Learning the Structure of Manifolds using Random Projections

Yoav Freund, Sanjoy Dasgupta, Mayank Kabra, and Nakul Verma
{yfreund, dasgupta, mkabra, nverma}@ucsd.edu
University of California, San Diego

Nature of collected data

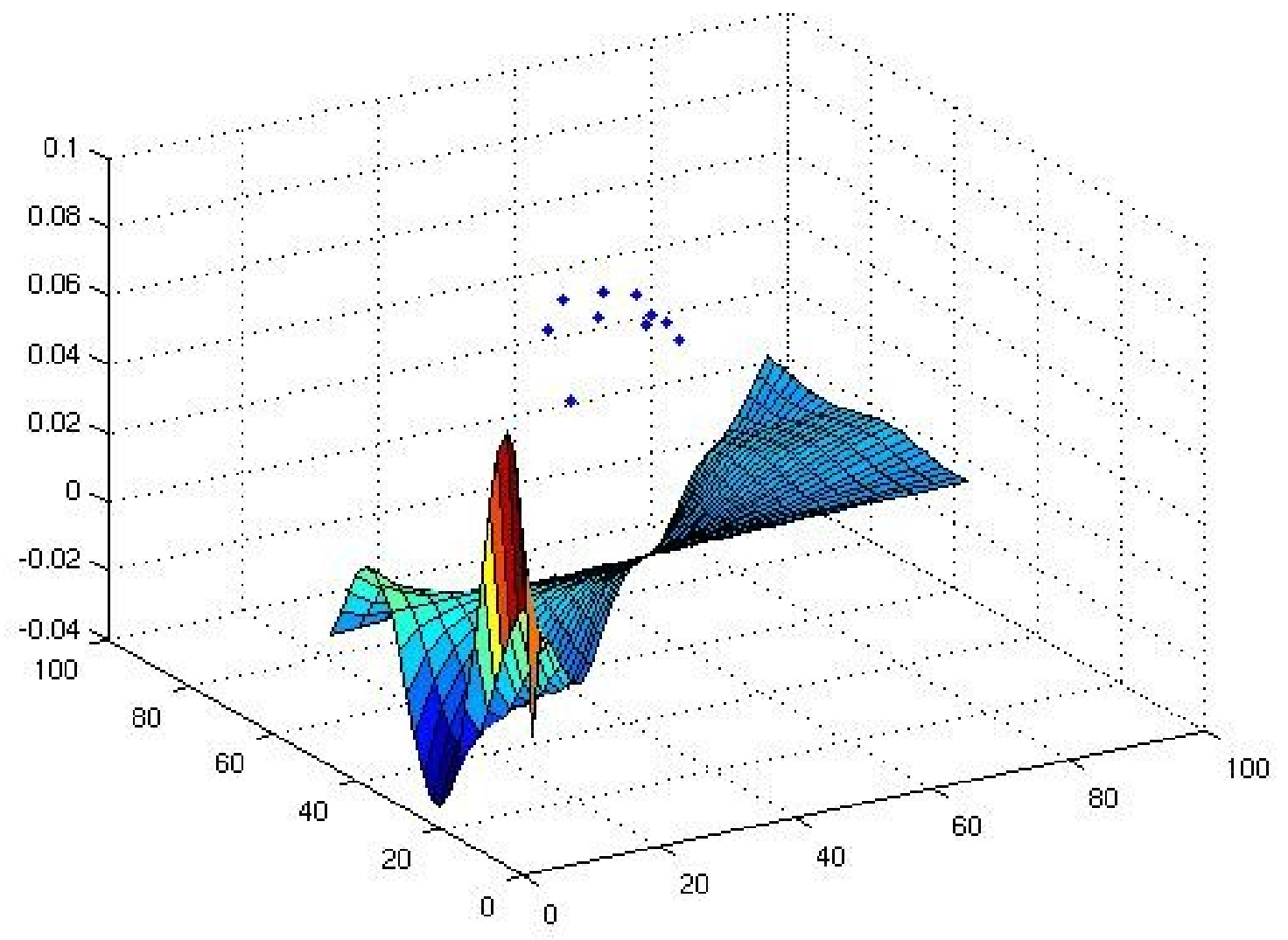
In many applications, the data we collect resides in a very low dimensional subspace (manifold) of the ambient space.

