That was fast! Speeding up NN search of high dimensional distributions.

Emanuele Coviello ECOVIELL@UCSD.EDU
University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Adeel Mumtaz ADEELMUMTAZ@GMAIL.COM
City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

Antoni B. Chan ABCHAN@CITYU.EDU.HK
City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

Gert R.G. Lanckriet GERT@ECE.UCSD.EDU
University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Abstract

We present a data structure for fast nearest neighbor retrieval of generative models of documents based on Kullback-Leibler (KL) divergence. Our data structure, which shares some similarity with Bregman Ball Trees, consists of a hierarchical partition of a database, and uses a novel branch and bound methodology for search. The main technical contribution of the paper is a novel and efficient algorithm for deciding whether to explore nodes during backtracking, based on a variational approximation. This reduces the number of computations per node, and overcomes the limitations of Bregman Ball Trees on high dimensional data. In addition, our strategy is applicable also to probability distributions with hidden state variables, and is not limited to regular exponential family distributions.

Experiments demonstrate substantial speedups over both Bregman Ball Trees and over brute force search, on both moderate and high dimensional histogram data. In addition, experiments on linear dynamical systems demonstrate the flexibility of our approach to latent variable models.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

1. Introduction

Nearest neighbor (NN) search is a core routine in many applications of machine learning and information retrieval. Formally, given a database X, a query q and a dissimilarity d, the problem consists of finding the $x \in X$ for which $d(x, q)$ is minimum. In this work we are particularly interested in the case where the data points represent probability distributions, and the dissimilarity measure is the Kullback-Leibler (KL) divergence (relative entropy). KL has been extensively used to compare generative models of documents, e.g., in text analysis (Pereira et al., 1993; Blei & Lafferty, 2007) and content-based image retrieval (Puzicha et al., 1999; Rasiwasia et al., 2007), where documents are often modeled using histograms, as well as in computer vision for video classification (Chan & Vasconcelos, 2005), where videos are modeled as linear dynamical systems.

Brute-force search is often prohibitive on large datasets. As a consequence, a large body of work, starting from KD-trees (Friedman et al., 1977) and metric-ball trees (Omohundro, 1989; Uhlmann, 1991; Moore, 2000), has investigated the use of spatial data structures to accelerate search. These consist of a hierarchical space decomposition based on geometric properties and database's statistics, that enables fast search by pruning out portions of the search space via a branch and bound exploration. Cayton (2008) extends these ideas to the class of Bregman divergences, of which KL is a member. In particular, Bregman Ball trees (bbtrees) are defined in terms of Bregman balls, and search uses bounds on the Bregman divergence from a query to a Bregman ball. Succesively, Nielsen

et al. (2009a; 2009b) developed an extension of the bbtree to symmetrized Bregman divergences, Zhang et al. (2009) adapted the VA-file and R-tree to decomposable Bregman divergence, and Cayton (2009) used the bbtree for efficient range search. Abdullah et al. (2012) formally analyze approximated Bregman NN search. These accelerated search methods are not immediately practical in high dimensions (Moore, 2000; Cayton, 2008), where the number of close neighbors is often very large, and consequently the procedure needs to run on many nodes, which results in several additional divergence computations and a total computation time that may become comparable or worse than brute force.

The pruning operation of (Cayton, 2008) requires projecting the query onto the shell of a Bregman ball, which is found using a bisection search. However, this has 2 limitations: 1) the procedure is not amenable to distributions outside the exponential family (e.g., latent variable models); 2) two divergence calculations are required for each iteration of the bisection search, which leads to slow performance when the divergence calculation is computationally intensive (e.g., in high dimensions, or latent variable models).

To do more efficient NN search, Bbtrees naturally handle approximate NN operations, by stopping search early after a *fixed budget* of leaves has been visited (Cayton, 2008). This increases efficiency without an excessive degradation of quality.

In this paper we propose a novel branch and bound method based on variational approximations. The corresponding bisection procedure has a small computational overhead, since it requires only one divergence computation at *each backtracked node*, independent of the number of iterations, and then needs to evaluate only scalar functions at each iteration. The algorithm provides a speedup over (Cayton, 2008) on medium and high-dimensionality, and over brute force search at each dimensionality, and returns almost perfect NNs. In addition, our algorithm readily serves search of latent variable distributions.

Other data structures for approximated NN search rely on mapping techniques, for example locality sensitive hashing (reviewed in (Slaney & Casey, 2008)) adapted to non-metric dissimilarities by (Mu & Yan, 2010).

In Section 2 we overview Bregman and KL divergences. In Section 3 we discuss Cayton's (2008) NN search with bbtrees, which lays the basis for our novel contribution, in Section 4. Experiments on histogram data and linear dynamical systems are reported in Sections 5 and 6, and conclusions are drawn in Section 7.

2. Bregman divergence and Kullback-Leibler divergence

This section provides background on Bregman and KL divergence and describes some related properties.

2.1. Bregman divergences

Given a strictly convex differentiable function $f(\cdot)$, the Bregman divergence based on f is $d_f(x,y) = f(x) - f(y) - \langle \nabla f(y), x-y \rangle$. A Bregman divergence $d_f(x,y)$ is convex in x, and the Bregman ball of radius R around μ

$$B(\mu, R) = \{x : d_f(x,\mu) \leq R\} \tag{1}$$

is a convex set. The interested reader may refer to (Cayton, 2008; Banerjee et al., 2005) for more details.

Let f^* be the dual function of f.[1] Since f is strictly convex, f^* is strictly convex as well, and f and $f^*$are Legendre dual of each other (Rockafellar, 1996). One important property is that the gradient ∇f defines a bijective mapping, and the inverse mapping is given by the gradient of the dual function f^*, i.e.:

$$\mu \underset{\nabla f^*(\mu')}{\overset{\nabla f(\mu)}{\rightleftarrows}} \mu' \tag{2}$$

Consistently with the exposition in (Cayton, 2008), we use prime (i.e., $'$) to denote the results of applying ∇f.

