_{kl}\mathbb{P}(c_l=1|\mathbf{w}_l, \mathbf{x}) + (1-\mu_{kl})(1-\mathbb{P}(c_l=1|\mathbf{w}_l, \mathbf{x}))\} \quad (26)
\end{aligned}$$

where $\{c_l\} = \{c_1, \dots, c_L\}$, $\{\mathbf{w}_l\} = \{\mathbf{w}_1, \dots, \mathbf{w}_L\}$, and $\mu_{kl} \in [0, 1]$ is the parameter in Bernoulli distribution $P(c_l = 1|y = k) = \mu_{kl}$. Moreover, given the learned bit predictors, $P(c_l = 1|\mathbf{w}_l, \mathbf{x})$ could be computed using a logistic link function as $P(c_l = 1|\mathbf{w}_l, \mathbf{x}) = 1/(1 + \exp(-\mathbf{w}_l^\top \mathbf{x}))$. Therefore, we need to learn the Bernoulli parameters $\{\mu_{kl}\}$,

which measures the probability of the l-th bit being $+1$ given the true class as $y = k$. Specifically, for the l-th column of the coding matrix, those classes corresponding to $+1$ in the l-th 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 l-th bit. For the example in Figure 1, the second bit predictor separates *(Shepherd, Wolf)* from *Fox*. Clearly, $P(c_l = 1|Shepherd) = P(c_l = 1|Wolf) = 0$ and $P(c_l = 1|Fox) = 1$. Since class *Husky* is not directly involved in this binary classification problem, a non-informative prior would put $P(c_l = 1|Husky) = 0.5$. 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 $P(c_l = 1|Husky) < 0.5$, and in such way those classes with non-zero values in the l-th bit effectively propagate their labels to those initially ignored classes.

3.1.1 Prior Distribution via Label Propagation

Since each column $\boldsymbol{\beta}_l$ in coding matrix has its own labeling (different composition of positive classes, negative classes, and ignored classes), label propagation for each column is independent of others. For l-th column $\boldsymbol{\beta}_l$, consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with K nodes $\mathcal{V} = \mathcal{L} \cup \mathcal{U}$ corresponding to the K classes, where $\mathcal{L}$ corresponds to the labeled classes, and $\mathcal{U}$ corresponds to ignored classes. Our task is then to assign probabilities of being labeled as positive to nodes in $\mathcal{U}$. 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 $\boldsymbol{\beta}_l$. Consequently, for any node $k \in \mathcal{V}$, $\mu_{kl} = P(c_l = 1|y = k)$. 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 [29]: $E(\boldsymbol{\mu}_l) = \frac{1}{2}\sum_{i,j} S_{ij}(\mu_{il} - \mu_{jl})^2$, where $\mathbf{S}$ is the semantic similarity matrix. To assign probability distribution on $\boldsymbol{\mu}_l$, we form Gaussian field $p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C}\exp(-CE(\boldsymbol{\mu}_l))$ [29], where C is inverse temperature parameter, and $Z_C = \int_{\boldsymbol{\mu}_l|\forall k\in\mathcal{L}:\mu_{kl}=\frac{1}{2}(B_{kl}+1)} \exp(-CE(\boldsymbol{\mu}_l))d\boldsymbol{\mu}_l$ is a normalizing constant. Define diagonal degree matrix D as $D_{ii} = \sum_j S_{ij}$ and graph Laplacian $\Delta = \mathbf{D} - \mathbf{S}$, the Gaussian field defined on $\boldsymbol{\mu}_l$ could be equivalently formulated as $p_C(\boldsymbol{\mu}_l) = \frac{1}{Z_C}\exp(-C\boldsymbol{\mu}_l^\top \Delta \boldsymbol{\mu}_l)$, with μ_{kl} clamped to 1 on positive classes and 0 on negative classes.

3.1.2 Parameter Learning

Given L bit predictors, and training data $\mathbf{Z} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, we have that

$$\log\mathbb{P}(Y|\{\mathbf{w}_l\}, \mathbf{X}, \boldsymbol{\mu}) = \sum_{i=1}^{m}\sum_{l=1}^{L}\log\{\mu_{y_i l}\mathbb{P}_{li} + (1-\mu_{y_i l})(1-\mathbb{P}_{li})\} \quad (27)$$

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

$$\min_{\boldsymbol{\mu}} \; -\sum_{i=1}^{m}\sum_{l=1}^{L}\log\{\mu_{y_i l}\mathbb{P}_{li} + (1-\mu_{y_i l})(1-\mathbb{P}_{li})\} + C\sum_{l=1}^{L}\boldsymbol{\mu}_l^\top \Delta \boldsymbol{\mu}_l \quad (28)$$

$$s.t. \;\; 0 \le \mu_{kl} \le 1, \; k = 1, \ldots, K, \; l = 1, \ldots, L \quad (29)$$

$$\mu_{kl} = 1, \text{ if } B_{kl} = +1; \;\; \mu_{kl} = 0, \text{ if } B_{kl} = -1 \quad (30)$$

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.

