Modular Dimensionality Reduction

DOI:
10.1007/978-3-030-10925-7_37

Document Version
Accepted author manuscript

Modular Dimensionality Reduction

Henry W J Reeve[1], Tingting Mu[2], and Gavin Brown[2]

[1] University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom
[2] University of Manchester, Oxford Rd, Manchester, M13 9PL, United Kingdom

Abstract. We introduce an approach to modular dimensionality reduction, allowing efficient learning of multiple complementary representations of the same object. Modules are trained by optimising an unsupervised cost function which balances two competing goals: Maintaining the inner product structure within the original space, and encouraging structural diversity between complementary representations. We derive an efficient learning algorithm which outperforms gradient based approaches without the need to choose a learning rate. We also demonstrate an intriguing connection with Dropout. Empirical results demonstrate the efficacy of the method for image retrieval and classification.

Keywords: Ensemble learning · Dimensionality reduction · Dropout · Kernel principal components analysis.

1 Introduction

High dimensional data is a widespread challenge in machine learning applications, from computer vision through to bioinformatics and natural language processing. A natural solution is to find a structure-preserving mapping to a low dimensional space, many techniques for which can be found in the literature, such as kernel PCA, Isomap, LLE and Laplacian Eigenmaps [6, 23]. This paper provides a meta-level tool for modular dimensionality reduction, applicable to each of the aforementioned approaches.

We start from the observation that *multiple* abstractions of the same concept can be taken, and may provide complementary views on a task of interest. We therefore propose a *modular* approach to unsupervised dimensionality reduction, in which we learn a diverse collection of low-dimensional representations of the data. Once a modular representation is learned, each module may be used independently – with their respective predictions combined at test time. This procedure is naturally parallelisable in a distributed computing architecture; and, since each representation is low-dimensional, processing for each module is fast and efficient.

In the context of supervised learning, successful ensemble performance emanates from a fruitful trade-off between the accuracy of the individual members of the ensemble and the degree of diversity [15],[4]. We carry this insight across

to the domain of unsupervised dimensionality reduction, by demonstrating the importance of diversity for a set of representation modules. We introduce an unsupervised loss function for training a *set* of dimensionality reduction modules, which balances two competing objectives. The first objective is for each module to preserve relational structure within the original feature space; the second is for modules to exhibit a *diversity* of relational structures.

The contributions of this paper are as follows:

1. An unsupervised loss function for modular dimensionality reduction.
2. A bespoke optimisation procedure which outperforms gradient based methods such as stochastic gradient descent in our setting.
3. A detailed empirical comparison with competitors.
4. An intriguing connection to the dropout algorithm from deep learning [13].

2 Background

We first review work on dimensionality reduction and ensemble learning.

2.1 Unsupervised Dimensionality Reduction

The canonical approach to unsupervised dimensionality reduction is PCA, and its kernelised generalisation, KPCA [21]. KPCA is a general approach which may be applied to a wide variety of application domains through an appropriate choice of kernel [19, 25]. Several manifold learning techniques have also been shown to be special cases of KPCA, with a data-dependent kernel [12].

Classically, KPCA has been viewed as the orthogonal projection which maximises the preserved variance [21]. We shall adopt an alternative perspective in which we view KPCA as a form of *unsupervised similarity learning*, whereby a mapping is chosen so that inner-products in the low dimensional space approximate the kernel. To make this precise we require some notation. We let $\mathcal{X}$ denote our original feature space and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denote a symmetric positive semi-definite kernel function. Take an unlabelled dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{X}$. For simplicity, we assume throughout that k is centred with respect to $\mathcal{D}$ [21]. Let $\mathbb{H}_k$ denote the associated reproducing kernel Hilbert space (RKHS) of real-valued functions. For each $H \in \mathbb{N}$, we let $\mathbb{H}_k^H$ denote the class of H-dimensional mappings $\boldsymbol{\varphi} : \mathcal{X} \to \mathbb{R}^H$ with coordinate functions taken from the RKHS $\mathbb{H}_k$. That is, for each $\varphi \in \mathbb{H}_k^H$ there exists $\varphi^1, \cdots, \varphi^H \in \mathbb{H}_k$ such that for all $x \in \mathcal{X}$, $\varphi(x) = \left(\varphi^h(x)\right)_{h=1}^H$.

Definition 1 (Inner product loss function). *Given an unsupervised data set $\mathcal{D}$ and a mapping $\boldsymbol{\varphi} \in \mathbb{H}_k^H$, the inner product loss is given by*

$$L_k\left(\boldsymbol{\varphi}, \mathcal{D}\right) = \frac{1}{N^2} \sum_{i,j \leq N} \left(\left\langle \boldsymbol{\varphi}\left(\boldsymbol{x}_i\right), \boldsymbol{\varphi}\left(\boldsymbol{x}_j\right)\right\rangle - k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right)^2.$$

We can interpret KPCA as a form of unsupervised similarity learning which minimises the inner product loss. Let $\xi : \mathcal{X} \to \mathbb{H}_k$ denote the canonical embedding given by $\xi(x)(y) = k(x, y)$.

Proposition 1. *The inner product loss $L_k(\boldsymbol{\varphi}, \mathcal{D})$ is minimised by taking $\boldsymbol{\varphi}$ to be the member of $\mathbb{H}_k^H$ obtained by embedding $\mathcal{D}$ into $\mathbb{H}_k$ via ξ and projecting onto the top H kernel principal components.*

The proofs of all results within the main text are given in the appendices.

2.2 Ensembles and Diversity

Combining the outputs of multiple predictors often brings both statistical advantages, such as bias or variance reduction, and computational advantages, through parallelism. In order outperform an individual model, ensembles promote a level of diversity or disagreement between the predictions the constituent models [10, 15]. Whilst methods such as bagging and boosting encourage diversity through a manipulation of the training data, a more direct approach is the *Negative Correlation Learning* (NCL) algorithm of Liu and Yao [18] in which diversity is targeted explicitly.

