Fast Spectral Learning using Lanczos Eigenspace Projections

Sridhar Mahadevan *
Department of Computer Science
University of Massachusetts
140 Governor's Drive
Amherst, MA 01003
(mahadeva)@cs.umass.edu

Abstract

The core computational step in spectral learning – finding the projection of a function onto the eigenspace of a symmetric operator, such as a graph Laplacian – generally incurs a cubic computational complexity $\mathcal{O}(\mathbb{N}^3)$. This paper describes the use of Lanczos eigenspace projections for accelerating spectral projections, which reduces the complexity to $\mathcal{O}(nT^{\text{op}} + n^2\mathbb{N})$ operations, where n is the number of distinct eigenvalues, and T^{op} is the complexity of multiplying T by a vector. This approach is based on diagonalizing the restriction of the operator to the *Krylov space* spanned by the operator and a projected function. Even further savings can be accrued by constructing an approximate Lanczos tridiagonal representation of the Krylov-space restricted operator. A key novelty of this paper is the use of Krylov-subspace modulated Lanczos acceleration for *multi-resolution wavelet analysis*. A challenging problem of learning to control a robot arm is used to test the proposed approach.

Introduction

At its core, spectral learning involve a possibly costly eigenvector computation: for example, fully diagonalizing the operator T of size $\mathbb{N} \times \mathbb{N}$ incurs a $\mathcal{O}(\mathbb{N}^3)$ computation. In many applications, such a direct approach may be infeasible. For example, in clustering and semi-supervised learning, $\mathbb{N}$ is the number of training examples, and large datasets may contain many hundreds of thousands of instances. In computer graphics, $\mathbb{N}$ is the number of vertices in an object's $3D$ representation, and $3D$ models can similarly have millions of vertices. In Markov decision processes, $\mathbb{N}$ is the number of sampled states, and many MDPs generate huge state spaces.

Rather than directly project on the eigenspaces of a symmetric operator, this paper uses a highly compact tridiagonal representation of the operator T, constructed by restricting it to the *Krylov space* spanned by powers of T and a function f whose eigenspace projections are desired. A key technical result shows that the projections of a function f on the eigenspaces of a symmetric operator T form a basis for the Krylov space spanned by T and f (e.g. see (Malsen, Orrison, and Rockmore 2003)). Furthermore, an efficient *Lanczos* type algorithm can be developed that uses the compact tridiagonal representation of the operator T.

Closest in spirit to this paper is the use of Lanczos-related Krylov subspace techniques described in (Freitas et al. 2006). However, a principal novelty of this paper is the use of Krylov preconditioning to accelerate multiscale *wavelet* methods on graphs (Coifman and Maggioni 2006). In contrast to standard spectral methods, which project onto 1-dimensional eigenspaces, wavelet methods combine time (or space) and frequency into *time-frequency* or *space-frequency* atoms. Projections are carried out, not by diagonalization, but by *dilation* of the operator onto a band of frequencies. Wavelet methods rely on compact multiscale basis functions, instead of one-dimensional global eigenvectors. This approach results in an intrinsically multi-resolution analysis, decomposing the underlying vector space of functions $\mathbb{C}^{\mathbb{N}}$ into a hierarchy of orthogonal spaces, where the coarse spaces are spanned by *scaling functions* and the orthogonal fine spaces are spanned by *wavelet* bases. The paper compares the standard diffusion wavelet method with the Krylov-subspace accelerated variant on a challenging problem of learning to control a robot arm (the Acrobot task).

Mathematical Preliminaries

Let V be a finite-dimensional vector space over some field, such as the $\mathbb{N}$-dimensional real Euclidean space $\mathbb{R}^{\mathbb{N}}$ or the $\mathbb{N}$-dimensional space over complex numbers $\mathbb{C}^{\mathbb{N}}$. The space of $\mathbb{N} \times \mathbb{N}$ matrices over $\mathbb{R}$ ($\mathbb{C}$) is denoted as $M_{\mathbb{N}}(\mathbb{R})$ ($M_{\mathbb{N}}(\mathbb{C})$). The matrix representations of a symmetric operator are denoted by T, so that $T \in M_{\mathbb{N}}(\mathbb{R})$ or $T \in M_{\mathbb{N}}(\mathbb{C})$. T is assumed to be completely diagonalizable, with $\mathbb{N}$ eigenvalues that are either real-valued or complex-valued. Eigenvalues may of course have geometric multiplicity > 1. Let n denote the number of distinct eigenvalues of T, with corresponding eigenspaces denoted by V_i. Any vector $\in \mathbb{C}^{\mathbb{N}}$ can be decomposed into a direct sum of components, each of which lies in a distinct eigenspace of T:

$$\mathbb{C}^{\mathbb{N}} = V_1 \oplus V_2 \ldots \oplus V_n$$

*This research is funded in part by a grant from the National Science Foundation IIS-0435999.

where $\oplus$ denotes the direct sum of eigenspaces. The projection of any function $f \in \mathbb{C}^{\mathbb{N}}$ can be written as:

$$f = f_1 + f_2 + \ldots + f_n$$

where each individual eigenspace projection $f_i \in V_i$.

