Augmented Hashing for Semi-Supervised Scenarios

Zalán Bodó and Lehel Csató

Faculty of Mathematics and Computer Science
Babeş–Bolyai University
Cluj-Napoca, Romania

Contact Information:
Faculty of Mathematics and Computer Science
Babeş–Bolyai University
Kogălniceanu 1, 400084 Cluj-Napoca, Romania

Email: {zbodo, lehel.csato}@cs.ubbcluj.ro

Motivation

1. improve on ANN methods
2. give a *generic* method on how to extend hash codes when some *labels* of the data are known
   - is given an unsupervised hash code generator
   - data is defined over some categories (e.g. tweets about music, technology, politics, etc.)
   - labels/categories of some data points are known

Introduction

Learned binary embeddings are used to index large data sets for efficient approximate nearest-neighbor (ANN) search. The embeddings are designed to approximately preserve similarity in the embedding Hamming space, thus leading to efficient Hamming distance computations for finding the nearest-neighbors.

We differentiate between two problems of ANN search with binary embeddings: the first one consists of generating the binary codes, and the second one is the actual searching process [4]. Here we address the first problem: we propose a general framework for augmenting hash codewords obtained by unsupervised techniques. We assume that we are given some class labels for the training data, thus creating a semi-supervised learning scenario. We propose to extend the codewords using error correcting output coding (ECOC) [3] with semi-supervised classifiers.

Augmenting the codewords

Outline of the method:

- form a *data-dependent* error correcting output coding matrix
- train *semi-supervised* classifiers that will generate the second part of the hash code

A coding matrix is a $k \times s$ matrix defined over the set $\{-1, 1\}$, where k denotes the number of classes and s is the codeword length. For each column of the coding matrix a binary classifier is trained, splitting the training data into two sets – of positive and negative examples – based on the actual column.

Error correcting codes

Error correcting output coding is used in machine learning to perform multi-class classification using binary classifiers. At prediction each of the s classifiers outputs a sign, the closest codeword to the resulting vector is looked up in the coding matrix, and the resulting class is output as the decision.

Two-versus-rest scheme (for $k = 4$ classes):

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

Good error correcting codes = *good* row and column separation.

If sufficient labeled data is present, one can define an optimization problem for finding such codes $(\mathbf{c}_i, i = 1, \ldots, k)$ [7]:

$$\min_{\mathbf{C} \in \mathbb{R}^{k \times s}} \sum_{i,j=1}^{k} a_{ij} \|\mathbf{c}_i - \mathbf{c}_j\|^2 \left(= \operatorname{tr}(\mathbf{C}'\mathbf{L}\mathbf{C})\right)$$
$$\text{s.t. } \mathbf{C}'\mathbf{1} = \mathbf{0}, \quad \mathbf{C}'\mathbf{C} = \mathbf{I}$$

Important question: how to calculate *class similarities* ($\mathbf{A} = (a_{ij})_{i,j=1,\ldots,k}$)?

Two approaches used in the experiments:

1. compute class centers and calculate similarities using dot products of the centers
2. train $k(k-1)/2$ SVMs (for each class pair) and use the inverse of the margin of the separating hyperplane as similarity measure – this will reflect how well-separated the classes are:

$$\|\mathbf{w}\|^2 = \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

Linear spectral hashing, Laplacian regularized least squares, semi-supervised SVMs

Linear spectral hashing

The optimization problem of spectral hashing [6] can be written as

$$\min_{\mathbf{B} \in \mathbb{R}^{N \times r}} \operatorname{tr}(\mathbf{B}'\mathbf{L}\mathbf{B})$$
$$\text{s.t. } \mathbf{B}'\mathbf{1} = \mathbf{0}, \quad \mathbf{B}'\mathbf{B} = \mathbf{I}$$

where r denotes the length of the codeword, $\mathbf{B}$ contains the codewords $\mathbf{b}_i'$ in its rows, and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian. Linear spectral hashing [2], a linear variant of spectral hashing – for cases when the dot product offers a good similarity measure – proposes a simple method to compute the codewords for previously unseen points, based on normalized cuts. The algorithm reduces to finding the first r eigenvectors $\{\mathbf{u}_2, \mathbf{u}_3, \ldots, \mathbf{u}_{r+1}\}$ of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}'$, starting with the second largest eigenvalue, where $\mathbf{X}$ denotes the training data set (training instances are put in the columns of $\mathbf{X}$). The codeword of a point is then computed as

$$[\mathbf{x}'\mathbf{u}_2, \mathbf{x}'\mathbf{u}_3, \ldots, \mathbf{x}'\mathbf{u}_{r+1}]'$$

Semi-supervised learners: LapRLS and semi-supervised SVM

General setting: we are given a training data set split into two parts, $\{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathcal{C}, i = 1, 2, \ldots, \ell\} \cup \{\mathbf{x}_j \mid j = \ell + 1, \ldots, \ell + u =: N\}$, where usually $\ell \ll u$.