Suppose we have have a supervised regression ensemble $\mathcal{H} = \{h_m\}_{m=1}^M$ consisting of predictors h_m. In the previous section we used an unsupervised dataset $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. To distinguish this we use notation $\mathcal{T} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ for a *supervised* dataset. We let $\mathbb{V}(\cdot)$ denote the empirical variance of a finite sequence. The NCL algorithm can be understood in terms of the following *modular loss function.*

Definition 2 (Modular loss function). *The modular loss E_λ is defined by*

$$E_\lambda(\mathcal{H}, \mathcal{T}) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \left(h_m(\boldsymbol{x}_n) - y_n\right)^2 - \lambda \cdot \frac{1}{N} \sum_{n=1}^{N} \mathbb{V}\left(\{h_m(\boldsymbol{x}_n)\}_{m=1}^M\right).$$

The modular loss function consists of two terms: A squared loss term which targets the average individual accuracy of the predictors h_m, combined with a diversity term which encourages disagreement between the predictors. The hyper-parameter λ controls the degree of emphasis placed on the diversity. This has the special property that when $\lambda = 1$, $E_\lambda(\mathcal{H}, \mathcal{T})$ is exactly the squared loss for the ensemble predictor $\frac{1}{M}\sum_m h_m(\mathbf{x})$ from the target y.

The NCL algorithm is equivalent to stochastic gradient descent applied to the modular loss. This perspective differs from original formulation of the NCL algorithm first introduced by Liu and Yao which utilises a multiplicity of interacting cost functions [18]. However, the updates of the two formulations are equal up to a factor of $1/M$ applied to the learning rate.

3 The Modular Inner Product Loss

Our goal is to train a collection of M distinct but complementary representations of the data. With this goal in mind, we introduce the modular inner product loss which combines two contrasting objectives. On the one hand, we seek high quality representations which faithfully preserve the relational structure encoded by the kernel. On the other hand, we would like the relational structure encoded in our different representations to be diverse. Let $\mathcal{F}(H, M)$ denote the class of all M-tuples $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_m\}_{m=1}^M$ with each $\boldsymbol{\varphi}_m \in \mathbb{H}_k^H$. Recall that $\mathbb{V}(\cdot)$ denotes the empirical variance.

Definition 3 (The modular inner product loss). *Suppose we have an unlabelled data set $\mathcal{D} \subset \mathcal{X}$ and a kernel k. Given $\boldsymbol{\Phi} \in \mathcal{F}(H, M)$, the modular inner product loss is given by*

$$L_k^{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right) = \frac{1}{M}\sum_{m=1}^{M} L_k\left(\boldsymbol{\varphi}_m, \mathcal{D}\right) - \lambda \cdot \frac{1}{N^2}\sum_{1 \leq i.j \leq N} \mathbb{V}\left(\{\varphi_m(\boldsymbol{x}_i) \cdot \varphi_m(\boldsymbol{x}_j)\}_{m=1}^M\right). \tag{1}$$

The modular inner product loss is an analogue of the supervised modular loss function (Definition 2), with inner products between a pair of examples in a representation module replacing predictions for a single example, and the target replaced by an unsupervised inner product.

An equivalent reformulation of the modular inner product loss is as a convex combination between the average inner product loss of the individual modules and the inner product loss of a composite representation. Given $\boldsymbol{\Phi} \in \mathcal{F}\left(H, M\right)$ we define $\overline{\boldsymbol{\Phi}} \in \left(\mathbb{H}_k\right)^{H \cdot M}$ by $\overline{\boldsymbol{\Phi}}(\boldsymbol{x}) = \left(1/\sqrt{M}\right) \cdot \left[\boldsymbol{\varphi}_1(\boldsymbol{x})^T, \cdots, \boldsymbol{\varphi}_M(\boldsymbol{x})^T\right]^T$. Proposition 2 is proved in Appendix 9.

Proposition 2. $L_k^{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right) = (1-\lambda) \cdot \frac{1}{M}\sum_{m=1}^{M} L_k\left(\boldsymbol{\varphi}_m, \mathcal{D}\right) + \lambda \cdot L_k\left(\overline{\boldsymbol{\Phi}}, \mathcal{D}\right)$.

When $\lambda = 0$ the loss $L_k^{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right)$ is minimised by taking each $\boldsymbol{\varphi}_m$ to be a projection onto the top H kernel principal components, whilst for $\lambda = 1$, $L_k^{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right)$ is minimised by taking $\overline{\boldsymbol{\Phi}}$ to be the projection onto the top $M \cdot H$ kernel principal components. Hence, $L_k^{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right)$ blends smoothly between training representation modules as individuals and targeting the composite representation.

4 Efficient Optimization

We now introduce the *module-by-module* (MBM) algorithm, which is a form of alternating optimisation designed to minimise the modular inner product loss without the need to choose a learning rate. Our objective is to minimise $L_{\lambda}\left(\boldsymbol{\Phi}, \mathcal{D}\right)$ over $\boldsymbol{\Phi} \in \mathcal{F}\left(H, M\right)$. We require an empirical kernel map.

Definition 4. *A rank R empirical kernel map is a function $\boldsymbol{\psi} \in \mathbb{H}_k^R$ such that $\boldsymbol{\psi}(\boldsymbol{x}_i)^T\psi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for all pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}^2$.*

One can always construct an empirical kernel map of rank N by taking $\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{K}(\mathcal{D})^{-\frac{1}{2}} \left[k(\boldsymbol{x}, \boldsymbol{x}_1), \cdots, k(\boldsymbol{x}, \boldsymbol{x}_N)\right]^T$, where $\boldsymbol{K}(\mathcal{D}) = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{ij}$ denotes the kernel gram matrix. Moreover, given a kernel k we can often obtain a low rank empirical kernel map ψ for a kernel $\tilde{k}$ which closely approximates k by employing a method such as random Fourier features [11] or the Nyström method [27]. By reasoning analogous to [20] we have the following useful proposition.

