Applications of Random Matrices in Spectral Computations and Machine Learning

Dimitris Achlioptas

UC Santa Cruz

This talk

Viewpoint:

use randomness to “transform” the data

This talk

Viewpoint:
use randomness to "transform" the data

- Random Projections
- Fast Spectral Computations
- Sampling in Kernel PCA

The Setting

Input: Set S of n points in $\mathbf{R}^d$ — d is "too big"

Output: Set S' of n points in $\mathbf{R}^k$ which is "like" S — k is "affordable/right"

The Setting

Input: Set S of n points in $\mathbf{R}^d$ — d is "too big"

Output: Set S' of n points in $\mathbf{R}^k$ which is "like" S — k is "affordable/right"

We will always represent n points in $\mathbf{R}^d$ as an $n \times d$ matrix.

$$S \longrightarrow A \in \mathbf{R}^{n \times d}$$

The Setting

Input: Set S of n points in $\mathbf{R}^d$ — d is "too big"

Output: Set S' of n points in $\mathbf{R}^k$ which is "like" S — k is "affordable/right"

We will always represent n points in $\mathbf{R}^d$ as an $n \times d$ matrix.

$$S \to A \in \mathbf{R}^{n \times d}$$

Solution 1: Compute the SVD of $A = UDV^T$

The Setting

Input: Set S of n points in $\mathbf{R}^d$ — d is "too big"

Output: Set S' of n points in $\mathbf{R}^k$ which is "like" S — k is "affordable/right"

We will always represent n points in $\mathbf{R}^d$ as an $n \times d$ matrix.

$$S \longrightarrow A \in \mathbf{R}^{n \times d}$$

Solution 1: Compute the SVD of $A = UDV^T$

Solution 2: Compute? Naah.. Flip coins to form $P \in \mathbf{R}^{d \times k}$

Output: AP

The Johnson-Lindenstrauss lemma

JL-lemma: For every set S of n points in $\mathbf{R}^d$ and every $\epsilon > 0$ there exists $f : \mathbf{R}^d \to \mathbf{R}^k$, where $k = O(\epsilon^{-2} \log n)$, such that for all pairs $u, v \in S$

$$(1-\epsilon)|u-v|^2 \leq |f(u) - f(v)|^2 \leq (1+\epsilon)|u-v|^2 \ .$$

The Johnson-Lindenstrauss lemma

JL-lemma: For every set S of n points in $\mathbf{R}^d$ and every $\epsilon > 0$ there exists $f : \mathbf{R}^d \to \mathbf{R}^k$, where $k = O(\epsilon^{-2} \log n)$, such that for all pairs $u, v \in S$

$$(1-\epsilon)|u-v|^2 \le |f(u) - f(v)|^2 \le (1+\epsilon)|u-v|^2 \ .$$

Algorithm:

Projecting onto a random hyperplane (subspace) of dimension

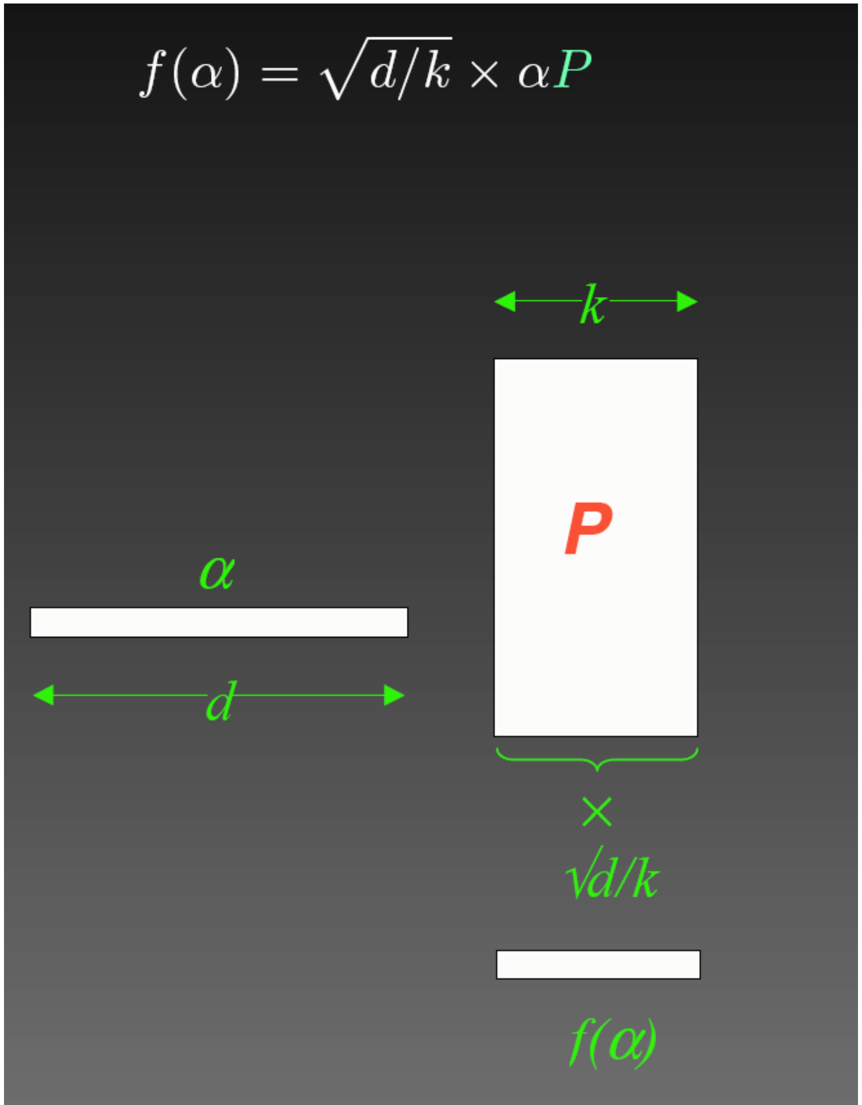
$$k = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$$

