ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Augmented Hashing for Semi-Supervised Scenarios

Zalán Bodó and Lehel Csató *

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

Abstract. Hashing methods for fast approximate nearest-neighbor search are getting more and more attention with the excessive growth of the available data today. Embedding the points into the Hamming space is an important question of the hashing process. Analogously to machine learning there exist unsupervised, supervised and semi-supervised hashing methods. In this paper we propose a generic procedure to extend unsupervised codeword generators using error correcting codes and semi-supervised classifiers. To show the effectiveness of the method we combine linear spectral hashing and two semi-supervised algorithms in the experiments.

1 Introduction

Learned binary embeddings for large data sets, where approximate nearest-neighbors (ANN) of a given point needed to be found, are efficient tools for indexing these sets. The embeddings are designed to approximately preserve similarity in the embedding Hamming space. The beneficial properties of these codewords lead 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 [5]. In this paper we address the first problem.

One can distinguish between unsupervised, supervised and semi-supervised codeword generations, based on the information they use to obtain the embedding [5]. Unsupervised methods use only the information carried by the points themselves. Supervised methods use additional information in form of labels as in a supervised machine learning problem, as well as neighborhood lists or paired constraints. Finally, semi-supervised methods can be viewed as a mixture of the above approaches.

In this paper 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) [4] with semi-supervised classifiers.

*The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

The paper is structured as follows: Sec. 2 describes and presents the main idea of the paper of augmenting the hash codewords obtained by an unsupervised hashing method when label information is available. Sec. 3 presents linear spectral hashing [2], an unsupervised hashing technique, that generates a set of input space hyperplanes, as well as semi-supervised least squares and support vector machines. These methods were chosen to show the effectiveness of the codeword extension in practice. Sec. 4 presents the experiments and discusses the results.

2 Augmenting the codewords

In unsupervised hashing methods supervision information (e.g. class labels) is not used when generating the codewords. In this section we propose a *generic* procedure to extend the codewords generated by an unsupervised method when label information is available. The method is generic because the semi-supervised learning algorithm applied in it can be chosen arbitrarily. Similarly, the base codeword generation method can also be an arbitrary unsupervised technique.

The main idea is simple: form a data-dependent error correcting output coding matrix, based on which 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.[1] In hashing we use the trained classifiers to obtain a better hash codeword.

We know that an error correcting code can detect $d_{min} - 1$ errors and can correct $\lfloor \frac{d_{min}-1}{2} \rfloor$ errors, where d_{min} is the minimum Hamming distance [3].

2.1 Error correcting codes

The most popular technique for multi-class supervised learning with binary classifiers is the *one-versus-rest* approach. This technique needs k classifiers (k denoting the number of classes), each of which is trained choosing a different positive class, while the rest of the training points are considered to constitute the negative class. The one-vs-rest scheme has a minimum codeword distance $d_{min} = 2$, therefore it can detect one error but has no correcting capability.

A very similar but more beneficial scheme is *two-versus-rest*. Here all two-class combinations are taken and the members of the selected class pair are taken as positive examples, whilst the points of the remaining classes are the negative examples. In this case $d_{min} = 2(k - 2)$, therefore $k - 3$ errors can be corrected.

[1] Error correcting output coding is used in machine learning to perform multi-class classification using binary classifiers. At prediction each of the s classifiers output 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.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

For $k = 4$ classes the coding matrix becomes

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

A *good* error correcting code is characterized by row and column separation. Well-separated rows are needed to provide a code with profitable correcting properties – this affects the minimum Hamming distance. Column separation means uncorrelated columns, thus preventing to train redundant classifiers.