Proposition 3. *Given a rank R empirical kernel map ψ, the minimum for $L_k^{\lambda}(\boldsymbol{\Phi}, \mathcal{D})$ is attained by $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_m\}_{m=1}^M$ with each $\boldsymbol{\varphi}_m$ of the form $\boldsymbol{\varphi}_m(\boldsymbol{x}) = \boldsymbol{W}_m \cdot \boldsymbol{\psi}(\boldsymbol{x})$ for some matrix $\boldsymbol{W}_m \in \mathbb{R}^{H \times R}$.*

Hence, our objective reduces to the following matrix optimisation problem: Minimise

$$C^{\lambda}(\mathcal{W}, \boldsymbol{\Psi}) := \sum_{m=1}^{M} \left|\left|\boldsymbol{F}_m^T \boldsymbol{F}_m - \boldsymbol{\Psi}^T \boldsymbol{\Psi}\right|\right|^2 - \lambda \cdot \sum_{m=1}^{M} \left|\left|\boldsymbol{F}_m^T \boldsymbol{F}_m - \frac{1}{M} \sum_{q=1}^{M} \boldsymbol{F}_q^T \boldsymbol{F}_q\right|\right|^2$$
$$\propto L_k^{\lambda}(\boldsymbol{\Phi}_{\boldsymbol{\mathcal{W}}}, \mathcal{D}),$$

where $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\boldsymbol{x}_1), \cdots, \boldsymbol{\psi}(\boldsymbol{x}_n)] \in \mathbb{R}^{R \times N}$, $\boldsymbol{F}_m = \boldsymbol{W}_m \cdot \boldsymbol{\Psi}$ and $\boldsymbol{\Phi}_{\boldsymbol{\mathcal{W}}} = \{\boldsymbol{W}_m \cdot \boldsymbol{\psi}\}_{m=1}^M$. We make use the concept of the rank-constrained approximate square root of a symmetric matrix.

Definition 5. *Define $RT_r : \mathbb{R}^{d \times d} \to \mathbb{R}^{r \times d}$ by*

$$RT_r(\boldsymbol{M}) = argmin_{\boldsymbol{F} \in \mathbb{R}^{r \times d}} \left\{\left|\left|\boldsymbol{F}^T \boldsymbol{F} - \boldsymbol{M}\right|\right|^2\right\}.$$

Dax has shown that the rank-constrained approximate square root $RT_r(\boldsymbol{M})$ of any $d \times d$ symmetric matrix $\boldsymbol{M}$ (not necessarily positive semi-definite) may be computed via the singular value decomposition in $O(d^2 \cdot r)$ time and $O(d^2)$ space complexity [7]. The following proposition allows us to optimise the weights of a single module $\boldsymbol{\varphi}_m$, whilst leaving the remaining modules fixed.

Proposition 4. *Suppose we take $m \in \{1, \cdots, M\}$, fix $\boldsymbol{W}_q$ for $q \neq m$, and let*

$$\boldsymbol{T}_m = \frac{M}{M - \lambda \cdot (M-1)} \cdot \boldsymbol{\Psi}^T \left(\boldsymbol{I}_D - \frac{\lambda}{M} \cdot \sum_{q \neq m} \boldsymbol{W}_q^T \boldsymbol{W}_q\right) \boldsymbol{\Psi}.$$

Take $\boldsymbol{F}_m = RT_H(\boldsymbol{T}_m)$. Setting $\boldsymbol{W}_m = \boldsymbol{F}_m \boldsymbol{\Psi}^{\dagger}$ minimises $C^{\lambda}(\mathcal{W}, \boldsymbol{\Psi})$ with respect to $\boldsymbol{W}_m$, under the constraint that $\boldsymbol{W}_q$ remains fixed for $q \neq m$, where $\boldsymbol{\Psi}^{\dagger}$ denotes the psuedo-inverse of $\boldsymbol{\Psi}$.

Unfortunately, computing $\boldsymbol{F}_m$ via Proposition 4 is $O(N^2 \cdot H)$ which is intractable for large N. The following proposition enables us to reduce the complexity of this optimisation whenever we have access to an empirical kernel map of rank $R \ll N$.

Proposition 5. *Suppose that $\boldsymbol{\psi}$ is an empirical kernel map of rank R. Take $\tilde{\boldsymbol{\Psi}} = (RT_R(\boldsymbol{\Psi}\boldsymbol{\Psi}^T))^T \in \mathbb{R}^{R\times R}$. For all $\mathcal{W} = \{\boldsymbol{W}_m\}_{m=1}^M$ with $\boldsymbol{W}_m \in \mathbb{R}^{H\times R}$ we have $C_\lambda\left(\mathcal{W}, \tilde{\boldsymbol{\Psi}}\right) = C_\lambda\left(\mathcal{W}, \boldsymbol{\Psi}\right)$. Moreover, computing $\tilde{\boldsymbol{\Psi}}$ is $O\left(R^2 \cdot N\right)$ in time complexity and $O(R^2)$ in space complexity.*

Combining Propositions 3, 4 and 5 gives rise to the *module-by-module* algorithm (MBM, Algorithm 1), which is $O(NR^2 + EHR^2)$ in time and $O(NR)$ in space complexity, and has the advantage of reducing the modular inner product loss at every iteration until a critical point is reached.