succeeds with probability $1 - 1/n^\beta$

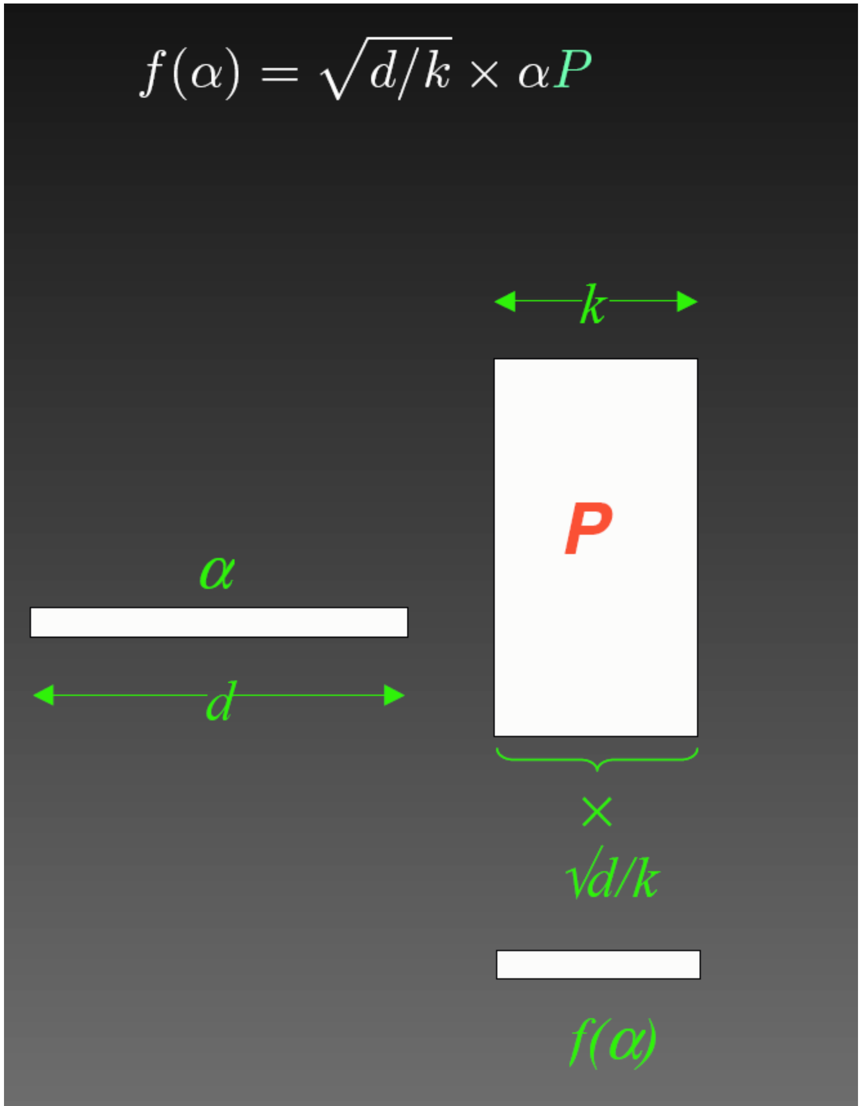
Applications

- Approximation algorithms [Charikar'02]
- Hardness of approximation [Trevisan '97]
- Learning mixtures of Gaussians [Arora, Kannan '01]
- Approximate nearest-neighbors [Kleinberg '97]
- Data-stream computations [Alon et al. '99, Indyk '00]
- Min-cost clustering [Schulman '00]
- ….
- Information Retrieval (LSI) [Papadimitriou et al. '97]

How to pick a random hyperplane



How to pick a random hyperplane

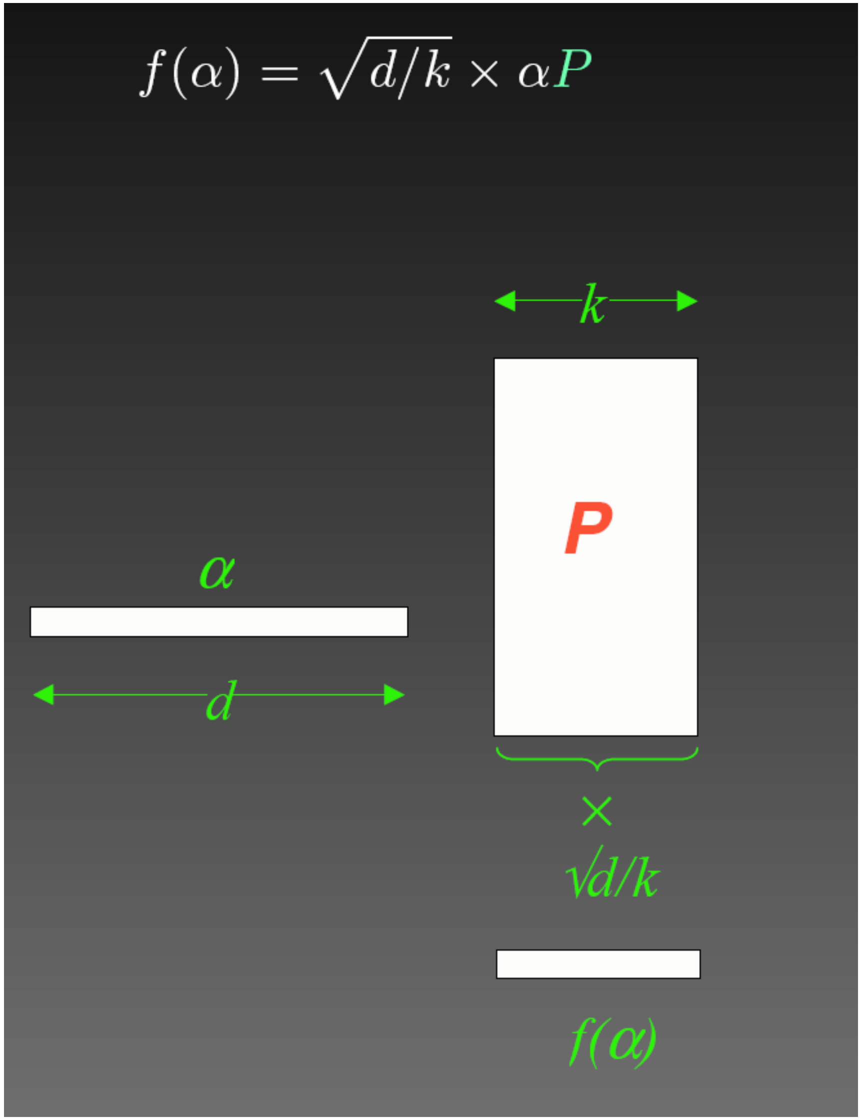


- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables
- $P \leftarrow \mathrm{Orthonormalize}(P)$

[Indyk Motwani 99]

[Johnson Lindenstrauss 82]