2.2. Regular exponential families, regular Bregman divergences and KL divergence

Given a pair of random variables with p.d.f.s $\mathcal{X}$ and $\mathcal{Y}$, respectively, their KL divergence is the functional

$$D(\mathcal{X}||\mathcal{Y}) = \int \mathcal{X}(\omega) \log \frac{\mathcal{X}(\omega)}{\mathcal{Y}(\omega)} d\omega = \mathrm{E}_{\mathcal{X}} \left[\log \frac{\mathcal{X}}{\mathcal{Y}} \right]. \tag{3}$$

The KL divergence is never negative, and is zero iff $\mathcal{X} = \mathcal{Y}$ (a.e.).

There is an interesting correspondence between Bregman divergences and KL divergence, which occurs when we deal with exponential family distributions (Banerjee et al., 2005). In particular, for any *regular* Bregman divergence d_f based on $f(\mu)$, there is associated a *regular* exponential family of distributions

$$\mathcal{F}_{f^*} = \{p_{f^*,\mu'}(\cdot) = \exp\left(\langle \phi(\cdot), \mu' \rangle - f^*(\mu')\right)\}, \tag{4}$$

where $\phi(\cdot)$ are the sufficient statistics, μ' are the canonical parameters, and the dual function f^* is the partition function. The gradient operation ∇f^* maps from

[1] Given $f(x)$, the dual function is defined as $f^*(y) = \sup_x \{\langle x, y \rangle - f(x)\}$

canonical to mean parameters μ (Wainwright & Jordan, 2008). $f(\mu)$ is the negative entropy and its gradient ∇f maps μ to the canonical parameters μ' (Wainwright & Jordan, 2008). d_f is the KL divergences between members of $\mathcal{F}_{f^*}$.

Hence, when $\mathcal{X}$ and $\mathcal{Y}$ are distributions from the same exponential family with mean parameters x and y respectively, we have that

$$D(\mathcal{X}||\mathcal{Y}) = d_f(x,y) = f(x) - f(y) - \langle \nabla f(y), x - y \rangle, \tag{5}$$

where the l.h.s is a functional of two p.d.f.s and the r.h.s. is a function of the mean parameters.

3. Branch and bound for BB trees

In this section we briefly review the branch and bound method for Bregman ball trees by Cayton (2008).

3.1. BB trees

Bregman ball trees and the associated search routine follow the same principles of KD trees (Friedman et al., 1977) and metric ball trees (Omohundro, 1989; Uhlmann, 1991; Moore, 2000), with the difference that they are based on Bregman balls instead of rectangular cells or metric balls. A bbtree consists of a binary tree that partitions a database $X = \{x_1, \ldots, x_n\}$. Every node i is associated with a subset of points $X_i \subset X$, and defines a Bregman ball with center μ_i and radius R_i such that $\forall x \in X_i : x \in B(\mu_i, R_i)$. Each non-leaf node i is associated with left and right child nodes, l and r, and X_i is consequently split between X_l and X_r. The entirety of leaf nodes covers X. A Bregman Ball tree can be grown from the database X in a top-down fashion, recursively using (Banerjee et al., 2005).

3.2. Searching with BB trees

Given a query q, we are interested in its left-NN[2]

$$x_L = \arg\min_{x \in X} d_f(x,q). \tag{6}$$

Search with a BB tree proceeds as follow. The algorithm initially descends the BB tree, starting from the root. At every non-leaf node the algorithm descends through the most promising child node and temporarily ignores the sibling node. Once the algorithm reaches a leaf node X_i, it selects the candidate nearest neighbor $x_c = \arg\min_{x \in X_i} d_f(x_i, q)$.

[2]Cayton (2008) shows that the right-NN can be found using the left-NN algorithm for the divergence d_{f^*} and the database $X' = \{x_1', \ldots, x_n'\}$.

At this point, the algorithm backtracks, and explores an originally ignored sibling j if

$$d_f(x_c, q) > \min_{x \in B(\mu_j, R_j)} d_f(x,q). \tag{7}$$

where the right side of (7) is the Bregman projection of q onto the Bregman ball $B(\mu_j, R_j)$ (see Figure 1(a)). Cayton (2008) proves that the optimal x_p of (7) corresponds to an $x_p{}' = \nabla f(x_p)$ along the line $\theta\mu' + (1-\theta)q'$, $0 \le \theta \le 1$,[3] lays on the shell of $B(\mu_j, R_j)$ and can be found by bisection search over θ.

At step i of the bisection search, given θ_i, a point $x'_{\theta_i} = \theta_i \mu' + (1-\theta_i)q'$ is identified along the line, and the corresponding $x_{\theta_i} = \nabla f^*(x'_{\theta_i})$ is recovered. This in fact consists of computing a left-sided centroid (Nielsen & Nock, 2009)

$$x_{\theta_i} = \arg\min_x \theta_i d_f(x,\mu) + (1-\theta_i) d_f(x,q) \tag{8}$$

$$= \nabla f^*(\theta_i \mu' + (1-\theta_i)q'). \tag{9}$$

3.3. Bregman projection vs. stopping early

Since exact evaluation of the Bregman projection (7) is not needed, Cayton (2008) derives stopping conditions based on upper and lower bounds

$$a \le \min_{x \in B(\mu_j, R_j)} d_f(x,q) \le A. \tag{10}$$

Upper and lower bounds are computed at each iteration of the bisection, until either $d(x_c, q) < a$ (prune the node) or $d(x_c, q) > A$ (explore the node).