3.2. Decoding

Given the learned Bernoulli parameters $\boldsymbol{\mu}$, the inference problem targets to find the label k^* that maximizes the conditional probability: $k^* = \arg\max_k P(y = k|\mathbf{w}_1, \ldots, \mathbf{w}_L, \mathbf{x}, \boldsymbol{\mu})$. Clearly, given $\boldsymbol{\mu}$, decoding takes linear time with the number of columns in coding matrix, which could be as small as $L = \mathcal{O}(\log K)$. Hence, our proposed probabilistic decoding is very efficient, and promising for large-scale multi-class categorization.

4. Experiments

In this section, we test the performance of *sparse output coding* on two data sets: ImageNet [10] for object recognition, and SUN database [26] for scene recognition.

Object recognition on ImageNet. We use 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* data set contains a total of 0.93 million images covering 1308 categories. For both data sets, we randomly pick 50% of images from each class as training data, and test on the remaining 50% images.

Scene recognition on SUN database. The SUN database is by far the largest scene recognition data set, with 899 scene categories. We use 397 well-sampled categories to run the experiment [26]. For each class, 50 images are used for training and the other 50 for test.

Feature representations. For each data set, we start with computing dense SIFT 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)* [25] for feature coding. Finally, a single feature vector is computed for each image using max pooling on a spatial pyramid [17].

4.1. Experiment Design and Evaluation

We compare *SpOC* against *one-vs-rest (OVR)*, one of the most widely applied frameworks for large-scale visual recognition. Since class taxonomy exists for all data sets,

Data set	#Class	#Train	#Test	#Feature
Flower	462	169691	169691	170006
Food	1308	467374	467374	170006
SUN	397	19850	19850	170006

Table 1. Data sets details.

we also test two hierarchical classifiers. The first hierarchical classifier *(HieSVM-1)* follows a top-down approach, and trains a multi-class SVM at each node in the class taxonomy [16]. The second one *(HieSVM-2)* adopts the strategy in [9]. Moreover, we also compare with several output coding based methods. Specifically, we compare with the *random dense code output coding (RDOC)* proposed in [1], where each element in the coding matrix is chosen at random from $\{-1, +1\}$, with probability $1/2$ for -1 and $+1$ each. Also, we provide results for the *random sparse code output coding (RSOC)* in [1], where each element in the coding matrix is chosen at random from $\{-1, 0, +1\}$, with probability $1/2$ for 0, and probability $1/4$ for -1 and $+1$ each. Furthermore, we also report results of the algorithm in [27], which builds dense output codes using spectral decomposition *(SpecOC)* of the graph Laplacian constructed using the class taxonomy. Finally, 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*.

For all algorithms, we train *linear SVM* using *stochastic gradient descent* [4]. For output coding based methods, we set code length $L = 200$ for *flower* and *SUN*, and $L = 300$ for *food*. For *SpOC* and *SpOC-H*, we simply set $\lambda_r = 1$, $\lambda_c = \lambda_1 = K$ and $\kappa = 5$ for all data sets. 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 and the one with the largest minimum pair-wise Hamming distances between all pairs of codewords and does not have any 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.

For every data point, each algorithm will produce a list of 10 classes in the descending order of confidence (except *HieSVM-1*, which only provides the most confident class label), based on which the top-n accuracy is computed, $n = 1, 5, 10$ in our case. Specifically, accuracy equals 1 if the true class is within the n most confident predictions, and 0 otherwise. The overall accuracy for each algorithm is the average over the entire test data set.

	$L = 100$	$L = 200$	$L = 300$	$L = 400$
Top 1 (%)	25.36	30.48	31.39	31.34
Top 5 (%)	58.77	65.72	66.02	66.16
Top 10 (%)	69.91	76.38	78.15	78.46

Table 3. Classification accuracy of SpOC vs. code length.

4.2. Results

Classification results for various algorithms are shown in Table 2, with the following observations: (1) *SpOC* systematically outperforms *OVR*. More interestingly, *SpOC* consists of much less binary classifiers than *OVR*. This shows that for large-scale visual recognition problems, output coding with a carefully designed coding matrix could outperform *OVR*, while maintaining cheaper computational cost, due to the error-correcting property introduced in the coding matrix. (2) Both *SpOC* and *OVR* beat *RDOC* and *RSOC*, revealing the importance of enforcing learnability of each bit predictor, since randomly generated coding matrix could very likely generate difficult binary separation problems. (3) *SpOC* performs better than *SpecOC*, which employs a dense coding 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* 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. On the other hand, the error-correcting property in *SpOC* introduces robustness to errors made in bit predictors. (5) *SpOC-H* generates inferior results than *SpOC* across the board, indicating the necessity of *probabilistic decoding*. Finally, *HieSVM-2* runs into out of memory problems on all three data sets.