How to pick a random hyperplane

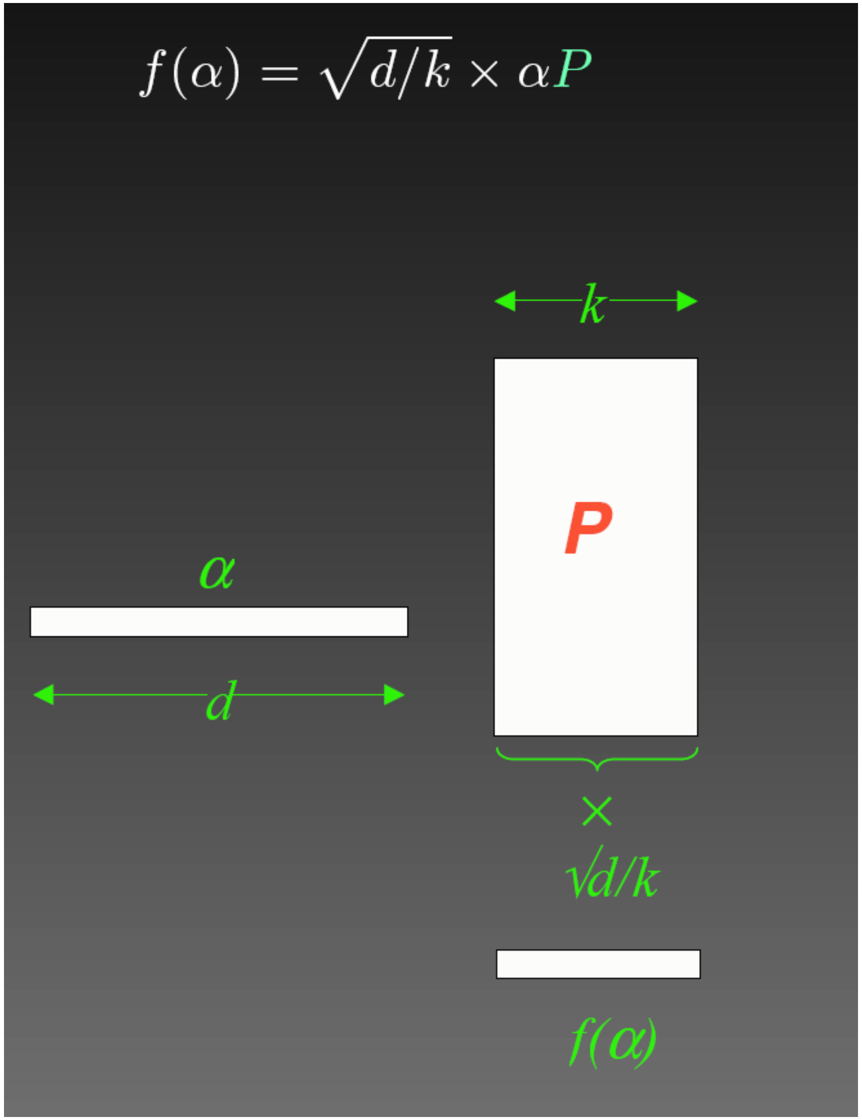


- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables

Intuition:

- Each column of *P* points to a uniformly random direction in $\mathbf{R}^d$.

How to pick a random hyperplane



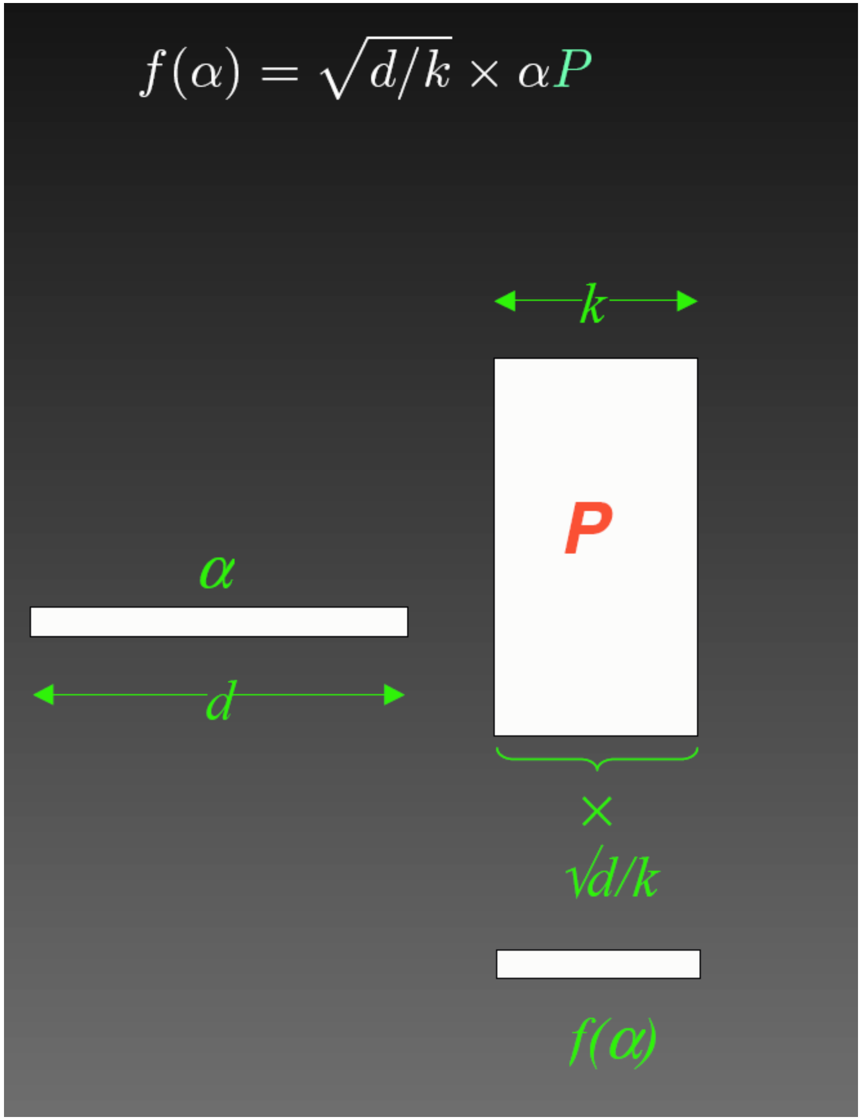
- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables

Intuition:

- Each column of *P* points to a uniformly random direction in $\mathbf{R}^d$.
- Each column is an unbiased, independent estimator of $|\alpha|^2$

(via its squared inner product)