The lower bound a is given by weak duality:

$$\mathcal{L}(\theta) = d_f(x_\theta, q) + \frac{\theta}{1-\theta}\left(d_f(x_\theta, \mu) - R\right), \tag{11}$$

where $\mathcal{L}(\theta)$ is the Lagrangian of the right side of (7) and $0 \le \theta \le 1$. The upper bound A comes directly from the primal problem:

$$\textbf{if } x_\theta \in B(\mu_j, R_j) \Rightarrow d_f(x_\theta, q) \ge \min_{x \in B(\mu_j, R_j)} d_f(x,q). \tag{12}$$

Hence the algorithm operates bisection search on θ as follows (see Figure 1(b)). At the i-th step, given θ_i, the algorithm computes the left-sided centroid $\nabla f^*(\theta_i \mu' + (1-\theta_i)q')$ and then checks the bounds. If $\mathcal{L}(\theta_i) > d_f(x_c, q)$, it prunes the node. Else, if $x_{\theta_i} \in B(\mu_j, R_j)$, it updates the upper bound, and if $d_f(x_{\theta_i}, q) < d_f(x_c, q)$ the node must be searched. If

[3]This requires to define the Lagrange dual function of (7), $\inf_x d_f(x,q) + \lambda(d_f(x,\mu) - R)$, $\lambda \ge 0$, set the gradient to zero, and use the change of variable $\theta = \frac{\lambda}{1+\lambda}$.

Algorithm 1 `CanPrune` by Cayton (2008)

1: **Input**: θ_l, θ_r, q, x_c, μ, R
2: Set $\theta = \frac{\theta_l+\theta_r}{2}$
3: Set $x_\theta = \nabla f^*(\theta\mu' + (1-\theta)q')$
4: Compute $d_f(x_\theta, q)$ and $d_f(x_\theta, \mu)$
5: **if** $\mathcal{L}(\theta) > d_f(x_c, q)$
Return `Yes`
6: **else if** $x_\theta \in B(\mu, R)$ and $d_f(x_\theta, q) < d_f(x_c, q)$
Return `No`
7: **else if** $x_\theta \notin B(\mu, R)$
Return `CanPrune`(θ, θ_r,q,x_c,μ,R)
8: **else if** $x_\theta \in B(\mu, R)$
Return `CanPrune`(θ_l, θ,q,x_c,μ,R)

neither bound holds, the algorithm continues the bisection. The procedure from (Cayton, 2008) is reported in Algorithm 1 for the reader's convenience. Note that in each iteration of `CanPrune`, two divergences are calculated in Step 4.

4. Variational Branch and Bound

In this section we present a novel and approximated bisection search based on variational inequalities. The proposed method requires only one divergence operation at each backtracked node (independent of the number of iterations), is efficient in high dimensions, and can be applied to latent variable models. The effect of the approximation is discussed in Section 4.4.

4.1. Overview and notation

In this section we assume that d_f is a regular Bregman divergence (i.e., KL divergence for an exponential family). Let $\mathcal{M}$, $\mathcal{Q}$, $\mathcal{X}_c$ be the distributions corresponding, respectively, to the node, the query, and the candidate nearest neighbor. Their mean parameters are, respectively, μ, q and x_c. Define a mixture distribution of components $\mathcal{M}$ and $\mathcal{Q}$ with weights θ and $1-\theta$:

$$\Theta = \theta\mathcal{M} + (1-\theta)\mathcal{Q}. \tag{13}$$

Our algorithm works as illustrated in Figure 1(c). Instead of explicitly computing the left-sided centroid x_θ (Step 3 of `CanPrune`, Figure 1(b)) at each recursive iteration, it uses a mixture model Θ (equation (13)). This allows to approximate $d_f(x_\theta, \mu)$ and $d_f(x_\theta, q)$ with lower bounds to $D(\Theta||\mathcal{M})$ and $D(\Theta||\mathcal{Q})$. The bounds depend only on $d_f(\mu, q)$ and $d_f(q, \mu)$ (which are fixed), and on the mixing parameter θ. As a consequence, bisection search only requires evaluating of simple scalar functions (as opposed to divergences).

4.2. Variational lower bounds

Our algorithm builds on the variational approximation to the KL divergence between mixtures by Hershey and Olsen (2007),[4] and on the observation that the approximation holds as an *inequality* whenever one of the two terms is unimodal.

4.2.1. Variational approximation to the KL (Hershey & Olsen, 2007)

Let $\mathcal{A} = \{\pi_i, \mathcal{A}_i\}$ be a mixture with weights π_i and components $\mathcal{A}_i$. Similarly, let $\mathcal{B} = \{\omega_j, \mathcal{B}_j\}$ be a different mixture. Assume the mixture components $\mathcal{A}_i$ and $\mathcal{B}_j$ belong to some family for which we can compute the KL divergence. Consider the KL divergence between $\mathcal{A}$ and $\mathcal{B}$:

$$D(\mathcal{A}||\mathcal{B}) = \mathrm{E}_{\mathcal{A}}[\log \frac{\mathcal{A}}{\mathcal{B}}] = \mathrm{E}_{\mathcal{A}}[\log \mathcal{A}] - \mathrm{E}_{\mathcal{A}}[\log \mathcal{B}]. \tag{14}$$

Hershey and Olsen (2007) derive a lower bound to the expected log-likelihood terms on the right-hand side of (14), and in turn an *approximation* (as the difference of two lower bounds) to the KL:

$$D(\mathcal{A}||\mathcal{B}) \approx \sum_i \pi_i \log \frac{\sum_{i'} \pi_{i'} \exp\{-D(\mathcal{A}_i||\mathcal{A}_{i'})\}}{\sum_j \omega_j \exp\{-D(\mathcal{A}_i||\mathcal{B}_j)\}}.$$