Examples

- Handwritten characters (there are only a few relevant parameters such as shape, tilt and curvature of the characters)
- Spoken Natural Language (only a certain set of phonemes follow other phonemes)
- News articles (frequencies of certain words increase if the document belongs to the politics class as opposed to sports)
- and much more!

Traditional learning algorithms, don't exploit this fact and suffer from computations done in high dimensions

Even though the surface (right) resides in $\Re^3$, we would like the learning algorithm only depend on the intrinsic dimension, which can be different in different parts of the space.



Theoretical basis

Notions of intrinsic dimension

Doubling dimension of $S \subset \Re^D$ is the smallest integer d such that for any ball $B(x,r)$, the set $B(x,r) \cap S$ can be covered by 2^d balls of radius $r/2$.

A set $S \subset \Re^D$ is said to have *local covariance dimension (d, ε)* if the largest d eigenvalues of its covariance matrix satisfy $\sigma_1^2 + \ldots + \sigma_d^2 \geq (1\text{-}\varepsilon)\,(\sigma_1^2 + \ldots + \sigma_D^2)$

Main Result

Theorem (Dasgupta & Freund '07) Pick any cell C in the RPTree, and suppose the data in C have intrinsic dimension (doubling dimension or local covariance dimension) d. Then with constant probability, any descendant cell $\geq O(d)$ levels below will have expected diameter at most half that of C.

Note that a d dimensional *Riemannian manifold* in $\Re^D$ has a doubling dimension of $O(d)$ (for bounded curvatures), so the theorem applies well to manifolds. A key benefit of using RPTree over traditional manifold learning techniques is that it doesn't construct an explicit nearest neighbor graph (which scales quadratically with number of points), instead it partitions the space by choosing random vectors (which scales linearly).

Proposed datastructure

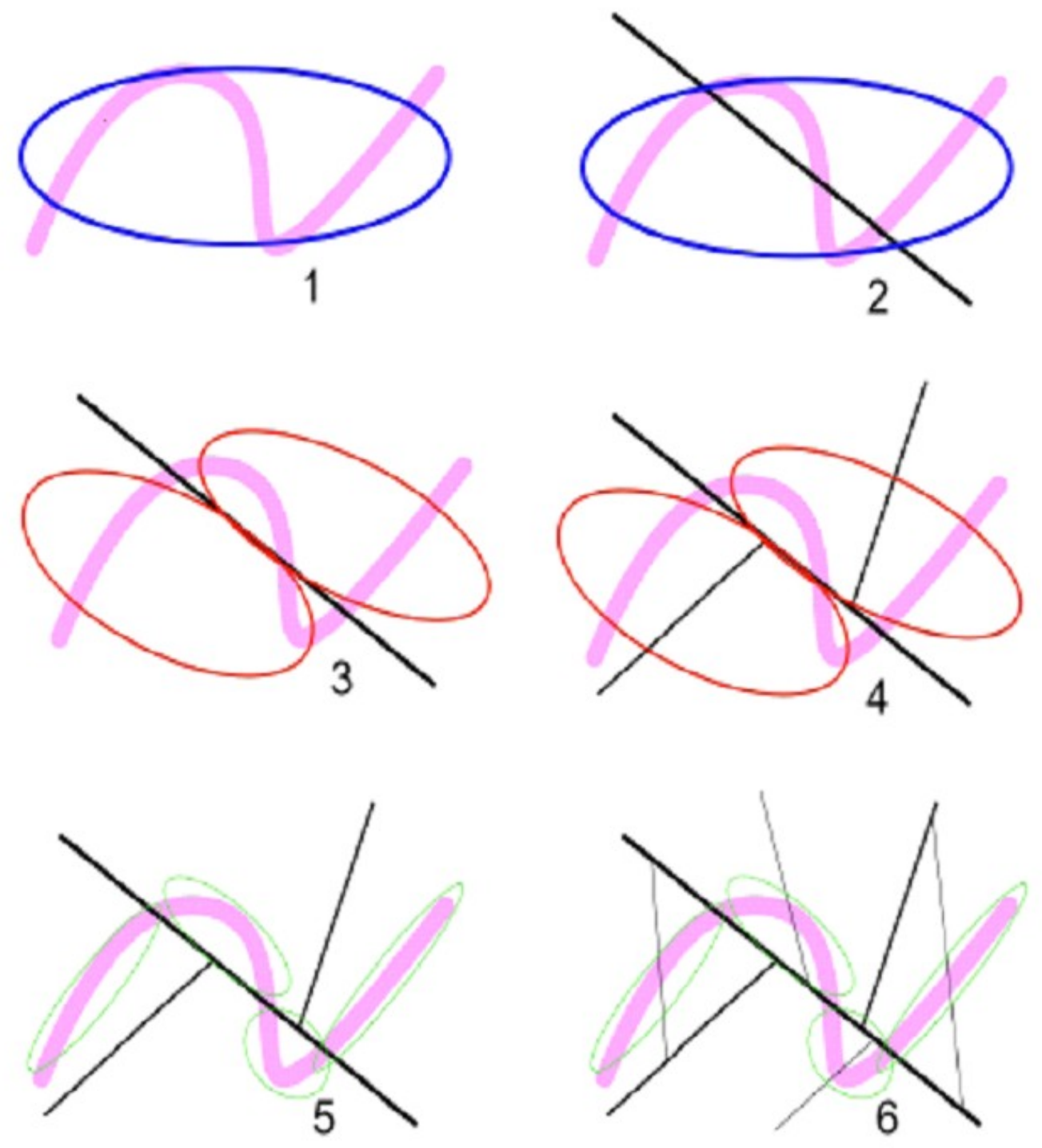
Random projection trees

```
procedure MakeRPTree (S)
  if |S| < MinSize then return(Leaf)
  else
    Rule ← ChooseRule (S)
    LTree ← MakeRPTree( {x∈ S: Rule(x)} )
    RTree ← MakeRPTree( {x∈ S: ¬ Rule(x)} )
    return ([Rule, LTree, RTree])

procedure ChooseRule (S)
  if $\Delta^2(S) \leq c.\Delta^2_{avg}(S)$
    $u$ ← random unit vector
    sort projected values $a_i \leftarrow s_i \cdot u$   [ $\forall\, s_i \in S$ ]
    $c_i = i\ \mathrm{var}(a_1, \ldots, a_i) + (n - i)\ \mathrm{var}(a_{i+1}, \ldots, a_n)$
    find $i$ that minimizes $c_i$ and set $\theta = (a_i + a_{i+1})/2$
    Rule(x) ← $x \cdot u \leq \theta$
  else
    Rule(x) ← $\| x - \mathrm{mean}(S) \| \leq \mathrm{median}\{ \| z - \mathrm{mean}(S) \| : z \in S \}$
  return (Rule)
```