Krylov Spaces

The j^{th} *Krylov* subspace $\mathcal{K}_j$ generated by a symmetric operator T and a function f is written as:

$$\mathcal{K}_j = \langle f, Tf, T^2 f, \ldots, T^{j-1} f \rangle$$

where $\mathcal{K}_j \subset \mathbb{C}^{\mathbb{N}}$. Note that $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \ldots$, such that for some $m, \mathcal{K}_m = \mathcal{K}_{m+1} = \mathcal{K}$. Thus, $\mathcal{K}$ is the T-invariant Krylov space generated by T and f. The projections of f onto the eigenspaces of T form a basis for the Krylov space $\mathcal{K}$ (Malsen, Orrison, and Rockmore 2003).

Theorem 1 *If $T \in \mathbb{M}_{\mathbb{N}}(\mathbb{C})$ is diagonalizable, and has n distinct eigenvalues, the non-trivial projections of f onto the eigenspaces of T form a basis for the Krylov space generated by T and f.*

The coefficients in the eigenspace expansions of $T^k f$ form a *Vandermonde* matrix, which can be shown to be invertible. Thus, each $f_i \in \mathcal{K}$. However, it also follows that each $T^k f$ can be written as a linear combination of the f_i. Thus, $\mathcal{K}$ is also spanned by the f_i.

Theorem 2 *The dimension of the Krylov space $\mathcal{K}$ generated by T and f is equal to the number of distinct eigenspace projections of f. The eigenvectors of the restriction of T to $\mathcal{K}$ are scalar multiples of f_i.*

Thus, eigenspace projections for T can be done much faster with respect to the Krylov subspace $\mathcal{K}$, since the dimension of the Krylov subspace n can be significantly less than N. To project f onto an eigenvector u of T requires computing the inner products $f_u = \frac{\langle f,u \rangle}{\langle u,u \rangle} u$, which would take $3\mathbb{N}$ steps relative to the default unit vector basis of $\mathbb{C}^{\mathbb{N}}$, but incur only a cost of $3n$ with respect to the Krylov basis $\mathcal{K}$.

Lanczos Eigenspace Projection Algorithm

A key algorithm is now presented, which represents the restriction of a real symmetric operator T with respect to the Krylov subspace $\mathcal{K}$ as a series of tridiagonal *Lanczos* matrices L_j:

$$L_j = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \vdots & \\ & \ldots & \ldots & \beta_{j-1} \\ & & \beta_{j-1} & \alpha_j \end{pmatrix}$$

The coefficients are computed using the iterative Lanczos method shown in Figure 1, a variant of the classical Gram-Schmidt orthogonalization method.

The main use of the Lanczos matrices L_j is summarized in the following result.

```
Lanczos Iteration:$(n, \epsilon, T)$
// $T$: Symmetric real operator (e.g. graph Laplacian $\mathcal{L}$)
// $f$: Function whose eigenspace projections are desired
// $q_1, \ldots, q_m$: Orthonormal basis for $\mathcal{K}$
// $L_m$: Lanczos representation of $T$ w.r.t Krylov basis
// $\delta$ : Tolerance parameter
// $n$: Number of desired eigenspace projections

1. Initialization: $\beta_0 = 0, q_0 = 0, q_1 = \frac{f}{\|f\|}$.
2. for $i = 1, 2, \ldots$
    • $v = Tq_i$
    • $\alpha_i = \langle q_i, v \rangle$
    • $v = v - \beta_{i-1} q_{i-1} - \alpha_i q_i$
    • for $j = 1$ to $i$
      – $\gamma = \langle q_{i-j+1}, v \rangle$
      – $v = v - \gamma q_{i-j+1}$
    • $\beta_i = \|v\|$
    • If $\beta_i > \delta$
      – $q_{i+1} = \frac{v}{\beta_i}$
    • else $q_{i+1} = 0$.
```

Figure 1: Pseudo-code for Lanczos iteration.