4.2.2. Variational lower bound to KL

Assume that $\mathcal{A} = \Theta = \theta\mathcal{M} + (1-\theta)\mathcal{Q}$ is a mixture of two components, and that $\mathcal{B}$ consists of a single component, and consider their KL divergence

$$\begin{aligned} D(\Theta||\mathcal{B}) = {} & \mathrm{E}_\Theta[\log\Theta] - \mathrm{E}_\Theta[\log\mathcal{B}] \\ = {} & \underbrace{\theta\mathrm{E}_{\mathcal{M}}[\log\Theta] + (1-\theta)\mathrm{E}_{\mathcal{Q}}[\log\Theta]}_{\mathrm{E}_\Theta[\log\Theta]} \\ & - \underbrace{\theta\mathrm{E}_{\mathcal{M}}[\log\mathcal{B}] + (1-\theta)\mathrm{E}_{\mathcal{Q}}[\log\mathcal{B}]}_{\mathrm{E}_\Theta[\log\mathcal{B}]}. \end{aligned} \tag{15}$$

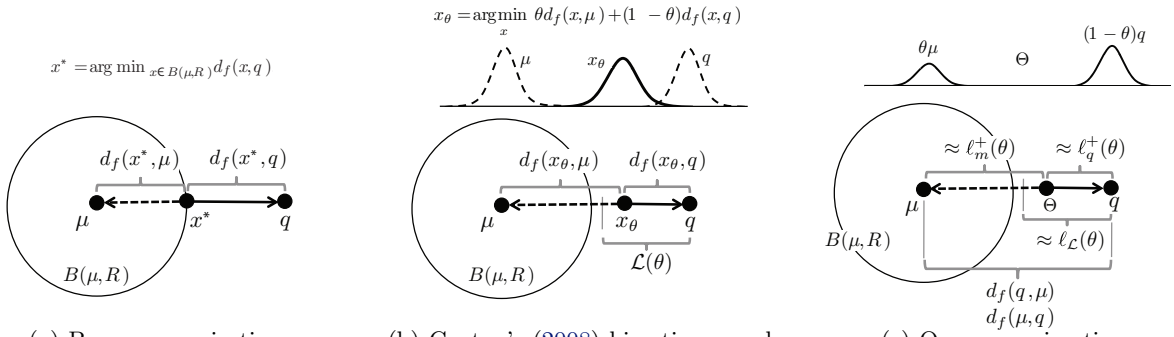
The first term cannot be computed exactly, but can be lower bounded (Hershey & Olsen, 2007). The second term can be computed exactly since it only involves expected log-likelihoods of individual components. Consequently, the variational approximation to $D(\Theta||\mathcal{B})$ holds as a *lower bound*. Dealing with lower bounds (as opposed to approximations) allows to characterize the errors made by Algorithm 2 (see Section 4.4).

Lower bounds to $D(\Theta||\mathcal{M})$ and $D(\Theta||\mathcal{Q})$ are easily derived (Coviello et al., 2013):

$$\begin{aligned} \ell_m = {} & \theta\log\left[\theta + (1-\theta)\exp\{-D(\mathcal{M}||\mathcal{Q})\}\right] \\ & + (1-\theta)\log\left[\theta + (1-\theta)\exp\{D(\mathcal{Q}||\mathcal{M})\}\right], \end{aligned} \tag{16}$$

$$\begin{aligned} \ell_q = {} & \theta\log\left[\theta\exp\{D(\mathcal{M}||\mathcal{Q})\} + (1-\theta)\right] \\ & + (1-\theta)\log\left[\theta\exp\{-D(\mathcal{Q}||\mathcal{M})\} + 1 - \theta\right]. \end{aligned} \tag{17}$$

[4] Cfr., Section 7 in (Hershey & Olsen, 2007)



(a) Bregman projection (b) Cayton's (2008) bisection search (c) Our approximation

Figure 1. a) Bregman projection of q onto the Bregman ball $B(\mu, R)$. The optimum x^* lies at the intersection of shell of $B(\mu, R)$ and the image under ∇f^* of the line segment between μ' and q'. There is no general analytical solution, and x^* can be found with bisection search. b) Algorithm 1 performs bisection search over θ, at each iteration computing a new centroid x_θ between μ and q (Step 3) and the two divergences from x_θ to the node μ and the query q, $d_f(x_\theta, \mu)$ and $d_f(x_\theta, q)$ (Step 4). Conditions to stop the bisection depend on these two divergences. c) Our method uses a mixture Θ with components μ and q and weights θ and $1-\theta$, instead of computing a new centroid x_θ. $d_f(x_\theta, \mu)$ and $d_f(x_\theta, q)$ are approximated with $\ell_m^+(\theta)$ and $\ell_q^+(\theta)$, which are functions only of the scalar θ, given the fixed quantities $d_f(\mu, q)$ and $d_f(q, \mu)$. Conditions to stop the bisection depend only on $\ell_m^+(\theta)$ and $\ell_q^+(\theta)$.

4.2.3. Monotonicity

When $D(\mathcal{Q}||\mathcal{M})$ is very close to zero, ℓ_m is not always monotonic (see (Coviello et al., 2013) for an illustration). This would make the bound useless for bisection search. Luckily, we can easily derive a monotonic bound $\ell_q^+(\theta) = \max(0, \ell_q(\theta))$, i.e., by setting ℓ_m to zero when it is not informative (i.e., when it is negative).

Lemma 4.1 *$\ell_m^+(\theta) = \max(0, \ell_m(\theta))$ is monotonic in $[0\,1]$.*

Proof $\ell_m(\theta)$ is convex in $[0\,1]$.[5] As a consequence of convexity, $\ell_m(\theta)$ can cross zero at most twice, and in particular at most twice in $[0\,1]$.
We have that $\ell_m(1) = 0$ and $\ell_m(0) = D(\mathcal{Q}||\mathcal{M}) \geq 0$. If $\ell_m(\theta)$ crosses zero only once (i.e., at $\theta = 1$), then $\ell_m(\theta)$ is monotonic for $\theta \in [0\,1]$. If $\ell_m(\theta)$ crosses zero also at $\theta^* \in (0\,1)$, we have that $\ell_m(\theta)$ is monotonic for $\theta \in [0\,\theta^*]$ (and until it reaches its minimum).[6] The result follows. ∎

Similarly, we can define the lower bound $\ell_q^+(\theta) = \max(0, \ell_q(\theta))$ (which is monotonic as well).