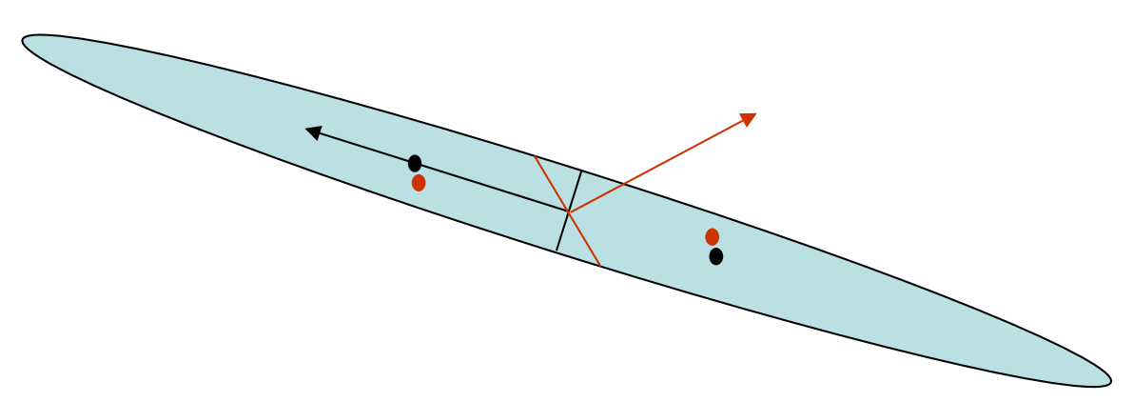
[where $\Delta^2(S)$ is the square diameter of S, $\Delta^2_{avg}(S)$ is the average square diameter of s]

Algorithm in action:



PCA vs. Random projection

Note that, locally, since most of the variance is concentrated in a few directions (in a manifold), the benefits of a PCA based split can be realized by simply picking a *random* direction.