How to pick a random hyperplane

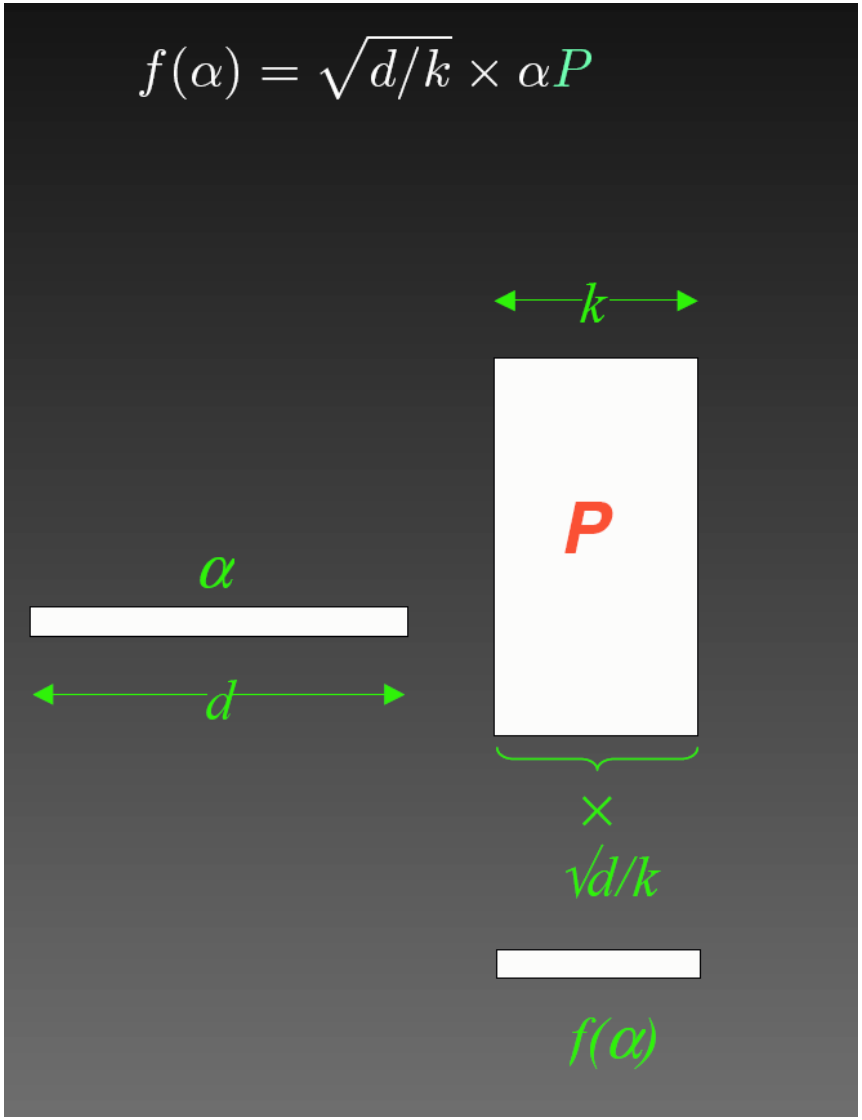


- Take $P(i, j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0, 1)$ random variables

Intuition:

- Each column of P points to a uniformly random direction in $\mathbf{R}^d$.
- Each column is an unbiased, independent estimator of $|\alpha|^2$
  (via its squared inner product)
- $|\alpha P|^2$ is the average estimate
  (since we take the sum)

How to pick a random hyperplane

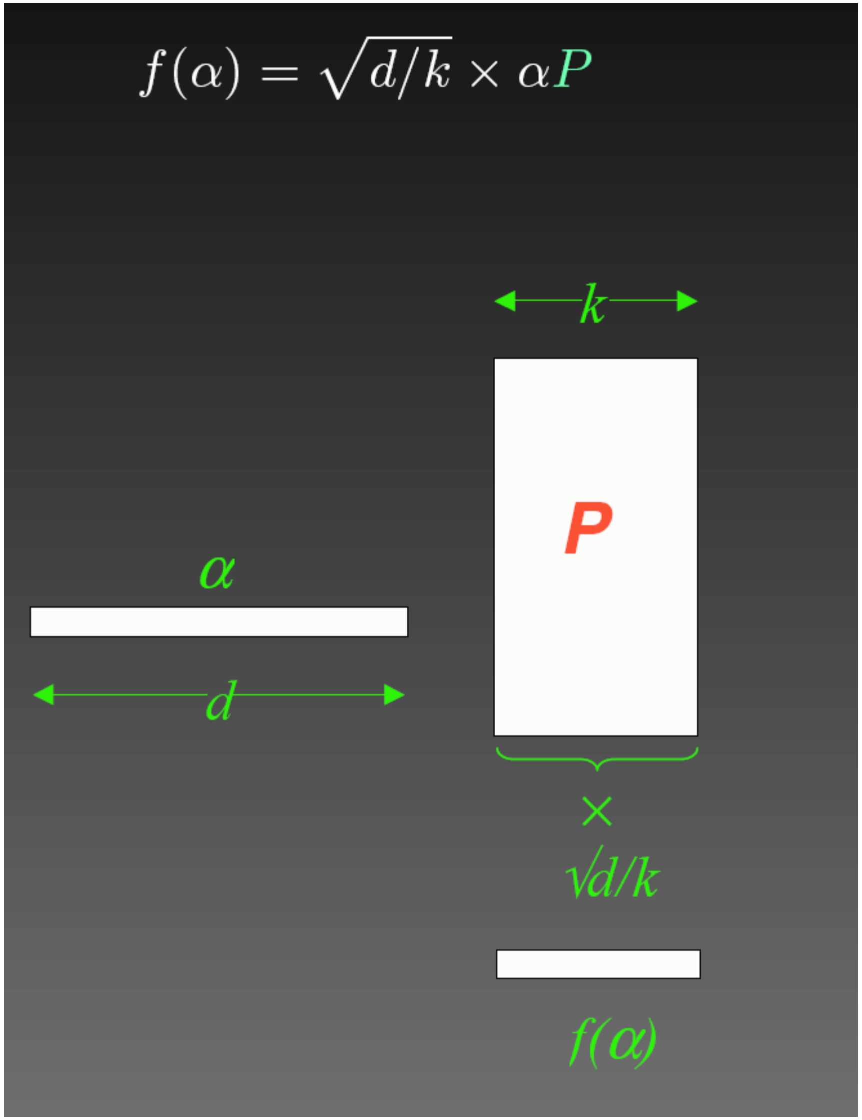


- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables

With orthonormalization:

- Estimators are "equal"
- Estimators are "uncorrelated"