To accomplish row and column separation, one approach is to define an optimization problem for finding such codes. We choose the codewords such that the codeword distances be proportional to class similarities. If $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ denote the codewords, we can write the optimization problem as follows [9]:

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

where s denotes the codeword length, and we already relaxed the codewords to real valued vectors. $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denotes the Laplacian, where $\mathbf{A}$ contains the pairwise class similarities and $\mathbf{D} = \operatorname{diag}(\mathbf{A}\mathbf{1})$ is the degree matrix. The constraints are introduced to give a balanced and uncorrelated coding matrix. The solution is given by the s eigenvectors of the Laplacian, starting with the eigenvector corresponding to the second smallest positive eigenvalue.

An important question is how to calculate class similarities. One – and probably the most simple – solution is to compute the class centers and use dot products to compute the similarities. Another approach uses support vector machines (SVM) [9]: the inverse of the margin of the separating hyperplane is used as class similarity – this will reflect how well-separated the classes are. In order to achieve this, $k(k-1)/2$ classifiers are built, one for each class pair. Then the similarity between two classes is given by $\|\mathbf{w}\|^2 = \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{w}$ is the normal of the resulting hyperplane, $\alpha_i, i = 1, 2, \ldots, \ell$ are the Lagrange multipliers from the Wolfe dual of the optimization problem and $k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})'\phi(\mathbf{z})$ is the kernel function. This induces a valid positive similarity measure, by which the Laplacian will be positive semi-definite [7].

3 Linear spectral hashing and Laplacian regularized least squares

This section briefly presents linear spectral hashing for hash codeword generation and two linear semi-supervised learning methods. We have chosen linear spectral hashing because it is linear and the optimization problem to be solved is the same as the problem of finding the optimal coding matrix. The methods can be simply combined by taking the union of the two sets of output vectors.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

3.1 Linear spectral hashing codewords

Linear spectral hashing was proposed in [2] being a linear variant of spectral hashing [8]. The optimization problem of spectral hashing is written as

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

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.[2] The first condition is for maintaining the balance between the bits – bits are distributed evenly over $\{-1,1\}$ – and the second one makes the bits uncorrelated. The solution is given by $\mathbf{B}^*=[\mathbf{v}_2,\ \mathbf{v}_3,\ \ldots,\ \mathbf{v}_{r+1}]$ where $\mathbf{v}_2$ denotes the eigenvector corresponding to the second smallest positive eigenvalue. In its original formulation, using the Gaussian kernel, generalization was done using the eigenfunctions of the weighted Laplace–Beltrami operators, assuming a multidimensional uniform distribution.

Linear spectral hashing – for cases when dot product offer a good similarity measure – proposes a simple and elegant way 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

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

3.2 Semi-supervised learning with Laplacian regularized least squares and semi-supervised SVMs

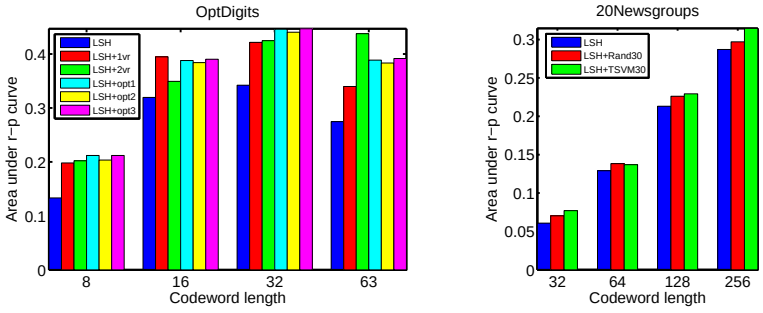
Laplacian regularized least squares [1] are hyperplane-based regression classifiers for semi-supervised learning, minimizing the error between the hyperplane projections and the known labels, whilst the regularization tag imposes smoothness conditions on the solutions. If $\mathbf{W}$ contains the normal vectors $\mathbf{w}_i, i=1,2,\ldots,s$ of the separating hyperplanes in its columns, then the optimization problem of linear LapRLS can be written as:

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

where α and β are the parameters setting the influence of the squared loss and the smoothness penalty, and γ is a regularization parameter. The matrix $\mathbf{W}$ is of size $d\times s$, and $\mathbf{y}_i$ denotes the codeword assigned to the class of $\mathbf{x}_i$ by the methods presented in Sec. 2.