Note that a split based on the red vector (random) is just as good as the black vector (PCA) in reducing the diameter of the set, since the fraction of points falling to the other side (two triangular regions) is very small.

RPTrees for vector quantization and nearest neighbors

Vector Quantization: A *quantization* technique in signal processing, which allows modeling of a probability density by a few vectors. VQ is a widely applicable technique for lossy data compression.

Since the diameter of a cell in the RPTree decreases quickly with height, the mean vector of a leaf cell can be regarded as a good *quantizer* for the data belonging to that cell.

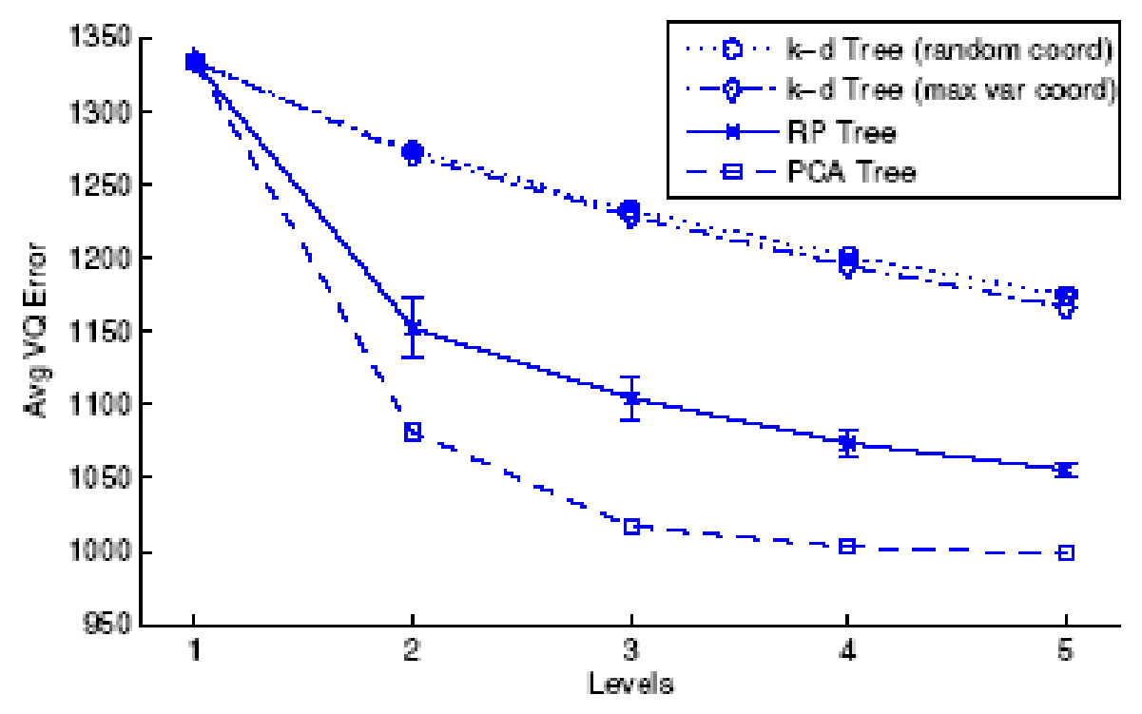
Near Neighbor queries: since Bayes Risk of nearest neighbor applies well to majority vote in a small enough cells, RPTrees can quickly find a near neighbor of a query point without significant amount of error.