Theorem 3 *If the dimension of the Krylov space $\mathcal{K} = \langle f, Tf, T^2 f, \ldots \rangle$ is m, then $\{q_1, \ldots, q_m\}$ is an orthonormal basis for $\mathcal{K}$, and L_m is the restriction of T to the subspace $\mathcal{K}$ with respect to this basis.*

The computational complexity of running the Lanczos iteration specified in Figure 1 is summarized in the following result.

Theorem 4 *If T is a real symmetric $\mathbb{N} \times \mathbb{N}$ operator, and f is any function (vector) $\in \mathbb{C}^{\mathbb{N}}$, then the number of operations required to carry out n iterations of the Lanczos algorithm is given by $\mathcal{O}(nT^{\mathrm{op}} + n^2\mathbb{N})$.*

Here T^{op} denotes the number of steps required to apply T to any function f. In most of the applications of spectral methods, T is typically highly sparse, and consequently T^{op} is never larger than the number of non-zero entries in the matrix representation (i.e. usually linear in $\mathbb{N}$).

Lanczos Eigenspace Projection

The overall procedure for computing eigenspace projections using the Lanczos approach is summarized in Figure 2. Let function $f \in \mathbb{C}^{\mathbb{N}}$ whose $m \leq n$ eigenspace projections with respect to an operator T are desired. Let L_m be the Lanczos tridiagonal matrix representation of the restriction of T to the Krylov subspace $\mathcal{K}$. Let $\{q_1, \ldots, q_m\}$ be the orthonormal basis for the Krylov space $\mathcal{K}$.

The main computational savings resulting from the use of the Lanczos eigenspace projection method described in Figure 2 is that the matrix diagonalized may be significantly smaller: L_m is an $m \times m$ tridiagonal matrix, rather than the original $\mathbb{N} \times \mathbb{N}$ matrix.

Theorem 5 *If T is an $\mathbb{N} \times \mathbb{N}$ matrix with n distinct eigenvalues, and f is a non-zero vector $\in \mathbb{C}^{\mathbb{N}}$, then the projection*

```
Lanczos Eigenspace Projection:$(m, f, T)$
// $m$: Number of desired eigenspace projections
// $T$: symmetric real operator
// $f$: Function whose eigenspace projections are desired
// $Q_m$: $\mathbb{N} \times m$ orthogonal matrix of Krylov basis vectors $q_i$.

1. Compute $L_m$ and $Q_m$ using the algorithm described in Figure 1, terminating when a 0 vector is produced.
2. Diagonalize $L_m$, and denote the eigenvalues as $\mu_1, \ldots, \mu_m$, and corresponding eigenvectors as $\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_m$.
3. Compute the eigenspace projections $\tilde{f}_i = \langle f, \tilde{u}_i \rangle \tilde{u}_i$ with respect to the basis $q_i$ of $\mathcal{K}$.
4. Compute the eigenspace projections $f_i = Q_m \tilde{f}_i$ of $f$ with respect to the original unit vector basis.
```

Figure 2: Algorithm for Lanczos eigenspace projection.

of f onto the eigenspaces of T requires $\mathcal{O}(nT^{op} + n^2\mathbb{N})$ operations

Crucially, note that the number of desired projections $m \leq n$, the maximum number of distinct eigenvalues. Consequently, the core step of computing eigenvectors scales at most cubically in m, rather than the much larger $\mathbb{N}$. In many of the intended applications of spectral learning, this difference can result in significant savings in computational time, as will be illustrated in some experiments below.

Multiscale Wavelet Analysis

A principal novelty of this paper is the use of Krylov methods to accelerate multiscale wavelet analysis on graphs, in particular using the framework of *diffusion wavelets* (Coifman and Maggioni 2006). The wavelet approach replaces diagonalization by dilation, uses compact basis functions rather than "flat" global eigenvectors, yielding a multi-resolution analysis. Diffusion wavelets generalize wavelet analysis to functions on manifolds and graphs. The input to the algorithm is a "precision" parameter $\varepsilon > 0$, and a weighted graph (G, E, W). The construction is based on using the natural random walk $P = D^{-1}W$ on a graph and its powers to "dilate", or "diffuse" functions on the graph, and then defining an associated coarse-graining of the graph. Symmetrizing P by conjugation, and taking powers gives:

$$H^t = D^{\frac{1}{2}} P^t D^{-\frac{1}{2}} = \sum_{i \geq 0} (1 - \lambda_i)^t \xi_i(\cdot) \xi_i(\cdot) \qquad (1)$$

where $\{\lambda_i\}$ and $\{\xi_i\}$ are the eigenvalues and eigenfunctions of the Laplacian as above. Hence the eigenfunctions of H^t are again ξ_i and the i^{th} eigenvalue is $(1 - \lambda_i)^t$. We assume that H^1 is a sparse matrix, and that the spectrum of H^1 has rapid decay.