How to pick a random hyperplane



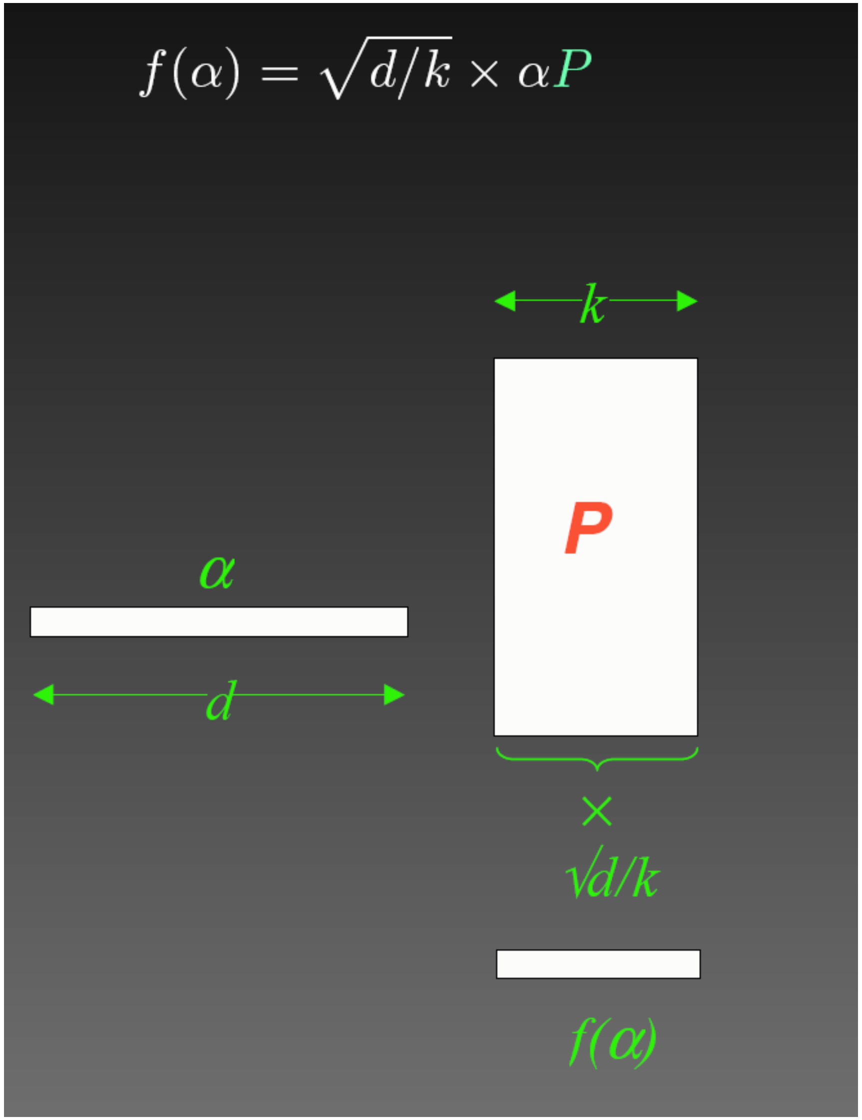
- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables

With orthonormalization:

- Estimators are "equal"
- Estimators are "uncorrelated"

With**out** orthonormalization:

How to pick a random hyperplane



- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent $N(0,1)$ random variables

With orthonormalization:

- Estimators are "equal"
- Estimators are "uncorrelated"

With**out** orthonormalization:

$\boxed{\text{Same thing!}}$

Orthonormality: Take #1

Random vectors in high-dimensional Euclidean space

are very nearly orthonormal.

Orthonormality: Take #1

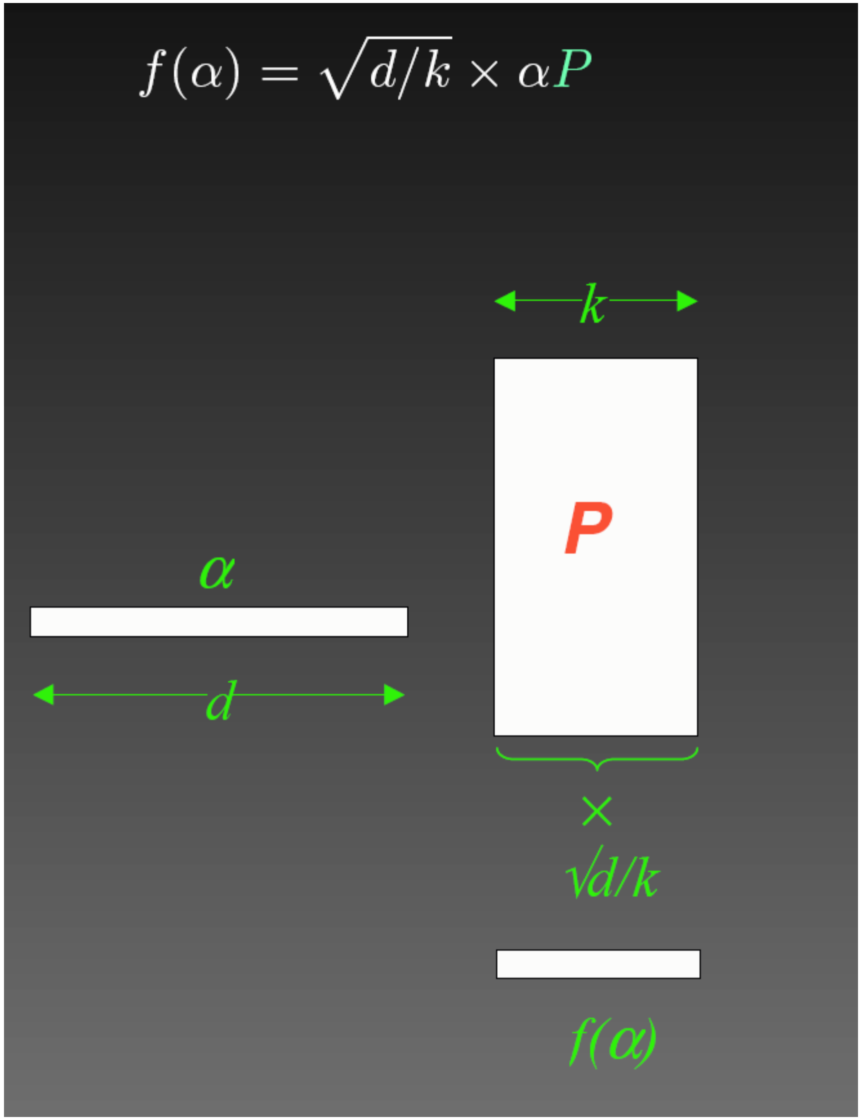
Random vectors in high-dimensional Euclidean space

are very nearly orthonormal.

Do they have to be uniformly random?

Is the Gaussian distribution magical?