Experiments

Synthetic datasets

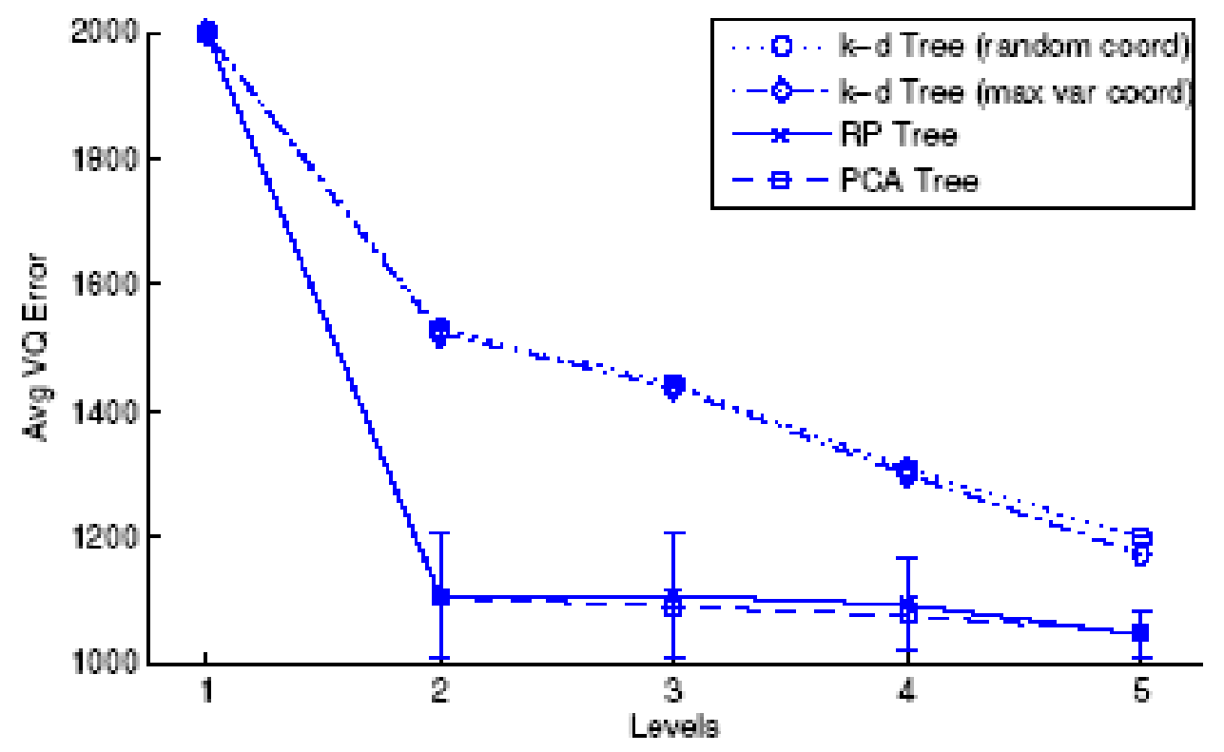
RPTrees quickly reduce the avg. VQ error, where k-D trees fail:

(*i*)



For each $x_i \in \Re^D$:
choose $p_i \sim U[0,1]$
$x_{ij} \sim N\,(p_i,1)$

(*ii*)