A diffusion wavelet tree consist of orthogonal diffusion scaling functions Φ_j that are smooth bump functions, with some oscillations, at scale roughly 2^j (measured with respect to geodesic distance, for small j), and orthogonal wavelets Ψ_j that are smooth localized oscillatory functions at the same scale. The scaling functions Φ_j span a subspace V_j, with the property that $V_{j+1} \subseteq V_j$, and the span of

```
DiffusionWaveletTree $(H_0, \Phi_0, J, \varepsilon)$:

// $H_0$: symmetric operator represented on the orthogonal basis $\Phi_0$
// $\Phi_0$ : initial (e.g. unit vector) basis
// $J$ : number of levels of the tree
// $\varepsilon$: precision

for j from 0 to J do,
    1. Orthogonalize operator at level $j$: $H_j \sim_\varepsilon Q_j R_j$, with $Q_j$ orthogonal.
    2. Compute scaling functions: $\Phi_{j+1} \leftarrow Q_j = H_j R_j^{-1}$
    3. Dilate the operator $[H_0^{2^j}]_{\Phi_{j+1}}^{\Phi_{j+1}} \sim_{j\varepsilon} H_{j+1} \leftarrow R_j R_j^*$.
    4. Compute sparse factorization $I - \Phi_{j+1}\Phi_{j+1}^* = Q_j' R_j'$, with $Q_j'$ orthogonal.
    5. Compute the wavelet bases $\Psi_{j+1} \leftarrow Q_j'$.
end
```

Figure 3: Pseudo-code for building a Diffusion Wavelet Tree.

Ψ_{j+1}, W_j, is the orthogonal complement of V_j into V_{j+1}. This is achieved by using the dyadic powers H^{2^j} as "dilations", to create smoother and wider (always in a geodesic sense) "bump" functions (which represent densities for the symmetrized random walk after 2^j steps), and orthogonalizing and downsampling appropriately to transform sets of "bumps" into orthonormal scaling functions. The algorithm is described in Figure 3.

Accelerating Diffusion Wavelets

The procedure for combining Krylov subspace methods with multiscale diffusion wavelets is outlined in Figure 4. The graph operator is first preconditioned using the Lanczos method shown in Figure 1, converting it into a tridiagonal Lanczos matrix. Then, the diffusion wavelet tree is constructed using the procedure described in Figure 3. Finally, the scaling functions and wavelets constructed using the tridiagonal Lanczos matrix are then transformed back to the original basis using the change of basis matrix Q_m.

Learning to Solve MDPs using Krylov-subspace accelerated Wavelets

Solving a Markov requires approximating a real-valued function $V^\pi : S \rightarrow \mathbb{R}$. *Representation policy iteration* (RPI) is a general framework to learning representation and control in MDPs (Mahadevan and Maggioni 2007), where basis functions for projecting the value function are constructed from spectral analysis of a graph. RPI approximates the true action-value function $Q^\pi(s, a)$ for a policy π using a set of basis functions $\phi(s, a)$ made up of the multiscale diffusion scaling and wavelet basis functions constructed as described above. Similar in spirit to the use of Krylov bases for Markov decison processes (Petrik 2007), the diffusion operator $T = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ is tridiagonalized with respect to the

```
Krylov-DiffusionWaveletTree ($\tilde{H}_0, Q_m$):

// $\tilde{H}_0$: symmetric tridiagonal Lanczos matrix, representing
$T$ using the Krylov basis

1. Run the diffusion wavelet procedure
   on the tridiagonalized Lanczos matrix:
   DiffusionWaveletTree($\tilde{H}_0, Q_m, J, \varepsilon$):
2. Remap the Krylov-subspace restricted scaling functions
   $\tilde{\Phi}_j$ at level $j$ onto the original basis: $\Phi_j \leftarrow Q_m \tilde{\Phi}_j$.
3. Remap the Krylov-subspace restricted wavelet bases at
   level $j$: $\tilde{\Psi}_j \leftarrow Q_m \Psi_j$.
end
```

Figure 4: Krylov-accelerated procedure for building a diffusion wavelet tree.

Krylov bases defined by R and T.