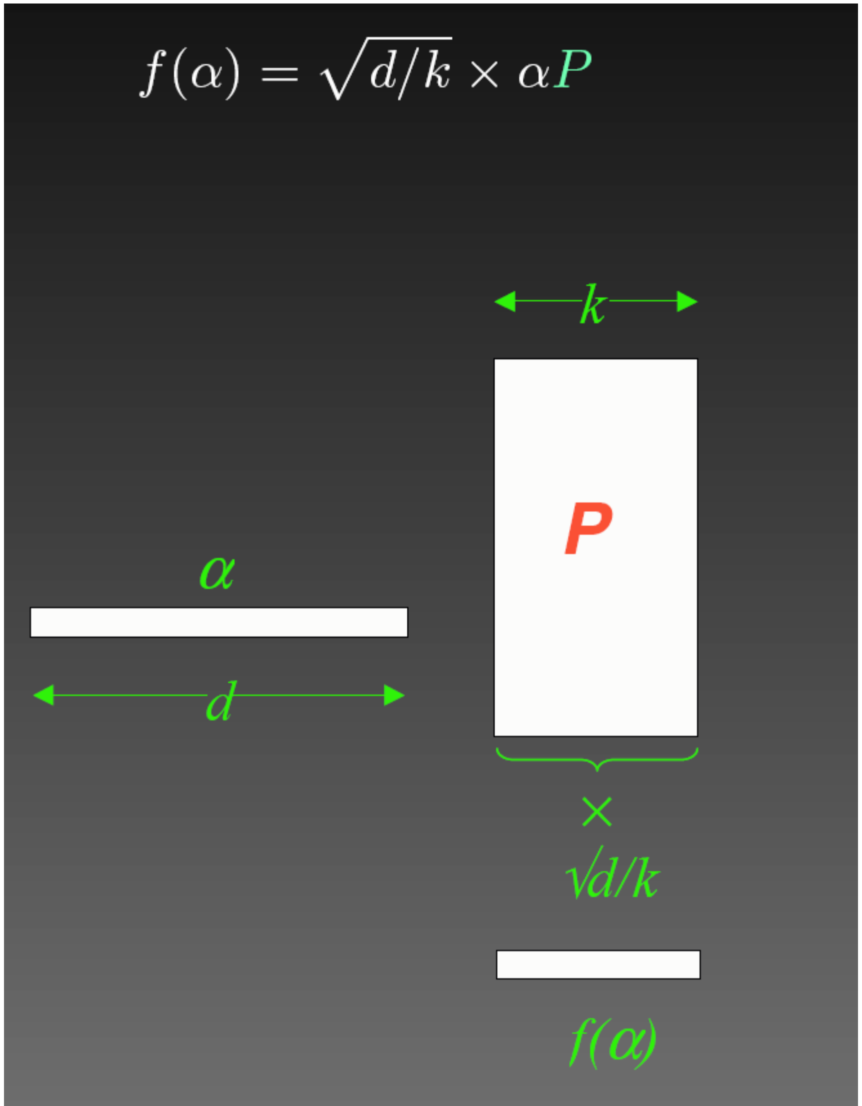
JL with binary coins



- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent random variables with

$$r_{ij} = \begin{cases} +1 & \text{with probability} \quad 1/2 \\ -1 & \quad\quad \text{''} \quad\quad\quad\quad 1/2 \end{cases}$$

JL with binary coins



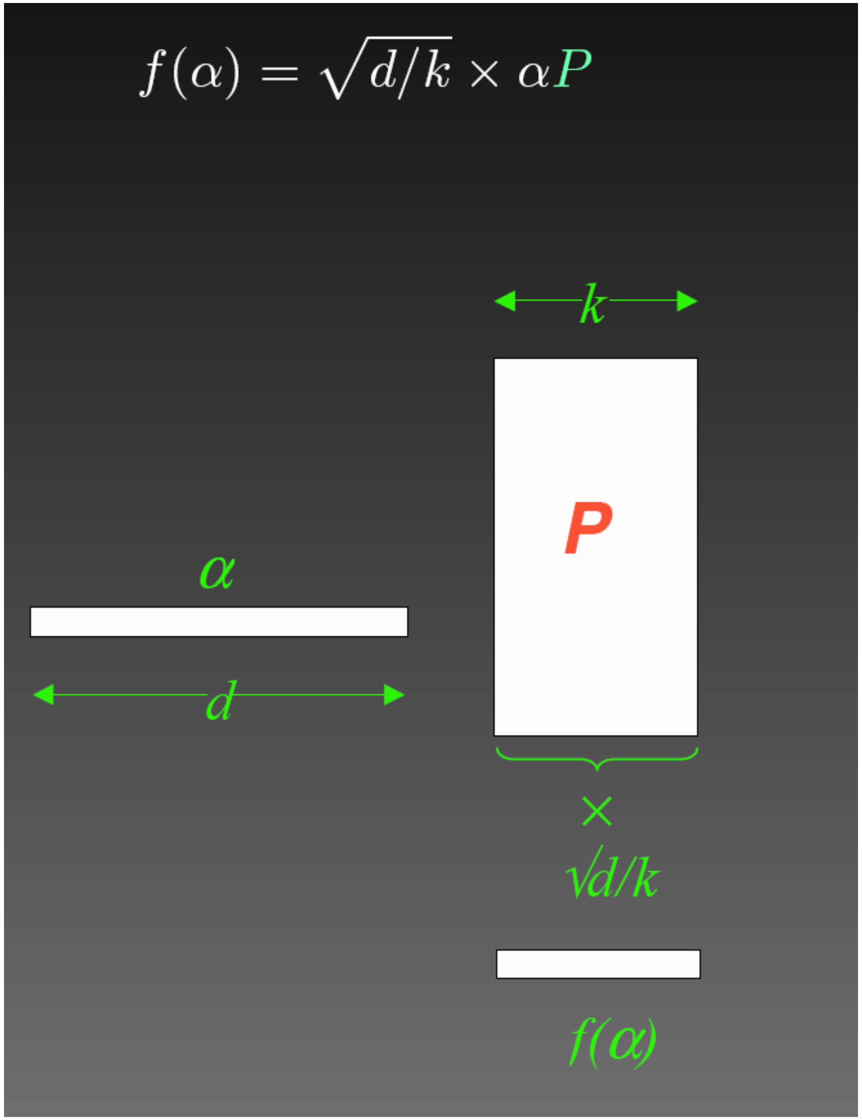
- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent random variables with

$$r_{ij} = \begin{cases} +1 & \text{with probability} \quad 1/2 \\ -1 & \quad \text{''} \quad\quad 1/2 \end{cases}$$

Benefits:

- Much faster in practice
- Only $\pm$ operations (no $*$)
- Fewer random bits
- Derandomization
- Slightly smaller(!) k