4.3. Approximated pruning algorithm

Our algorithm is based on the quantities:

$$\ell_q^+(\theta), \;\; \ell_m^+(\theta), \;\; \ell_{\mathcal{L}}(\theta) \equiv \ell_q^+(\theta) + \frac{\theta}{1-\theta}\left(\ell_m^+(\theta) - R\right). \quad (18)$$

The algorithm performs bisection search over θ, and attempts to locate the θ for which $l_m^+(\theta) = R$, using early stopping. As a lower bound we use $\ell_{\mathcal{L}}(\theta)$ instead of $\mathcal{L}(\theta)$. We set the upper bound to $\ell_q^+(\theta)$ whenever $\ell_m^+(\theta) < R$.[7]

At each iteration i, the algorithm computes the quantities in (18). If $\ell_{\mathcal{L}}(\theta_i) > d_f(x_c, q)$, it prunes the node. Else, if $\ell_m^+(\theta_i) < R$, it updates the upper bound, and if $\ell_q^+(\theta_i) < d_f(x_c, q)$ the node is searched. If neither bound holds, the algorithm continues the bisection. The procedure is summarized in Algorithm 2.

Algorithm 2 `CanPruneApprox`

1: **Input**: θ_l, θ_r; Const: $d_f(\mu, q)$, $d_f(q, \mu)$, R, $d_f(x_c, q)$
2: Set $\theta = \frac{\theta_l + \theta_r}{2}$
3: Compute $\ell_q^+(\theta)$, $\ell_m^+(\theta)$ and $\ell_{\mathcal{L}}(\theta)$
4: **if** $\ell_{\mathcal{L}}(\theta) > d_f(x_c, q)$
Return `Yes`
5: **else if** $\ell_m^+(\theta) < R$ and $\ell_q^+(\theta) < d_f(x_c, q)$
Return `No`
6: **else if** $\ell_m^+(\theta) > R$
Return `CanPruneApprox`(θ, θ_r)
7: **else if** $\ell_m^+(\theta) < R$
Return `CanPruneApprox`(θ_l, θ)

4.4. Discussion

The proposed algorithm executes faster on a node than Cayton's original algorithm. In particular, at *every* recursive iteration the latter requires updating the centroid x_θ plus two divergence operations (i.e., computing $d_f(x_\theta, q)$ and $d_f(x_\theta, \mu)$). On high dimen-

[5] This follows from the positivity of the second derivative $\frac{d^2 \ell_m(\theta)}{d\theta^2}$ in $[0\,1]$ see (Coviello et al., 2013).
[6] For $\theta \in [\theta^*\,1]$, $\ell_m(\theta)$ is negative, and is a trivial bound.
[7] We could update the upper bound for early stopping using an upper bound to $d_f(x_\theta, q)$ instead of ℓ_q^+. The reason we *do not* is to make the algorithm over-explorative under the circumstances explained in Section 4.4.

sional data, this results in a significant overhead, since individual divergence operations are expensive, and, most importantly, the procedure needs to be executed on a large fraction of the nodes (due to the large number of close neighbors in high dimensions (Moore, 2000)). Not surprisingly, Algorithm 1 did not work well for exact NN search on high dimensional data (Cayton, 2008), where it registered slow-downs (instead of speedups) relative to brute force search.

On the other hand, our method drastically reduces the amount of computation per backtracked node to a single divergence operation, independently of the number of iterations in the bisection search. At each recursive call of Algorithm 2, $\ell_m^+(\theta)$ and $\ell_q^+(\theta)$ are functions of the scalar θ and of the *fixed* quantities $d_f(q,\mu)$ and $d_f(\mu,q)$. Since $d_f(q,\mu)$ is already computed by the search routine when descending the tree (for choosing between a node's left and right child), only one additional divergence needs to be computed. We expect this to give our algorithm an edge for efficiency on high-dimensional data.

The solution proposed by Cayton (2008) to speed up retrieval time uses the bbtree for approximate search, by fixing a maximum budget of leaves that can be explored for each query, after which backtracking is stopped. This is suboptimal, since the approximation is independent of the query and may blindly ignore (once the budget is depleted) promising portions of the search space only because they appear later in the backtracking order. Our approximation, on the other hand, only depends on the parameters of the query and of the nodes of the tree (as opposed to a fixed leaf budget), and the resulting backtracking will adapt better to individual queries and the local structure of the tree, as illustrated below.

In fact, we can argue that Algorithm 2 tends to be over-explorative on nodes close to the query, and under-explorative on nodes further away. Since we are using in sequence an approximation and a lower bound,

$$d_f(x_\theta, q) \approx \quad D(\Theta||\mathcal{Q}) \geq \quad \ell_q^+(\theta), \tag{19}$$

$$d_f(x_\theta, \mu) \approx \quad D(\Theta||\mathcal{M}) \geq \quad \ell_m^+(\theta). \tag{20}$$

in *general* (19) (20) are not bounds to $d_f(x_\theta, q)$ and $d_f(x_\theta, \mu)$. However, as shown by the next claim, when the query and the node have very close distributions, (19) and (20) hold as lower bounds.