Laplacian regularized least squares [1] are hyperplane-based regression classifiers for semi-supervised learning, minimizing the error between the hyperplane projections and the known labels, whilst the regularization tag imposes smoothness conditions on the solutions:

$$\min_{\mathbf{W}} \frac{\alpha}{N^2} \sum_{i,j=1}^{N} a_{i,j} \|\mathbf{W}'\mathbf{x}_i - \mathbf{W}'\mathbf{x}_j\|^2 + \frac{\beta}{\ell} \sum_{i=1}^{\ell} \|\mathbf{W}'\mathbf{x}_i - \mathbf{y}_i\|^2 + \gamma \|\mathbf{W}\|_F^2$$

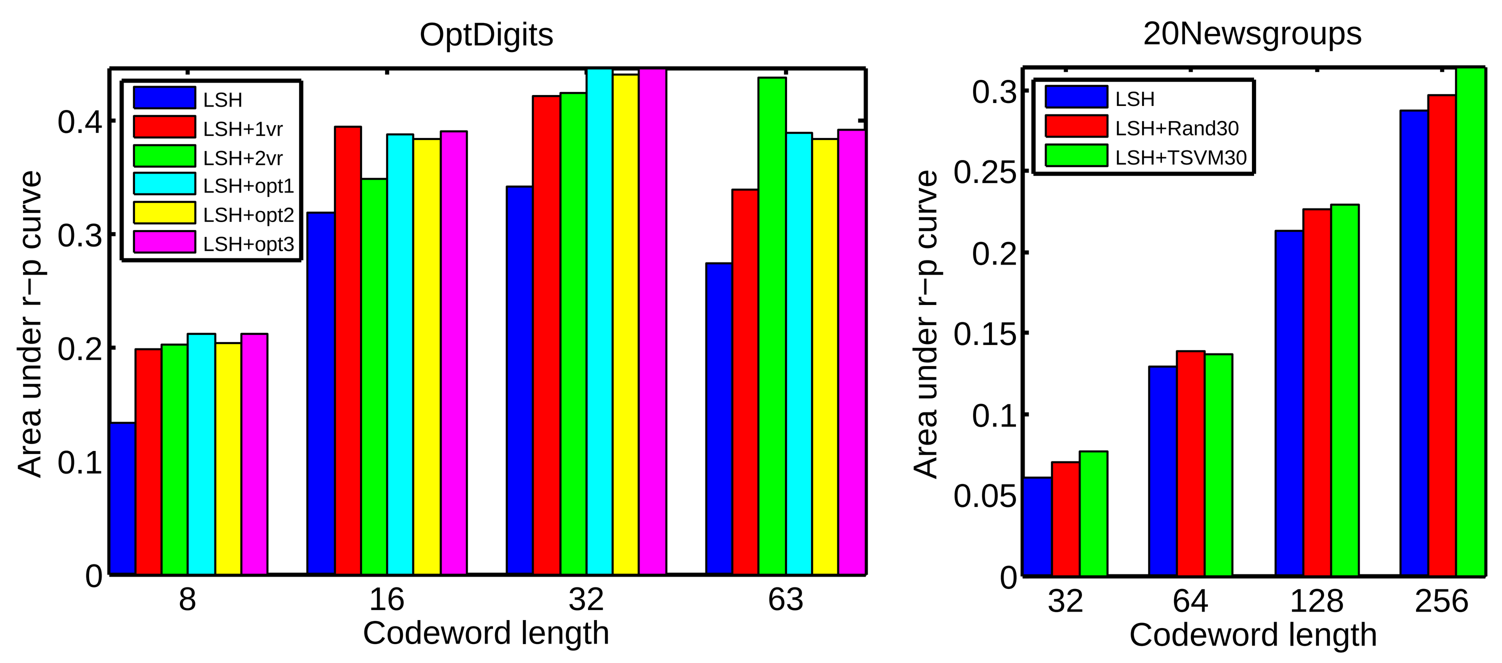
Semi-supervised SVMs [5] appends an additional term to the objective function of the SVM to drive the separating hyperplane towards low density regions:

$$\min_{\mathbf{w}, \{y_j\}_{j=\ell+1}^{N}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{2\ell} \sum_{i=1}^{\ell} l(y_i \mathbf{w}'\mathbf{x}_i) + \frac{\lambda'}{2u} \sum_{j=\ell+1}^{N} l(y_j \mathbf{w}'\mathbf{x}_j)$$
$$\text{s.t. } \frac{1}{u} \sum_{j=\ell+1}^{N} \max(0, \operatorname{sgn}(\mathbf{w}'\mathbf{x}_j)) = t$$

Experiments

Experiments were performed on the following data sets:

Data set	Classes	Dimensionality	Training data (labeled)	Training data (unlabeled)	Test data
OptDigits	10	64	2000	1823	1797
20Newsgroups	20	5000	approx. 200	approx. 11114	7532



(a) (b)

Figure 1: Area under the precision–recall curve for various code lengths: (a) OptDigits, (b) 20Newsgroups data set.

Algorithms used in the experiments:

- LSH – Linear Spectral Hashing
- LSH+1vr – LSH + one-versus-rest approach
- LSH+2vr – LSH + two-versus-rest approach
- LSH+opt1 – LSH + optimized coding matrix using class centroids for calculating class similarities
- LSH+opt2 – LSH + optimized coding matrix using linear SVMs
- LSH+opt3 – LSH + optimized coding matrix using Gaussian SVMs
- LSH+Rand30 – LSH + 30 randomly chosen hyperplanes
- LSH+TSVM30 – LSH + 30 semi-supervised SVMs (coding matrix generated by uniform random class assignment)

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. of Machine Learning Research*, 7:2399–2434, 2006.

[2] Zalán Bodó and Lehel Csató. Linear spectral hashing. In *ESANN*, pages 303–308, 2013.

[3] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. of Artificial Intelligence Research*, 2:263–286, 1995.

[4] Kristen Grauman and Rob Fergus. Learning binary hash codes for large-scale image search. *Machine Learning for Computer Vision*, 411:49–87, 2013.

[5] Vikas Sindhwani and S. Sathiya Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR*, pages 477–484. ACM, 2006.

[6] Yair Weiss, Antonio B. Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, pages 1753–1760. MIT Press, 2008.

[7] Xiao Zhang, Lin Liang, and Heung-Yeung Shum. Spectral error correcting output codes for efficient multiclass recognition. In *ICCV*, pages 1111–1118. IEEE, 2009.