JL with binary coins



- Take $P(i,j) = r_{ij}$ where the $\{r_{ij}\}$ are independent random variables with

$$r_{ij} = \begin{cases} +1 & \text{with probability} \quad 1/2 \\ -1 & \quad\quad\quad \text{..} \quad\quad\quad 1/2 \end{cases}$$

- Preprocessing with a randomized FFT

[Ailon, Chazelle '06]

$$O\left(d \log d + \min\{d\varepsilon^{-2} \log n, \varepsilon^{p-4} \log^{p+1} n\}\right)$$

Let's at least

look at the data

The Setting

Input: Set S of n points in $\mathbf{R}^d$ — d is "too big"

Output: Set S' of n points in $\mathbf{R}^k$ which is "like" S — k is "affordable/right"

We will always represent n points in $\mathbf{R}^d$ as an $n \times d$ matrix.

$$S \rightarrow A \in \mathbf{R}^{n \times d}$$

Solution 2: Compute? Naah.. Flip coins to form $P \in \mathbf{R}^{d \times k}$

Output: AP

Low Rank Approximations

Spectral Norm: $$\|A\|_2 = \max_{\|x\|=1} \|Ax\|$$

Low Rank Approximations

Spectral Norm: $$\|A\|_2 = \max_{\|x\|=1} \|Ax\|$$

Frobenius Norm: $$\|A\|_F = \left(\sum_{i,j} A_{ij}^2\right)^{1/2} = \operatorname*{Avg}_{\|x\|=1} \|Ax\|$$

Low Rank Approximations

Spectral Norm: $$\|A\|_2 = \max_{\|x\|=1} \|Ax\|$$

Frobenius Norm: $$\|A\|_F = \left(\sum_{i,j} A_{ij}^2\right)^{1/2} = \operatorname*{Avg}_{\|x\|=1} \|Ax\|$$

For any matrix A, there is a well defined matrix A_k that is the "best" rank k approximation to A for many norms.

Low Rank Approximations

Spectral Norm: $$\|A\|_2 = \max_{\|x\|=1} \|Ax\|$$

Frobenius Norm: $$\|A\|_F = \left(\sum_{i,j} A_{ij}^2 \right)^{1/2} = \operatorname*{Avg}_{\|x\|=1} \|Ax\|$$

For any matrix A, there is a well defined matrix A_k that is the "best" rank k approximation to A for many norms.

For any rank k matrix B,
$$\|A - A_k\|_2 \leq \|A - B\|_2$$
$$\|A - A_k\|_F \leq \|A - B\|_F$$

Low Rank Approximations

Spectral Norm: $\|A\|_2 = \max_{\|x\|=1} \|Ax\|$

Frobenius Norm: $\|A\|_F = \left(\sum_{i,j} A_{ij}^2\right)^{1/2} = \operatorname{Avg}_{\|x\|=1} \|Ax\|$

A_k is the maximizer of $\|AP\|$ over all projections P into $\mathbf{R}^k$

How to compute A_k

- Start with a random $x \in \mathbf{R}^d$

How to compute A_k

- Start with a random $x \in \mathbf{R}^d$
- Repeat until fixpoint

  - Have each row in A vote for x: $y = Ax \in \mathbf{R}^n$

How to compute A_k

- Start with a random $x \in \mathbf{R}^d$
- Repeat until fixpoint

- Have each row in A vote for x: $y = Ax \in \mathbf{R}^n$
- Synthesize a new candidate by combining the rows of A according to their enthusiasm for x:

$$x \leftarrow \frac{A^T y}{||A^T y||} \in \mathbf{R}^d$$

(This is power iteration on $A^T A$. Also known as PCA.)

How to compute A_k

- Start with a random $x \in \mathbf{R}^d$
- Repeat until fixpoint

  - Have each row in A vote for x: $y = Ax \in \mathbf{R}^n$
  - Synthesize a new candidate by combining the rows of A according to their enthusiasm for x:

$$x \leftarrow \frac{A^T y}{\|A^T y\|} \in \mathbf{R}^d$$

(This is power iteration on $A^T A$. Also known as PCA.)

- Project A on subspace orthogonal to x and repeat

PCA for Denoising

- Assume that we perturb the entries of a matrix A by adding independent Gaussian noise $N(0, \sigma)$

$$\widehat{A} = A + G$$

PCA for Denoising

- Assume that we perturb the entries of a matrix A by adding independent Gaussian noise $N(0, \sigma)$

$$\widehat{A} = A + G$$

- **Claim**: If σ is not "too big" then the optimal projections for $\widehat{A}$ are "close" to those for A.

PCA for Denoising

- Assume that we perturb the entries of a matrix A by adding independent Gaussian noise $N(0, \sigma)$

$$\widehat{A} = A + G$$

- **Claim**: If σ is not "too big" then the optimal projections for $\widehat{A}$ are "close" to those for A.

- **Intuition**:
  - The perturbation vectors are nearly orthogonal

PCA for Denoising

- Assume that we perturb the entries of a matrix A by adding independent Gaussian noise $N(0, \sigma)$

$$\widehat{A} = A + G$$

- **Claim**: If σ is not "too big" then the optimal projections for $\widehat{A}$ are "close" to those for A.

- **Intuition**:
  - The perturbation vectors are nearly orthogonal
  - No small subspace accommodates many of them

Rigorously

Lemma: For any matrices A and $\widehat{A}$

$$\|A - \widehat{A}_k\|_2 \;\leq\; \|A - A_k\|_2 + 2\|A - \widehat{A}\|_2$$

Rigorously

Lemma: For any matrices A and $\widehat{A}$

$$||A - \widehat{A}_k||_2 \ \leq \ ||A - A_k||_2 + 2||A - \widehat{A}||_2$$

Theorem [Füredi Komlos] Let R be an $n \times d$ random matrix whose entries are independent random variables with mean 0 and variance at most σ^2. Then with [very] high probability,

$$||R||_2 \leq 4\sigma\sqrt{n}$$

Perspective: For any fixed x we have $||Rx||_2 \sim \sigma\sqrt{n}$ w.h.p.

Two new ideas

- A rigorous criterion for choosing k:

Stop when A-A_k has
“as much structure as” a random matrix

Two new ideas

- A rigorous criterion for choosing k:

> Stop when A-A_k has
> "as much structure as" a random matrix

- Computation-friendly **noise**:

Two new ideas

- A rigorous criterion for choosing k:

> Stop when A-A_k has
> "as much structure as" a random matrix

- Computation-friendly noise:

> **Inject data-dependent noise**

Quantization

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} +1 & \text{with probability } 1/2 + A_{ij}/2 \\ \\ -1 & \text{with probability } 1/2 - A_{ij}/2 \end{cases}$$

Quantization

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} +1 & \text{with probability } 1/2 + A_{ij}/2 \\ -1 & \text{with probability } 1/2 - A_{ij}/2 \end{cases}$$

- The expected value of $\widehat{A}_{ij}$ is A_{ij}.