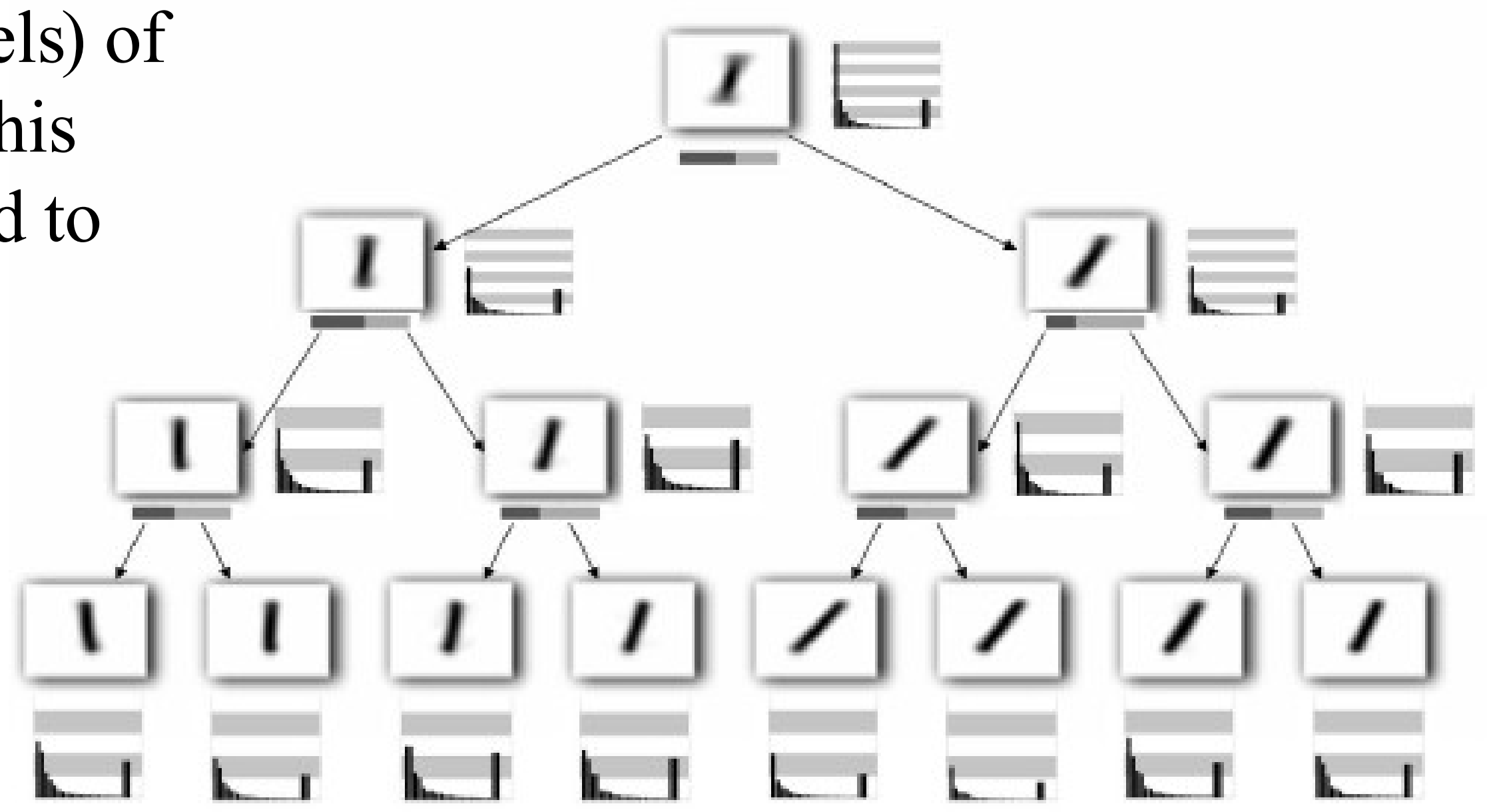


For each $x_i \in \Re^D$:
choose p_i = -1 or +1
(with equal probability)
$x_{ij} \sim N\,(p_i,1)$

MNIST dataset (digit 1 cluster)

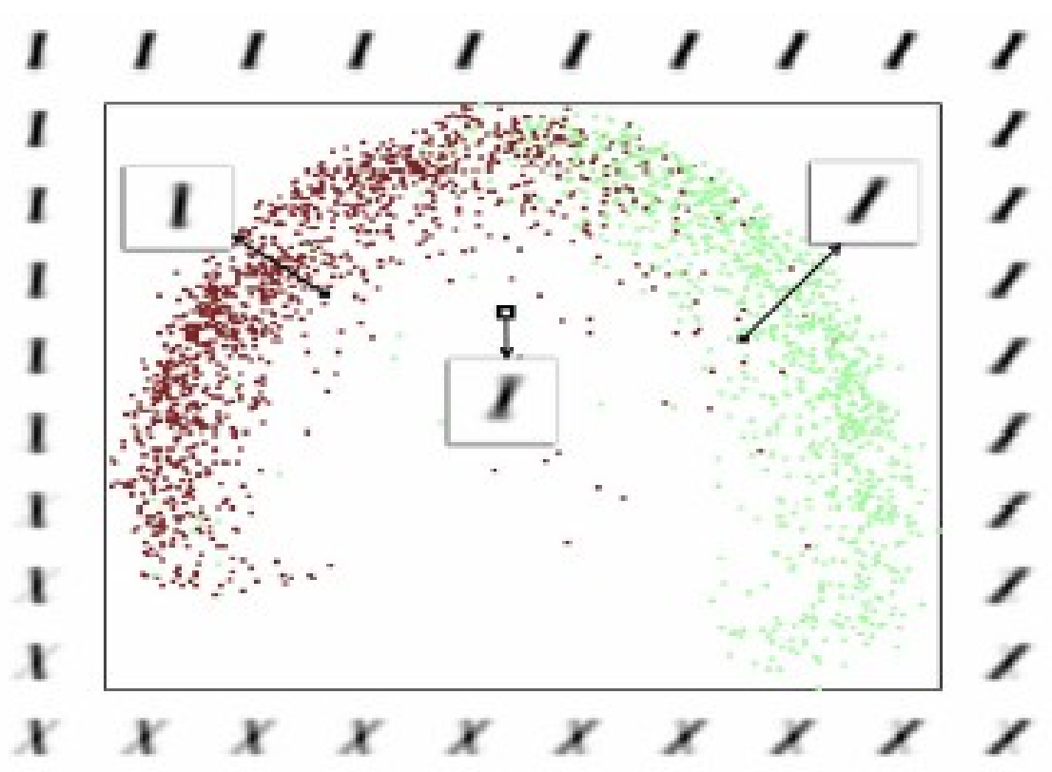
2000 images (28x28 pixels) of handwritten digit one. This dataset is widely believed to have a low intrinsic dimension.