Inputs: A data set $\mathcal{D} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, a rank R empirical kernel map ψ, a number of modules M, a number of dimensions per module H, a diversity parameter λ and $\epsilon > 0$.
Compute $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\boldsymbol{x}_1), \cdots, \boldsymbol{\psi}(\boldsymbol{x}_n)]$;
Update $\boldsymbol{\Psi} = (RT_\rho\left(\boldsymbol{\Psi}\boldsymbol{\Psi}^T\right))^T$;
Randomly initialise $\boldsymbol{F}_m \in \mathbb{R}^{H\times R}$ for $m = 1, \cdots, M$;
Compute $\boldsymbol{Q} = \boldsymbol{\Psi}^T\boldsymbol{\Psi}$ and $\boldsymbol{S} = \sum_{m=1}^M \boldsymbol{F}_m^T\boldsymbol{F}_m$;
Compute $c = ((1-\lambda) + \lambda/M + \epsilon)^{-1}$;
for $e = 1, \cdots, E$ **do**
  for $m = 1, \cdots, M$ **do**
    Compute $\boldsymbol{S}_{-m} = \boldsymbol{S} - \boldsymbol{F}_m^T\boldsymbol{F}_m$;
    Compute $\boldsymbol{T} = c \cdot \left(\boldsymbol{Q} - (\lambda/M)\,\boldsymbol{S}_{-m} + \epsilon \cdot \boldsymbol{F}_m^T\boldsymbol{F}_m\right)$;
    Update $\boldsymbol{F}_m = RT_H\left(\boldsymbol{T}\right)$;
    Update $\boldsymbol{S} = \boldsymbol{S}_{-m} + \boldsymbol{F}_m^T\boldsymbol{F}_m$;
  end
end
Compute $\boldsymbol{W}_m = \boldsymbol{F}_m\boldsymbol{\Psi}^\dagger$ for $m = 1, \cdots, M$;
Output: $\boldsymbol{\Phi} = \{\boldsymbol{W}_m \cdot \boldsymbol{\psi}\}_{m=1}^M$.

Algorithm 1: The module-by-module (MBM) algorithm.

The following theorem justifies the use of the MBM algorithm - it is guaranteed to reduce the modular inner product loss at every epoch until a critical point is reached.

Theorem 1. *Given $E \in \mathbb{N}$, let $\boldsymbol{\Phi}^E \in \mathcal{F}(H, M)$ denote the set obtained by training with Algorithm 1, for E epochs. Then for all $E \in \mathbb{N}$, $L_k^\lambda\left(\boldsymbol{\Phi}^{E+1}, \mathcal{D}\right) < L_k^\lambda\left(\boldsymbol{\Phi}^E, \mathcal{D}\right)$, unless $\boldsymbol{\Phi}^E$ is a critical point of $L_k^\lambda\left(\boldsymbol{\Phi}, \mathcal{D}\right)$, in which case $L_k^\lambda\left(\boldsymbol{\Phi}^{E+1}, \mathcal{D}\right) \leq L_k^\lambda\left(\boldsymbol{\Phi}^E, \mathcal{D}\right)$.*

5 The Dropout Connection

In this section we introduce a surprising connection between the modular inner product loss and the dropout algorithm [13, 22]. Dropout is a state of the art approach to regularising deep neural networks in which a random collection of hidden neurons is "dropped out" at each stochastic gradient update. The dropout algorithm can be understood as implicitly minimising the expectation of a stochastic loss function based on predictions from a random subnetwork [22, 26]. There is a natural analogue of this, in our setting: to minimise the expectation of a stochastic variant of the inner product loss, based on inner products computed from a random subset of modules. We refer to this analogue as the drop-module (DM) algorithm. To be precise, given an ensemble of feature mappings $\boldsymbol{\Phi} \in \mathcal{F}(H, M)$ each binary vector $\boldsymbol{\eta} = \{\eta_m\}_{m=1}^{M} \in \{0,1\}^M$ corresponds to a 'noisy' representation $\boldsymbol{\Phi}_{\boldsymbol{\eta}}$ given by $\boldsymbol{\Phi}_{\boldsymbol{\eta}}(\boldsymbol{x}) = (1/\sqrt{M}) \cdot (\eta_1 \cdot \boldsymbol{\varphi}_1(\boldsymbol{x}), \cdots, \eta_M \cdot \boldsymbol{\varphi}_M(\boldsymbol{x}))$.

Fix a probability $p \in [0,1]$ and let $B(p)$ denote the probability measure on $\{0,1\}$ with $\mathbb{E}_{B(p)}(\eta) = p$. Let $\boldsymbol{\Theta}$ denote the parameters of $\boldsymbol{\Phi}$. The DM algorithm proceeds by randomly sampling $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$ and $\eta_m \sim B(p)$ and updating

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \alpha \cdot \frac{\partial}{\partial \boldsymbol{\Theta}} \left(\left\langle \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\boldsymbol{x}_i), \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\boldsymbol{x}_j) \right\rangle - k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)^2 .$$

The DM algorithm implicitly minimises the following stochastic loss function

$$L_{k,p}^{\text{drop}}(\boldsymbol{\Phi}, \mathcal{D}) = \mathbb{E}_{\boldsymbol{\eta} \sim B(p)^M} \left[L_k(\boldsymbol{\Phi}_{\boldsymbol{\eta}}, \mathcal{D}) \right].$$

Previously, Baldi et al. demonstrated that dropout may be understood as training an exponentially large ensemble with shared weights [1]. In our setting this corresponds to shared weight a ensemble of size 2^M with an ensemble member for each $\boldsymbol{\eta} \in \{0,1\}^M$. We demonstrate that the DM algorithm can be related to an ensemble of size M, trained via the modular inner product loss (see Definition 3). We emphasise that unlike the shared weight ensembles considered by Baldi et al. [1], here we consider ensembles with separate weights in which the interaction takes place purely via the diversity term in the modular inner product loss.

Theorem 2. *The drop-module inner product loss at p is equivalent to the modular inner product loss at $\lambda = Mp/(1+p(M-1))$. To be precise, for $\boldsymbol{\Phi}_0 \in \mathcal{F}(H, M)$ we have*

$$\left. \frac{\partial L_{k,p}^{drop}(\boldsymbol{\Phi}, \mathcal{D})}{\partial \boldsymbol{\Phi}} \right|_{\boldsymbol{\Phi}=\boldsymbol{\Phi}_0} = p \cdot \left. \frac{\partial L_k^{\lambda}(\boldsymbol{\Phi}, \mathcal{D})}{\partial \boldsymbol{\Phi}} \right|_{\boldsymbol{\Phi}=(\sqrt{p/\lambda}) \cdot \boldsymbol{\Phi}_0} .$$

Theorem 2 implies that if we take $\lambda = Mp/(1 + p(M-1))$ then the minima of $L_k^{\lambda}(\boldsymbol{\Phi}, \mathcal{D})$ are equal to the minima of $L_{k,p}^{\text{drop}}(\boldsymbol{\Phi}, \mathcal{D})$, up to a constant scaling factor of $\sqrt{p/\lambda}$. In this sense, the minima for the two loss functions are *representationally equivalent*. The relationship between the diversity parameter λ in the MBM algorithm and the probability of keeping module p in the DM algorithm is illustrated in Figure 1.