Claim 4.2 *If $q' = \mu' + \delta'$, for δ' small, we have $d_f(x_\theta, \mu) \geq l_m^+(\theta)$ (similarly, $d_f(x_\theta, q) \geq \ell_q^+(\theta)$).*

Proof We have $x'_\theta = \mu' + (1-\theta)\delta'$ and $\mu' = x'_\theta - (1-\theta)\delta'$. Consider the Riemannian manifold around x'_θ (with curvature $\nabla f^*(x'_\theta)$), and that the Riemannian metrics associated to f and f^* have identical infinitesimal length (Amari, 2009; Nielsen & Nock, 2009). Consequently we have:[8]

$$d_f(x_\theta, \mu) = \frac{1}{2}(1-\theta)^2 \delta'^t \nabla^2 f^*(x'_\theta)\delta' \tag{21}$$

$$= \frac{1}{2}(1-\theta)^2 \delta^t \nabla^2 f(x_\theta)\delta = (1-\theta)^2\Delta \tag{22}$$

where we use the notation $\Delta = \frac{1}{2}\delta^t \nabla^2 f(x_\theta)\delta$ to reduce clutter. Similarly, we have that $d_f(q,\mu) = \frac{1}{2}\Delta$ and $d_f(\mu,q) = \frac{1}{2}\Delta$. Using the approximations $\exp\{a\} = 1+a$ and $\log(1+a) = a$ (for $|a|$ small), we have that:

$$\ell_m(\theta) = (1-\theta)\log\left[1+(1-\theta)\Delta\right] + \theta\log\left[1-(1-\theta)\Delta\right] \tag{23}$$

$$\leq \log\left\{(1-\theta)\left[1+(1-\theta)\Delta\right] + \theta\left[1-(1-\theta)\Delta\right]\right\} \tag{24}$$

$$= \log\left\{1 + \left[(1-\theta)^2 - \theta(1-\theta)\right]\Delta\right\} \tag{25}$$

$$= \left[(1-\theta)^2 - \theta(1-\theta)\right]\Delta \leq (1-\theta)^2\Delta \tag{26}$$

where (24) follow from Jensen inequality and (26) form the fact that $\theta(1-\theta) \geq 0$ for $\theta \in [0\,1]$. Since $d_f(x_\theta, \mu) \geq 0$ we also have $d_f(x_\theta, \mu) \geq \ell_m^+(\theta)$. ∎

In this case, the decision whether to prune a node is based on a looser lower bound (i.e., $\ell_{\mathcal{L}}(\theta) \leq \mathcal{L}(\theta)$ instead of $\mathcal{L}(\theta)$ in Algorithm 1), and Algorithm 2 will consequently *prune less frequently*. Similarly, checking the condition $\ell_m^+(\theta) < R$ results in inflating the ball around the node, and may determine *more explorations*.

On the opposite, if q is far from μ, (19) and (20) may actually hold as upper bounds, making the backtracking *under-explorative*.[9] This has the effect that our algorithm better accounts for the *sparsity* of data relative to the dimensionality, reducing the backtracking for high-dimensional data. Skipping a *large* region of space centered far away form the query determines computational savings, but does not affect NN performance too much. Even if the region could *potentially* contain points very close to query, it most likely does not, since the points it contains are very few (relative to the dimensionality).

Our results in Section 5 demonstrate that, on moderate- and high-dimensional data, our algorithm is very efficient and returns almost perfect nearest neighbors. In addition, the approximation does not require

[8] To show (21) we can use Legendre duality $d_f(x_\theta, \mu) = f(x_\theta) + f^*(\mu') - \langle \mu', x_\theta \rangle$ and second order Taylor expansion of $f^*(\mu') = f^*(x'_\theta - (1-\theta)\delta')$ around x'_θ.

[9] Since x_θ is zero forcing (as a left-sided centroid), it will have smaller support than Θ, and consequently $d_f(x_\theta, \mu)$ (respectively, $d_f(x_\theta, q)$) will be smaller than $D(\Theta||\mathcal{M})$ (respectively, $D(\Theta||\mathcal{Q})$).

empirically tuning any meta-parameter (such as the leaf budget in (Cayton, 2008)).

Interestingly, since our algorithm requires only the computation of divergence terms (and bypasses the computation of the centroid), it is by no means limited to the exponential family. In fact, it can be readily applied to NN-search of any family of distributions,[10] as long as the divergence between individual members can be computed (efficiently). In Section 6 we illustrate this for time series models with latent variables.

Algorithm 1, on the other hand, cannot be adapted to latent variable models in an efficient way. In particular, the left-sided centroid x_θ cannot be computed analytically, since the hidden-state bases corresponding to the query and the node (i.e., $\mathcal{Q}$ and $\mathcal{M}$) may be mismatched.[11,12]

5. Experiments on histogram data

We first consider experiments on histogram data, where we can directly compare our proposed method against Cayton's (2008) exact and approximate NN search.[13] We consider the 9 histogram datasets from (Cayton, 2008) listed in Table 1, most of which are fairly high dimensional. We refer to our algorithm as `Variational`, Cayton's exact and approximated search as `Cayton` and `CaytonApprox`, respectively, and brute force search as `Brute`.

Performance is measured in terms of speedup relative to brute force search, and by the *average* number of elements in the database that are closer to the query than the returned one (NC) (Cayton, 2008). For exact algorithms (e.g., `Cayton` and `Brute`), NC = 0 always. All results are averages over queries not in the database.