The Acrobot task (Sutton and Barto 1998) is a two-link under-actuated robot that is an idealized model of a gymnast swinging on a high bar. The only action available is a torque on the second joint, discretized to one of three values (positive, negative, and none). The reward is -1 for all transitions leading up to the goal state. The detailed equations of motion are given in (Sutton and Barto 1998). The state space for the Acrobot is 4-dimensional. Each state is a 4-tuple represented by $(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$. θ_1 and θ_2 represent the angle of the first and second links to the vertical, respectively, and are naturally in the range $(0, 2\pi)$. $\dot{\theta}_1$ and $\dot{\theta}_2$ represent the angular velocities of the two links. Notice that angles near 0 are actually very close to angles near 2π due to the rotational symmetry in the state space.

The time required to construct Krylov accelerated diffusion wavelets with regular diffusion wavelets is shown in Figure 5 (left plot). There is a very significant decrease in running time using the Krylov-subspace restricted Lanczos tridiagonal matrix. In this experiment, data was generated doing random walks in the Acrobot domain, from an initial sample size of 100 to a final sample size of 1000 states. The performance of regular diffusion wavelets with Krylov accelerated wavelets is shown on the right plot. This experiment was carried out using a modified form of RPI with on-policy resampling. Specifically, additional samples were collected during each new training episode from the current policy if it was the best-performing policy (in terms of the overall performance measure of the number of steps), otherwise a random policy was used. Note that the performance of Krylov accelerated diffusion wavelets is slightly better than regular diffusion wavelets, suggesting that there is no loss in performance.

Figure 6 shows the dependence of the time required to construct the wavelet tree for various values of δ. As δ decreases, the approximation quality increases and the size of the Lanczos matrix grows. This figure also shows the extra pre-processing time required in constructing the tridiagonal Lanczos matrix.

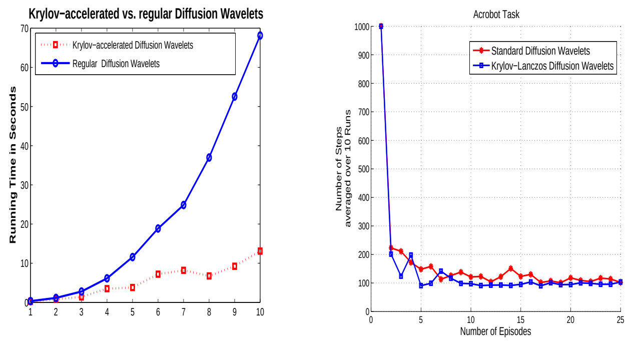

Figure 5: Left: time required to construct regular vs. Krylov accelerated diffusion wavelets on the Acrobot control task; right: the resulting performance with RPI on the Acrobot task.

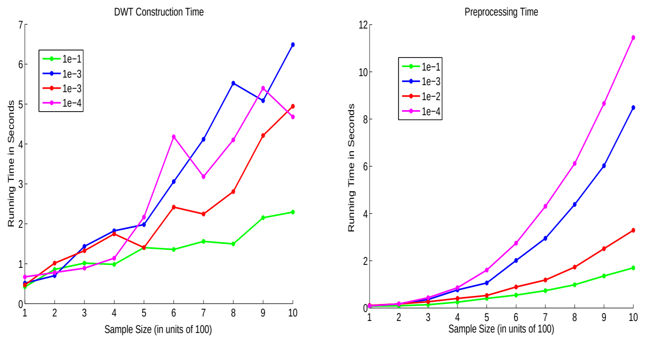

Figure 6: Left: time to construct diffusion wavelet tree; Right: Lanczos preprocessing time (both plots for various values of δ. Note the difference in scale from Figure 5.

References

Coifman, R., and Maggioni, M. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21(1):53–94.

Freitas, N.; Wang, Y.; Mahdaviani, M.; and Lang, D. 2006. Fast krylov methods for N-body learning. *Advances in Neural Information Processing Systems* 17.

Mahadevan, S., and Maggioni, M. 2006. Value function approximation with Diffusion Wavelets and Laplacian Eigenfunctions. In *Proceedings of the Neural Information Processing Systems (NIPS)*. MIT Press.

Mahadevan, S., and Maggioni, M. 2007. Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research* 8:2169–2231.

Malsen, D.; Orrison, M.; and Rockmore, D. 2003. Computing isotypic projections with the lanczos iteration. *SIAM* 2(60/61):601–28.

Petrik, M. 2007. An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2574–2579.

Sutton, R., and Barto, A. G. 1998. *An Introduction to Reinforcement Learning*. MIT Press.