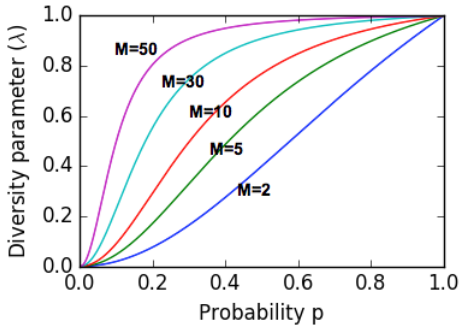

Fig. 1. The diversity parameter λ in MBM vs. the corresponding p in DM.

6 Experimental Results

In this section we first demonstrate the optimisation performance of the MBM algorithm before comparing our method for other natural approaches for training multiple kernelised representations. The data sets used in all experiments are described in Section 12.1.

6.1 Optimisation performance of the MBM algorithm

In this section we assess the MBM algorithm (Algorithm 1) in terms of its efficiency at optimising the modular inner product loss. We compare with two gradient based approaches. As a baseline we consider stochastic gradient descent (SGD) applied directly to the modular inner product loss. This is expected to perform poorly in our setting since the modular inner product loss sums over all pairs of examples. We also consider the state of the art Adam optimiser of Kingma et al. [14] (Adam). The Adam optimiser is applied in batch mode, first compressing the data by applying Proposition 5. We set $M = 10$, $H = 10$, $\lambda = 0.9$ and let k to be the Gaussian kernel with γ set using the the heuristic of Kwok and Tsang [16, Section 4]. In addition, we employ a rank $R = 1000$ Nyström approximation [27]. For SGD and Adam we we consider learning rates in the range $\{10^{-6}, \cdots, 10^{2}\}$. We evaluated the algorithms by training for one hour and evaluating the minimum value of the loss function attained during training and the convergence time - the time taken for the loss function to fall within 1% of its minimum. For SGD and Adam we report results corresponding to the learning rate which achieves the lowest minimum loss. The results are shown in Table 1. The SGD method was extremely slow and typically failed to converge within one hour. The compressed Adam method performed much better and typically converged within 30 minutes. However, the bespoke MBM algorithm achieved the same minimum loss in at most twice the speed on each of the data sets. The MBM algorithm also has the advantage of not requiring the user to set a learning rate.

Table 1. A comparison of the MBM algorithm with gradient based methods.

	Loss function minimum			Convergence time (seconds)		
Data set	SGD	Adam	MBM	SGD	Adam	MBM
Convex	110.0±21.2	**13.4±0.0**	**13.3±0.0**	3369.8±163.8	963.9±368.9	**334.7±151.2**
MNIST	427.6±16.3	**59.9±0.0**	**59.9±0.0**	3484.7±63.2	1609.1±69.2	**400.6±69.4**
NUS Wide	598.6±8.4	**73.1±0.0**	**73.1±0.0**	3376.3±185.2	1495.1±149.5	**422.7±47.7**
NVDR	110.0±18.5	**10.7±0.0**	**10.6±0.0**	1891.3±578.8	1157.2±227.0	**402.4±35.1**
Rectangles	29.4±0.4	**10.3±0.0**	**10.3±0.0**	1796.1±687.6	746.9±162.0	**256.2±36.5**

6.2 Image retrieval & classification performance of MBM modules

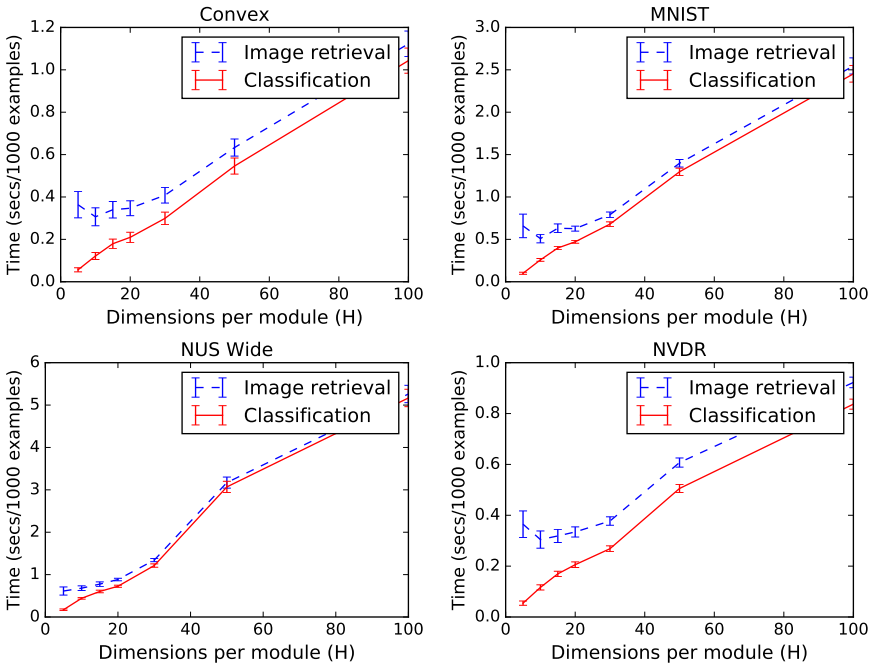

Fig. 2. Test time as vs. H. See Appendix 12.2 for discussion & other data sets.

We compare four unsupervised approaches to training multiple kernelised feature mappings:

Partition We compute the top HM KPCAs and randomly partition these into M sets of H, so that each mapping $\boldsymbol{\varphi}_m \in \mathbb{H}_k^H$ is a projection onto a disjoint subset of the top $H \cdot M$ KPCAs.