In Table 1 we compare performance of `Variational` to `Cayton`. In general, on moderate to high dimensional data ($d \geq 32$), our algorithm provides larger speedups than `Cayton`. The gain in computational efficiency has a very modest effect on the quality of `Variational`, as demonstrated by the low NC values. In addition, whereas `Cayton` is *slower* than brute force search on the two datasets of highest dimensionality ($d = 371$ and $d = 1111$), `Variational` always registers a speedup. In particular, on "Sift signatures", `Variational` provides a substantial $6\times$ speedup over brute force search, with an NC of 0.0784. Further examining the NC values, `Variational` finds the NN 95.7% of the time, the 2nd NN 2.6%, and 2+ NN 1.7% (with a low average approximation ratio of $\epsilon = 0.3$).[14]

Table 1. Results on histogram data, using different datasets. The speedup is w.r.t. brute force search. The databases for rcv-n, Corel hist, Semantic space, and SIFT signature have size 500,000, 60,000, 4,500 and 10,000, respectively. The number of queries is 10,114, 6,616, 500 and 2,360, respectively

		`Cayton`	`Variational`	
Data set	dim	speedup	speedup	NC
rcv-8	8	60.75	21.23	0.0001
rcv-16	16	30.49	19.27	0.0000
rcv-32	32	17.18	24.60	0.0016
rcv-64	64	8.19	24.21	0.0003
Corel hist	64	2.56	4.14	0.0020
rcv-128	128	4.29	14.51	0.0018
rcv-256	256	2.58	2.95	0.0000
Semantic space	371	0.93	1.29	0.0020
SIFT signatures	1111	0.86	5.88	0.0784

In Figure 2 we compare `Variational` to `CaytonApprox`, on three datasets of medium to high dimensionality. For both methods, we set a fixed budget β_d of divergence computations allowed per query, after which backtracking stops.[15] In general, `Variational` is superior to `CaytonApprox`, achieving more effective speedups and lower NC values—note that in Figure 2(c) the left-most point for `CaytonApprox` (dashed red line) corresponds to no speedup, and is hence of no practical interest. In particular, the threshold-free version of `Variational` (from Table 1 and circled in Figure 2) does not require setting any meta-parameter and is efficient also on high dimensional data. These results show that, by adapting backtracking to the query and the local structure of the tree, our approximation effectively explores promising regions of the search space and achieves computational savings by under-exploring less promising ones.

[10] For distributions not in the exponential family, Algorithm 2 (and in particular the quantities in (18)), need to be expressed in terms of KL divergences between the distributions $\mathcal{Q}$, $\mathcal{M}$, $\mathcal{X}_c$, since they do not correspond to a regular Bregman divergence.

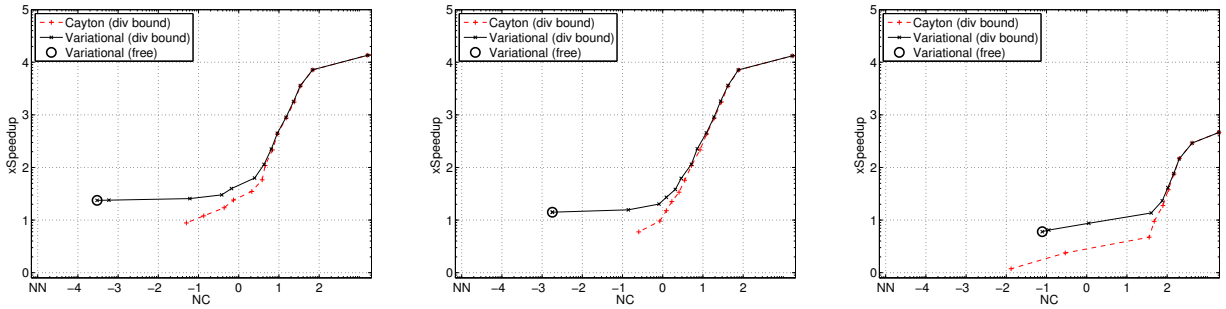
[11] For example, for hidden Markov models, the labeling of hidden states may be swapped.

[12] Note that naïvely approximating the centroid with iterative algorithms or numerical techniques would be inefficient.

[13] For Cayton's (2008) NN search we use the code available at `http://lcayton.com`. Our method is implemented in the same framework and is available on the authors' web pages.

[14] The approximation ratio is the smallest ϵ satisfying $d_f(x,q) \leq (1+\epsilon)d_f(x_q,q)$, where x_q is q's true NN and x is the returned element.

[15] Backtracking stops either when the budget is consumed or when one of the early stopping conditions is met.



(a) rcv-64 ($d = 64$) (b) rcv-128 ($d = 128$) (c) SIFT hist ($d = 1111$)

Figure 2. Log-log plots (base 10) for approximated NN-search using `Variational` and `CaytonApprox`, $\beta_d \in [2^0\, 2^{\log_2 |X|}]$. The vertical axis is the exponent of the speedup over `Brute`; the horizontal axis is the exponent of the average number of DB points closer to the query than the returned result (NC). Points toward the top-left corner are better.

6. Experiments on dynamic textures

In this section we test our algorithm on NN search of dynamic textures (DTs) (Doretto et al., 2003), which have been used to model video (Coviello et al., 2012; Mumtaz et al., 2012) and music (Barrington et al., 2010; Coviello et al., 2011).

A DT model represents a n-dimensional time series of length T, $y_1, \ldots, y_T$, as generated by a *linear dynamical system* (LDS):

$$\begin{aligned} s_t &= As_{t-1} + v_t \ , \\ y_t &= Cs_t + w_t + \bar{y} \ . \end{aligned} \tag{27}$$

where $S_t \in \mathbb{R}^m$ is a lower-dimensional hidden state process ($m < n$) that governs the dynamics, A is a $m \times m$ transition matrix, C a $n \times m$ bases matrix, and V_t and W_t are Gaussian noise processes. The KL divergence between two DTs can be computed efficiently with a recursion (Chan & Vasconcelos, 2005), so we can readily use our branch and bound search.

We consider the 4 datasets used by Coviello et al. (2012), three of video and one of music. A dataset is first divided into training/test splits, and a database of DTs X is compiled from the training set, with each DT modeling a portion of a video (or song). The database (of DTs) is then hierarchically organized in a tree structure using the Bag-of-Systems (BoS) Tree from (Coviello et al., 2012), which produces balls around each DT-node $\mathcal{M}$, compact in terms of $D(\cdot\,||\mathcal{M})$. Finally, we model portions of test videos (or songs) as DTs, and for each we find its NN in X using the BoS Tree in tandem with our backtracking algorithm.