Effect of code length: To investigate the effect of code length on classification accuracy of *SpOC*, we test *SpOC* on the *flower* data set with various L. According to Table 3, classification accuracy of *SpOC* improves 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 categorization problem.

Time complexity: We compare the time complexity of *SpOC* with *OVR* in Table 4. Specifically, computational time for *SpOC* consists 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 or binary classifiers 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. According to Table 4, time for learning coding matrix and probabilistic decoding is almost negligible compared to the time spent on training bit predictors. Moreover, the total CPU time spent for training bit predic-

	Flower (%)			Food (%)			SUN (%)		
Algorithm	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
OVR	27.23	60.05	72.57	24.98	53.41	65.77	16.62	30.28	38.17
HieSVM-1	23.81	–	–	17.24	–	–	12.40	–	–
RDOC	13.09	45.22	59.13	11.07	32.53	43.14	11.76	24.91	29.55
RSOC	12.88	46.31	60.96	13.48	33.12	42.55	11.89	24.88	28.17
SpecOC	21.37	52.25	64.09	18.06	41.88	54.30	14.09	27.37	35.62
SpOC-H	27.14	61.26	74.33	27.61	55.38	67.32	16.93	29.46	35.83
SpOC	**30.48**	**65.72**	**76.38**	**29.03**	**56.28**	**67.84**	**18.31**	**31.73**	**39.02**

Table 2. Classification accuracy (%) comparison.

	Flower	Food	SUN
OVR	$1.68E8$	$2.63E8$	$1.71E7$
SpOC	$189.4\|3.25E7$	$489.4\|5.14E7$	$182.1\|3.73E6$

Table 4. Time complexity comparison. For *SpOC*, the time (seconds) before | is for learning coding matrix and decoding, and the time after | is for learning bit predictors. ($1E6 = 1 \times 10^6$)

tors in *SpOC* is systematically shorter than *OVR*, revealing the advantage of *SpOC* on large-scale problems.

5. Conclusions

Sparse output coding provides an initial foray into large-scale visual recognition, by turning high-cardinality multi-class classification into a bit-by-bit decoding problem. We also propose *probabilistic decoding* to decode the optimal class label. Effectiveness of *SpOC* is demonstrated on object recognition and scene classification, with hundreds or thousands of classes. The fact that *SpOC* takes less bit predictors than *OVR* while achieving better accuracy, renders it especially promising for large-scale visual recognition.

Acknowledgements

This research is supported by Google, NSF IIS-0713379, and NSF DBI-0640543, ONR N000140910758 and AFOSR FA9550010247.

References

[1] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *JMLR*, 1:113–141, 2001. 1, 7

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010. 1, 3

[3] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 1

[4] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010. 7

[5] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32:13–47, 2006. 3

[6] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, 2004. 3

[7] P. Cheung and J. Kwok. A regularization framework for multiple-instance learning. In *ICML*, 2006. 4

[8] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 2:265–292, 2002. 1, 2, 3

[9] O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML*, 2004. 7

[10] 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. 1, 3, 6

[11] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011. 1

[12] S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *PAMI*, 32(1):120–134, 2010. 2

[13] T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In *ICML*, 2011. 1, 2

[14] D. Haussler. Convolution kernels on discrete structures. *Technical report*, 1999. 3

[15] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *JMLR*, 12:2297–2334, 2011. 4

[16] A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham. The ecir 2010 large scale hierarchical classification workshop. *SIGIR Forum*, 44(1):23–32, 2010. 7

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 6

[18] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 1

[19] 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. 1

[20] M. Parsana, S. Bhattacharya, C. Bhattacharyya, and K. Ramakrishnan. Kernels on attributed pointsets with applications. In *NIPS*, 2007. 3

[21] B. Póczos, L. Xiong, and J. Schneider. Nonparametric divergence estimation with applications to machine learning on distributions. In *UAI*, 2011. 3

[22] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004. 1

[23] R. Schapire. Using output codes to boost multiclass learing problems. In *ICML*, 1997. 1, 2

[24] A. J. Smola, S. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *AISTATS*, 2005. 4

[25] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 6

[26] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 1, 6

[27] X. Zhang, L. Liang, and H. Shum. Spectral error correcting output codes for efficient multiclass recognition. In *ICCV*, 2009. 7

[28] B. Zhao, F. Wang, and C. Zhang. Efficient multi-class maximum margin clustering. In *ICML*, 2008. 4

[29] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003. 5, 6