Bootstrap Bagging [3] applied to KPCA. For each $m \in \{1, \cdots, M\}$ take a bootstrap sample $\tilde{\mathcal{D}}_m$, of size N, and let $\boldsymbol{\varphi}_m \in \mathbb{H}_k^H$ be the KPCA projection mapping onto H dimensions for $\tilde{\mathcal{D}}_m$.

Random A kernelised variant of the widely used technique of random projections [2, 8]. For each $m = 1, \cdots, M$ we sample a random matrix $\boldsymbol{R}_m \in \mathbb{R}^{H \times D}$ from an $H \times D$ standard normal distribution, and normalise each row so that it has unit norm. In order to kernelise this technique the feature space for the random matrices is the output of the empirical kernel map ψ (see Section 4).

MBM Our proposed approach in which $\boldsymbol{\Phi}$ is trained to minimise the modular inner product loss via MBM algorithm. The diversity parameter λ is set based upon performance on a validation set. We shall consider $H \in \{5, 10, 15, 20, 30, 50, 100\}$ and take M so that $H \cdot M = 300$. We also compare with the following non-modular base line.

Monolithic A single mapping $\boldsymbol{\varphi} \in \mathbb{H}_k^H$ - the projection onto the top 300 KPCAs.

In each case we take k to be the Gaussian kernel with the γ parameter set via the heuristic of Kwok and Tsang [16, Section 4]. For computational efficiency we employ a rank 1000 Nyström approximation [27] in each case. We shall consider two distinct tasks:

Image retrieval: We shall consider the modular low dimensional representation's capability for efficiently retrieving a set of κ close-by images. Let $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_m\}_{m=1}^M \in \mathcal{F}(H, M)$ be a modular representation and $\mathcal{D}$ an unlabelled training set. Given a test point $\boldsymbol{x} \in \mathcal{X}$, for each module, we compute the set $\mathcal{I}_{\kappa,n}^{\varphi_m}(\boldsymbol{x}) \subset \mathcal{D}$ of κ-nearest neighbours of $\boldsymbol{x}$ based the distance $\|\varphi_m(\boldsymbol{x}_q) - \varphi_m(\boldsymbol{x})\|_2$. We then extract subset of size κ, $\mathcal{I}_{\kappa,n}^{\boldsymbol{\Phi}}(\boldsymbol{x}) \subset \bigcup_{m=1}^M \mathcal{I}_{\kappa,n}^{\varphi_m}(\boldsymbol{x})$ so that the elements $\boldsymbol{x}_q \in \mathcal{I}_{\kappa,n}^{\boldsymbol{\Phi}}(\boldsymbol{x})$ minimise the average squared distance from the test point $\boldsymbol{x}$ over the low dimensional spaces ie. $(1/M) \cdot \sum_{m=1}^M \|\varphi_m(\boldsymbol{x}_q) - \varphi_m(\boldsymbol{x})\|_2^2$ is minimised. Let $\mathcal{I}_{\kappa,n}(\boldsymbol{x})$ denote the set of κ nearest neighbours as computed in the original space $\mathcal{X}$. To assess performance we compute the precision: the average value of $(1/\kappa) \cdot \#\left(\mathcal{I}_{\kappa,n}^{\boldsymbol{\Phi}}(\boldsymbol{x}) \cap \mathcal{I}_{\kappa,n}(\boldsymbol{x})\right)$. This procedure is based upon the method of [24] and gives a quantitative assessment of the representation's ability to preserve structural information. The results of the image retrieval task for $\kappa = 10$, $H = 20$ and $M = 15$ are shown in Table 2. On each of the eight data sets the precision attained by the MBM method significantly exceeds the precision attained by the other modular methods: partition, bootstrap and random. Table 3 compares MBM with the monolithic method in which we simply compute the 10 nearest neighbours in the φ-projected space, where φ is the projection onto the top 300 KPCAs. For a relatively modest reduction in performance the MBM method obtains a significant speed up at test time. The speed up is due to the fact that each set of nearest neighbours $\mathcal{I}_{\kappa,n}^{\varphi_m}(\boldsymbol{x})$ may computed in parallel on a low-dimensional space (see Figure 2 and Appendix 12.2). Figure 3 demonstrates

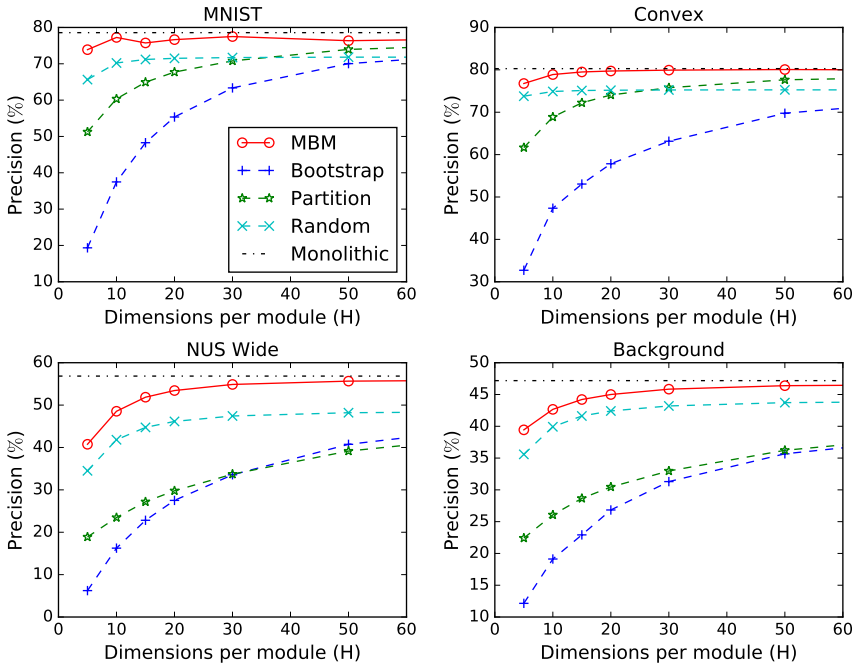

Fig. 3. Precision as a function of H (See Section 6.2).

the precision as a function of the number of dimensions per module (H) with $\kappa = 10$ and $H \cdot M = 300$. As H increases, the precision monotonically approaches the precision attained by the 300-dimensional Monolithic approach. The precision attained by the MBM approach typically exceeds that attained by the other modular approaches (Bootstrap, Partition, Random) across a range of values of H. Corresponding figures for other data sets are given in Appendix 12.3.