Videos are preprocessed into a dense sampling of spatio-temporal cubes of pixels (as in (Coviello et al., 2012)). Each training video is then modeled as a dynamic texture mixture (DTM) with 4 components, and the databases X are formed by selecting all the video-level DT components. For each test video, we compute a query-DT from each cube, using (Doretto et al., 2003). Songs are first represented as a dense sampling of sequences of low-level features (as in (Coviello et al., 2012)). Each training song is modeled as a DTM with 4 components, and all the song-level DTs form the database X. For each experiment we compute average speedup over `Brute` and NC, using the same cross validation splits as (Coviello et al., 2012). Results on the video and music datasets are in Table 2. Note that in average `Variational` returns good answers. For example, on the CAL500 music data, where we registered the highest NC value, `Variational` still returns in average answers in the top 0.35%.

Table 2. Results for NN-search of DTs. Speedup is w.r.t. `Brute`. DIM are the degrees of freedom of the DT models.

DATA SET	SOURCE	DIM	$\lvert X\rvert$	SPEEDUP	NC
UCLA-39	VIDEO	211	624	2.11	0.13
DYTEX	VIDEO	355	630	6.16	1.24
KTH	VIDEO	760	3040	1.74	1.42
CAL500	MUSIC	385	1600	2.60	4.52

7. Conclusion

We have presented a branch and bound algorithm based on variational approximations. We have shown nearly perfect and efficient NN-search on histogram data of challenging dimensionality, as well as applicability to more complex generative time series models.

Acknowledgements

The authors thank Lawrence Cayton for providing code and data from (Cayton, 2008), and Malcolm Slaney for helpful discussion. E.C., A.B.C. and G.R.G.L. acknowledge support from Google, Inc. E.C. and G.R.G.L. acknowledge support from Yahoo!, Inc., the Sloan Foundation, KETI under the PHTM program, and NSF Grants CCF-0830535 and IIS-1054960. A.M. and A.B.C. were supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 110610).

References

Abdullah, A., Moeller, J., and Venkatasubramanian, S. Approximate Bregman near neighbors in sublinear time: Beyond the triangle inequality. *SCG*, 2012.

Amari, S. Information geometry and its applications: convex function and dually flat manifold. *Emerging Trends in Visual Computing*, 2009.

Banerjee, A., Merugu, S., Dhillon, I.S., and Ghosh, J. Clustering with Bregman divergences. *JMLR*, 2005.

Barrington, L., Chan, A.B., and Lanckriet, G. Modeling music as a dynamic texture. *IEEE TASLP*, 2010.

Blei, D.M. and Lafferty, J.D. A correlated topic model of science. *The Annals of Applied Statistics*, 2007.

Cayton, L. Fast nearest neighbor retrieval for bregman divergences. In *ICML*, 2008.

Cayton, L. Efficient Bregman range search. *NIPS*, 2009.

Chan, Antoni B. and Vasconcelos, Nuno. Probabilistic kernels for the classification of auto-regressive visual processes. In *IEEE CVRP*, 2005.

Coviello, E., Chan, A.B., and Lanckriet, G.R.G. Time series models for semantic music annotation. *IEEE TASLP*, 2011.

Coviello, E., Mumtaz, A., Chan, A.B., and Lanckriet, G.R.G. Growing a Bag of Systems Tree for fast and accurate classification. In *IEEE CVPR*, 2012.

Coviello, E., Mumtaz, A., Chan, A.B., and Lanckriet, G.R.G. Supplement to "That was fast! Speeding up NN search of high dimensional distributions". 2013.

Doretto, G., Chiuso, A., Wu, Y. N., and Soatto, S. Dynamic textures. *Intl. J. Computer Vision*, 2003.

Friedman, J.H., Bentley, J.L., and Finkel, R.A. An algorithm for finding best matches in logarithmic expected time. *ACM TOMS*, 1977.

Hershey, J.R. and Olsen, P.A. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE ICASSP*, 2007.

Moore, A.W. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *UAI*, 2000.

Mu, Y. and Yan, S. Non-metric locality-sensitive hashing. In *AAAI CAI*, 2010.

Mumtaz, A., Coviello, E., Lanckriet, G., and Chan, A. Clustering Dynamic Textures with the Hierarchical EM Algorithm for Modeling Video. *IEEE TPAMI*, 2012.

Nielsen, F. and Nock, R. Sided and symmetrized Bregman centroids. *IEEE Transactions on IT*, 2009.

Nielsen, F., Piro, P., and Barlaud, M. Bregman vantage point trees for efficient nearest neighbor queries. In *IEEE ICME*, 2009a.

Nielsen, F., Piro, P., Barlaud, M., et al. Tailored Bregman ball trees for effective nearest neighbors. In *EuroCG*, 2009b.

Omohundro, S.M. *Five balltree construction algorithms*. International Computer Science Institute, 1989.

Pereira, F., Tishby, N., and Lee, L. Distributional clustering of English words. In *Association for Computational Linguistics*, 1993.

Puzicha, J., Buhmann, J.M., Rubner, Y., and Tomasi, C. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE ICCV*, 1999.

Rasiwasia, N., Moreno, P.J., and Vasconcelos, N. Bridging the gap: Query by semantic example. *IEEE Transactions on Multimedia*, 2007.

Rockafellar, R.T. *Convex analysis*. Princeton university press, 1996.

Slaney, M. and Casey, M. Locality-sensitive hashing for finding nearest neighbors. *IEEE, SPM*, 2008.

Uhlmann, J.K. Satisfying general proximity/similarity queries with metric trees. *IP letters*, 1991.

Wainwright, M.J. and Jordan, M.I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

Zhang, Z., Ooi, B.C., Parthasarathy, S., and Tung, A.K.H. Similarity search on Bregman divergence: towards non-metric indexing. *VLDB E*, 2009.