[2] The matrices $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{L}$ denote the same similarity, degree and Laplacian matrix as in Sec. 2.1, but they are defined over different data.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.



(a) (b)

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

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

$$\min_{\mathbf{w},\{y_j\}_{j=\ell+1}^{N}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{2\ell}\sum_{i=1}^{\ell} l(y_i \mathbf{w}'\mathbf{x}_i) + \frac{\lambda'}{2u}\sum_{j=\ell+1}^{N} l(y_j \mathbf{w}'\mathbf{x}_j)$$

$$\text{s.t.} \ \frac{1}{u}\sum_{j=\ell+1}^{N} \max\left(0, \operatorname{sgn}(\mathbf{w}'\mathbf{x}_j)\right) = t$$

where l is the loss function, λ' controls the influence of the unlabeled points and t is the fraction of the unlabeled data required to be positive. We used the Modified Finite Newton Linear L_2-SVM implementation [6][3], where the squared hinge loss is used, $l(z) = l_2(z) = \max(0, 1-z)^2$. For every column of the coding matrix a semi-supervised SVM is built and the corresponding binary label will be given by $\operatorname{sgn}(\mathbf{w}'\mathbf{x})$.

4 Experimental results

The experiments were performed on OptDigits[4], a handwritten digit recognition database, and the 20Newsgroups data set[5] of newswire articles and newsgroup documents. In OptDigits the training (3823) and test examples (1797) are distributed approximately equally among the 10 classes. To create a semi-supervised scenario, we used the labels of only the first 2000 digits from the training set. The 20Newsgroups data set (20 classes, 11314 training and 7532 test documents) was processed as described in [2]. We randomly sampled 10

[3]SVMlin, `http://vikas.sindhwani.org/svmlin.html`

[4]Optical Recognition of Handwritten Digits Data Set, `http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`

[5]`http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz`

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

documents (with replacement) from each class to form the labeled documents, the rest was used as the unlabeled training set.

In Fig. 1 the areas under recall-precision curves are shown. Precision and recall (using 50 nearest neighbors) were calculated as the function of the Hamming neighborhood, and the number of evaluations was determined by the length of the codeword. In Fig. 1(a) the following results are shown: LSH – linear spectral hashing, LSH+1vr – LSH with one-vs-rest scheme, LSH+2vr – LSH with two-vs-rest scheme, LSH+opt1 – LSH with optimized coding matrix using class centroids to calculate class similarity, LSH+opt2 – LSH with optimized coding matrix using linear SVM, LSH+opt3 – LSH with optimized coding matrix using Gaussian SVM. Similarity in LapRLS was calculated using dot products and the normalized Laplacian was used; parameters α and β of LapRLS were set to 1 and q to 10^{-4}. When using optimized coding matrices the normalized Laplacian was used in Eq. (1) and s was set to 64. In LSH+opt3 the parameter σ of the Gaussian kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\sigma\|\mathbf{x} - \mathbf{z}\|^2\right)$ was chosen to be 10^{-3}. Fig. 1(b) compares linear spectral hashing (LSH) with its augmented variants using 30 randomly chosen hyperplanes (LSH+Rand30) and 30 semi-supervised SVM classifiers (LSH+TSVM30). The coding matrix for semi-supervised SVMs was generated by uniform random class assignment, and the parameters were set as $\lambda = 0.01$, $\lambda' = 1$ and $t = 0.5$.

The results show that the augmented codewords outperform the unsupervised codes. It was interesting to note the results of the two-vs-rest scheme. Using optimized coding matrices, however, has the benefit to choose a manageable codeword length when many classes are present. Nonetheless, optimization of the coding matrix is possible only if a larger number of labeled examples are available. Using a semi-supervised method with only a few labeled samples can lead to a costly process with minor improvements over random hyperplanes.

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] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, second edition, 2006.

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

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

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

[7] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

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

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