Classification We compare the methods in terms of their capacity for extracting multiple sets of features for use in a classification ensemble. Given a modular representation $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_m\}_{m=1}^{M} \in \mathcal{F}(H, M)$, for each m we train a classifier f_m based on the features extracted by φ_m. Given a test point $\boldsymbol{x}$ we combine the outputs of $\{f_m(\varphi_m(\boldsymbol{x}))\}_{m=1}^{M}$ by taking a modal average. Table 2 shows the classification accuracy for ensembles consisting of 15 5-nearest neighbour classifiers trained on 20-dimensional spaces. The MBM approach significantly outperforms the other approaches on five out of eight data sets, and performs comparably or better than the alternatives on every data set. Table 3 compares with the monolithic approach - a single 5-nearest neighbour classifier on 300 KPCAs. The MBM approach is both faster and more accurate than the monolithic method on all but one data set.

Table 2. A comparison of methods for modular dimensionality reduction. See Section 6.2 for details.

	Image retrieval precision (%)				Ensemble classification accuracy (%)			
Data set	Partition	Bootstrap	Random	MBM	Partition	Bootstrap	Random	MBM
NUS Wide	29.8±0.3	27.5±0.3	46.1±0.3	**53.4±0.3**	32.2±0.7	37.7±0.7	42.0±0.7	**43.7±0.7**
NVDR	68.0±0.5	48.1±0.5	73.4±0.5	**77.6±0.4**	**44.9±1.0**	39.6±1.0	**43.9±1.0**	**44.8±1.0**
MNIST	67.7±0.1	55.4±0.1	71.5±0.1	**76.6±0.1**	89.2±0.2	95.0±0.2	94.5±0.2	**95.8±0.1**
Background	30.5±0.2	26.8±0.2	42.4±0.2	**45.0±0.2**	41.0±0.4	47.3±0.4	53.8±0.4	**57.3±0.4**
Random	7.0±0.1	10.9±0.1	6.1±0.1	**17.2±0.1**	40.9±0.4	89.5±0.2	53.1±0.4	**90.3±0.2**
Rotations	59.6±0.2	54.3±0.1	68.4±0.1	**75.3±0.1**	72.0±0.3	85.5±0.3	83.8±0.3	**87.4±0.2**
Convex	74.1±0.1	57.8±0.2	75.2±0.1	**79.7±0.1**	55.7±0.4	**65.5±0.3**	60.3±0.4	**65.6±0.3**
Rectangles	63.3±0.1	44.6±0.1	46.7±0.1	**71.6±0.1**	93.0±0.2	95.5±0.2	93.9±0.2	**98.1±0.1**

Table 3. Comparing the MBM & the Monolithic approach (See Section 6.2).

	Image retrieval				Ensemble classification			
Data set	Monolithic	MBM Δ	Speed	λ	Monolithic	MBM Δ	Speed	λ
NUS Wide	56.8±0.3	-3.4±0.5	10.8×	0.999	40.6±0.7	+3.1±1.0	11.7×	0.990
NVDR	79.0±0.5	-1.4±0.6	7.7×	0.999	40.8±1.0	+3.9±1.4	12.6×	0.999
MNIST	78.6±0.1	-1.9±0.1	9.1×	0.990	95.7±0.1	+0.1±0.2	12.9×	0.900
Background	47.2±0.2	-2.2±0.2	5.8×	0.999	52.5±0.4	+4.9±0.5	8.1×	0.999
Random	23.1±0.1	-5.9±0.1	5.7×	0.950	83.9±0.3	+6.4±0.3	7.1×	0.500
Rotations	76.5±0.1	-1.2±0.1	7.9×	0.990	86.3±0.3	+1.0±0.4	10.9×	0.800
Convex	80.3±0.1	-0.6±0.1	7.2×	0.999	57.6±0.4	+8.0±0.5	23.0×	0.200
Rectangles	70.1±0.1	+1.5±0.2	2.4×	0.990	95.8±0.1	-2.3±0.2	10.7×	0.990

The two Monolothic columns show the image retrieval precision (%) and the classification accuracy (%) of the monolithic method. The MBM columns show the corresponding change in performance due to using the MBM method for each task. The Speed columns show the corresponding speed ups ie. the test time for the Monolithic method divided by the test time for the MBM method.

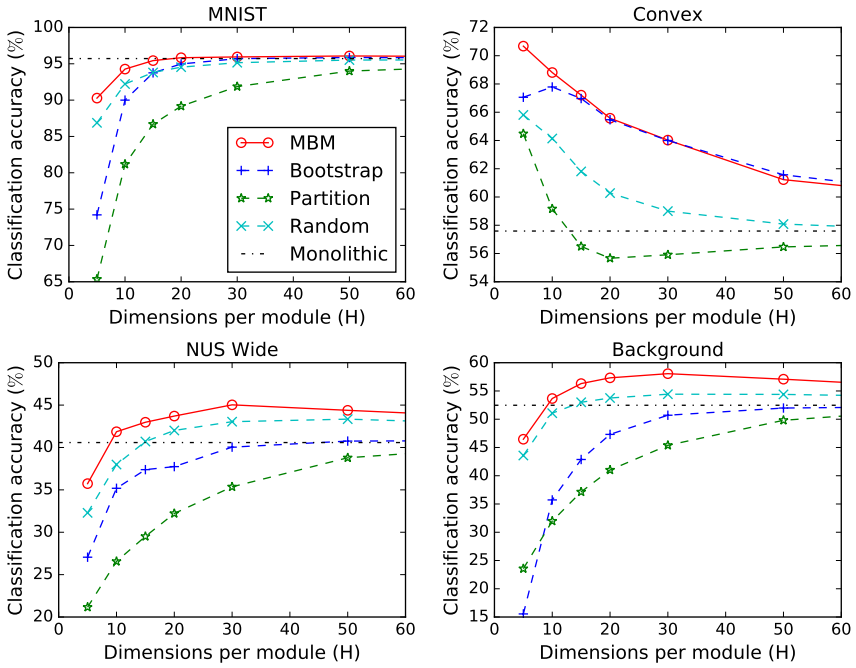

Fig. 4. Classification accuracy as a function of H (See Section 6.2).