RPTree datastructure (right) adapts very well to the underlying manifold.

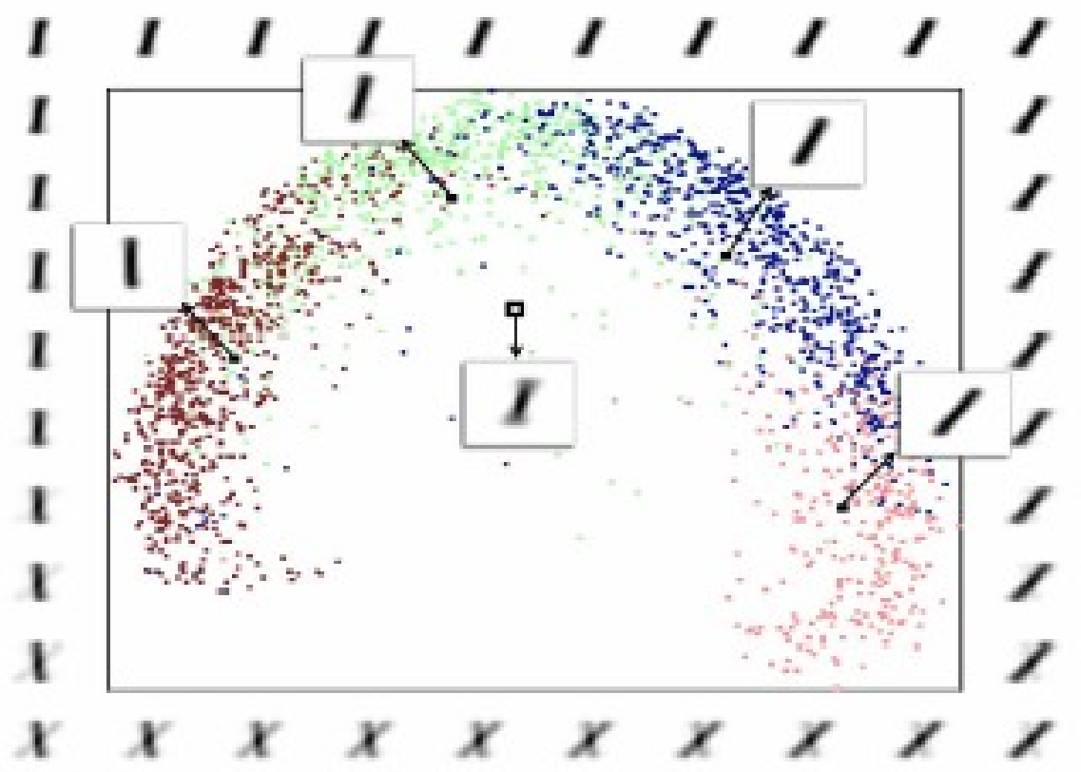


Visualizing the data in the top principal components. (Colors represent different cells of the RPTree)

Note that even though we choose a random direction to partition the data, it is as good as using the PCA direction.