The diversity parameter The diversity parameter λ in the MBM algorithm controls the level of emphasis placed upon encouraging a diversity of representations. We found that the optimal performance (both in terms of information retrieval and classification) was typically attained with λ just below 1, with performance declining sharply by taking $\lambda = 1$ (see Figure 5, cols 1& 2). It is interesting to observe that the dropout algorithm often performs well with $p \approx 0.5$ and this corresponds a value of λ just below 1, when M is large (see Figure 1). However, whilst this pattern was observed on all data sets for image retrieval (see Appendix 12.6), for some data sets the best classification performance was attained by taking much lower values of λ (see Figure 5, col 3 and Appendix 12.6). Ultimately, the optimal value of λ is data dependent and must be set based on validation performance.

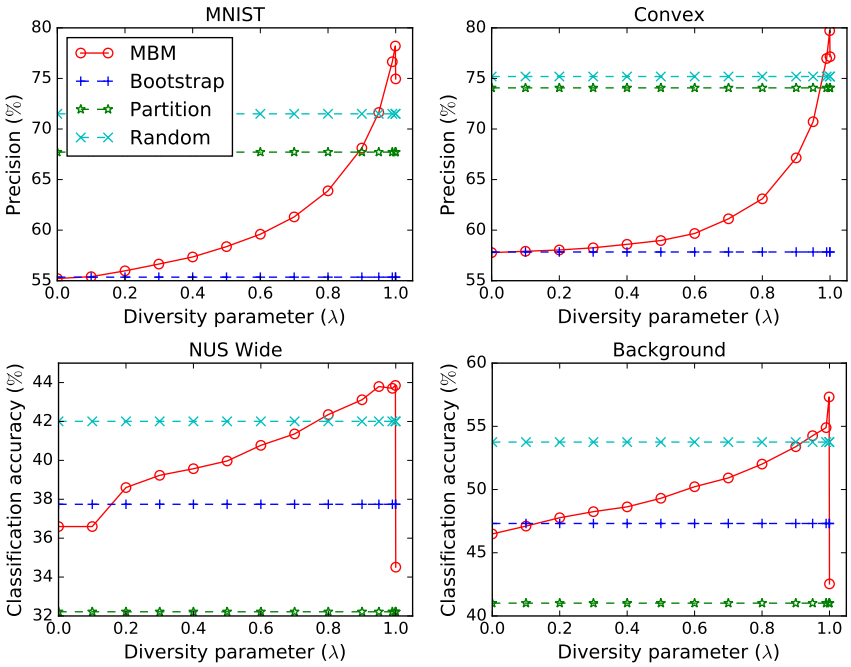

Fig. 5. Performance as a function of the diversity parameter (λ) (see Section 6.2).

7 Discussion

We have investigated a method for *modular* unsupervised dimensionality reduction. Our method is based upon the *modular inner product loss* (Definition 3), an adaptation of concepts from both negative correlation learning [4, 18] and kernel principal components analysis [21]. Whilst the modular loss could be optimised by gradient based methods we introduced a novel *module-by-module* algorithm, which converges at least twice as fast as a state of the art gradient based optimiser [14] without the need to tune the learning rate.

Modular representations have the potential to be applied on range of tasks. Empirical results on both image retrieval and classification tasks confirm that the MBM algorithm is superior to a range of competitors including random projections and bootstrapping, whilst providing a parallelisation advantage over "monolithic" dimensionality reduction. We also demonstrated an intriguing equivalency between our proposal and an analogue of the dropout algorithm - drop module, which deserves further attention.

In summary, this work has shown the potential of explicitly managing diversity in unsupervised representation learning.

References

1. Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.
2. Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.
3. Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
4. Gavin Brown, Jeremy L Wyatt, and Peter Tiňo. Managing diversity in regression ensembles. *The Journal of Machine Learning Research*, 6:1621–1650, 2005.
5. Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.
6. John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16:2859–2900, 2015.
7. Achiya Dax. Low-rank positive approximants of symmetric matrices. *Advances in Linear Algebra & Matrix Theory*, 4(03):172, 2014.
8. Robert J Durrant and Ata Kabán. Random projections as regularizers: Learning a linear discriminant ensemble from fewer observations than dimensions.
9. Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
10. Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario Marchand, and Jean-Francis Roy. Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm. *The Journal of Machine Learning Research*, 16(1):787–860, 2015.
11. Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.
12. Jihun Ham, Daniel D Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on Machine learning*, page 47. ACM, 2004.
13. Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
14. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
15. Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, pages 231–238, 1995.
16. James Tin-Yau Kwok and Ivor Wai-Hung Tsang. The pre-image problem in kernel methods. *Neural Networks, IEEE Transactions on*, 15(6):1517–1525, 2004.
17. Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.
18. Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

19. Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.
20. Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.
21. Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural NetworksICANN'97*, pages 583–588. Springer, 1997.
22. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
23. Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar. A survey of modern questions and challenges in feature extraction. In *Proceedings of The 1st International Workshop on Feature Extraction: Modern Questions and Challenges, NIPS*, pages 1–18, 2015.
24. Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb):451–490, 2010.
25. S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.
26. Sida I Wang and Christopher D Manning. Fast dropout training.
27. Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161322, pages 682–688, 2001.
28. Xiao Wu, Alexander G Hauptmann, and Chong-Wah Ngo. Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 218–227. ACM, 2007.