1st level of RPTree

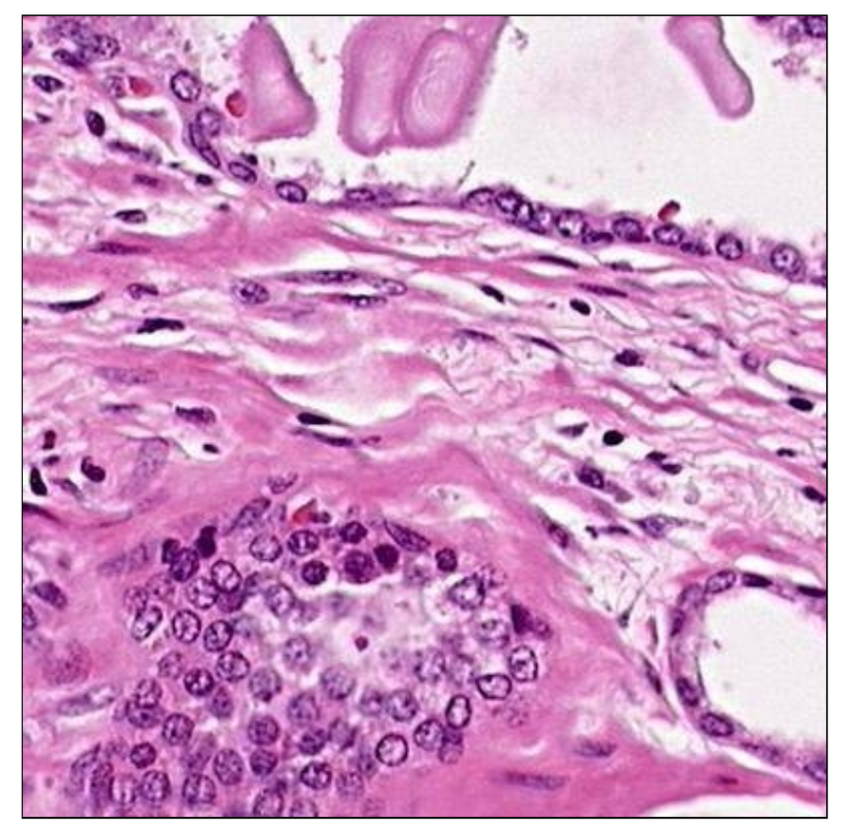


2nd level of RPTree

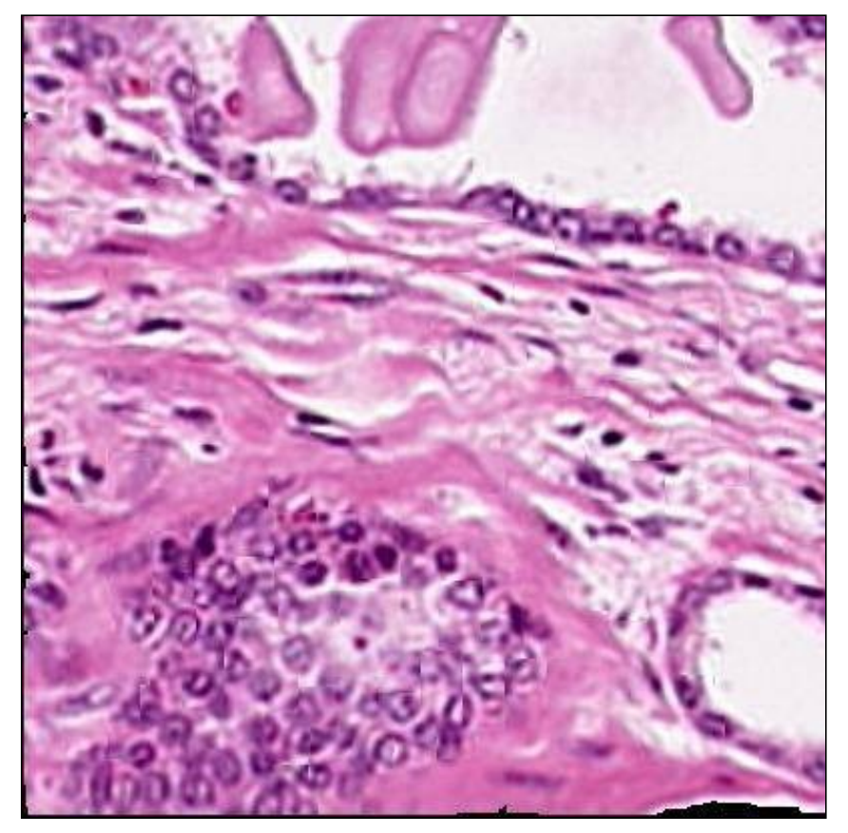
Compression of pathology images

RPTrees learn the structure of the underlying image manifold from a specific domain. Preliminary results show good reconstruction accuracy.

RPTree is trained on data generated by sliding a 20x20 window over the training images.
For compressing a test image, we encode a few representative leaf cells and its deviations from the mean.



Original Image



Reconstructed Image