Quantization

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} +1 & \text{with probability } 1/2 + A_{ij}/2 \\ -1 & \text{with probability } 1/2 - A_{ij}/2 \end{cases}$$

- The expected value of $\widehat{A}_{ij}$ is A_{ij}.
- The variance of each $\widehat{A}_{ij}$ is at most 1.

Quantization

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} +1 & \text{with probability } 1/2 + A_{ij}/2 \\ -1 & \text{with probability } 1/2 - A_{ij}/2 \end{cases}$$

- The expected value of $\widehat{A}_{ij}$ is A_{ij}.
- The variance of each $\widehat{A}_{ij}$ is at most 1.
- Each entry in $\widehat{A}$ can be represented by a single bit.

Sparsification

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} \;=\; \begin{cases} A_{ij}/p & \text{with probability } p \\[2ex] 0 & \text{with probability } 1-p \end{cases}$$

for some $0 < p < 1$.

Sparsification

Consider the matrix $\hat{A}$, defined as

$$\hat{A}_{ij} = \begin{cases} A_{ij}/p & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

for some $0 < p < 1$.

- The expected value of $\hat{A}_{ij}$ is A_{ij}.

Sparsification

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} A_{ij}/p & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

for some $0 < p < 1$.

- The expected value of $\widehat{A}_{ij}$ is A_{ij}.
- The variance of each $\widehat{A}_{ij}$ is at most $1/p$.

Sparsification

Consider the matrix $\widehat{A}$, defined as

$$\widehat{A}_{ij} = \begin{cases} A_{ij}/p & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

for some $0 < p < 1$.

- The expected value of $\widehat{A}_{ij}$ is A_{ij}.
- The variance of each $\widehat{A}_{ij}$ is at most $1/p$.
- $\widehat{A}$ is much sparser than A.

Accelerating spectral computations

- By injecting sparsification/quantization “noise” we can accelerate spectral computations:
  - Fewer/simpler arithmetic operations
  - Reduced memory footprint

Accelerating spectral computations

- By injecting sparsification/quantization “noise” we can accelerate spectral computations:
    - Fewer/simpler arithmetic operations
    - Reduced memory footprint
- Amount of “noise” that can be tolerated increases with redundancy in data

Accelerating spectral computations

- By injecting sparsification/quantization “noise” we can accelerate spectral computations:
  - Fewer/simpler arithmetic operations
  - Reduced memory footprint

- Amount of “noise” that can be tolerated increases with redundancy in data

- L2 error can be quadratically better than “Nystrom”

Orthonormality: Take #2

Matrices with independent, 0-mean entries

are

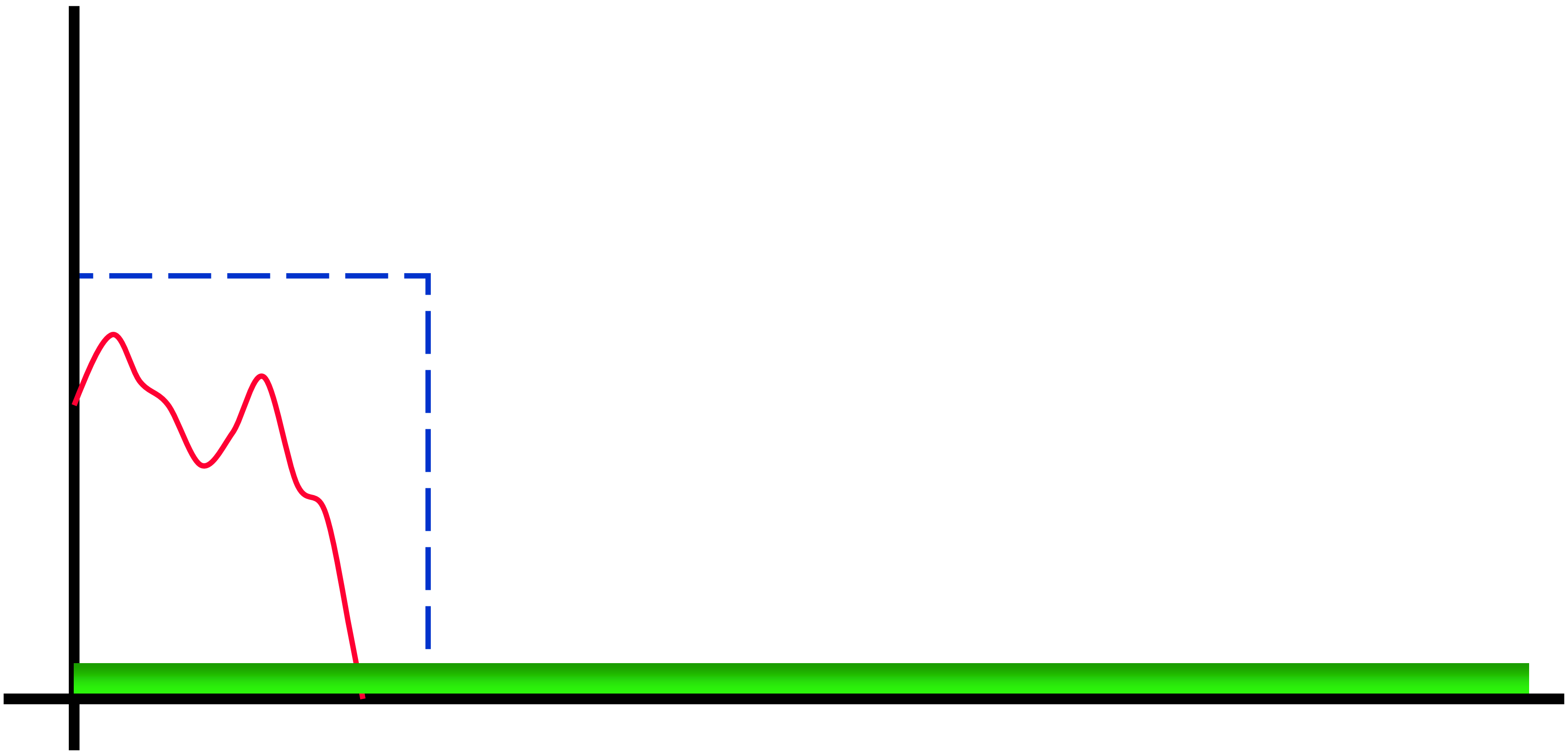
"white noise" matrices

A scalar analogue

A scalar analogue

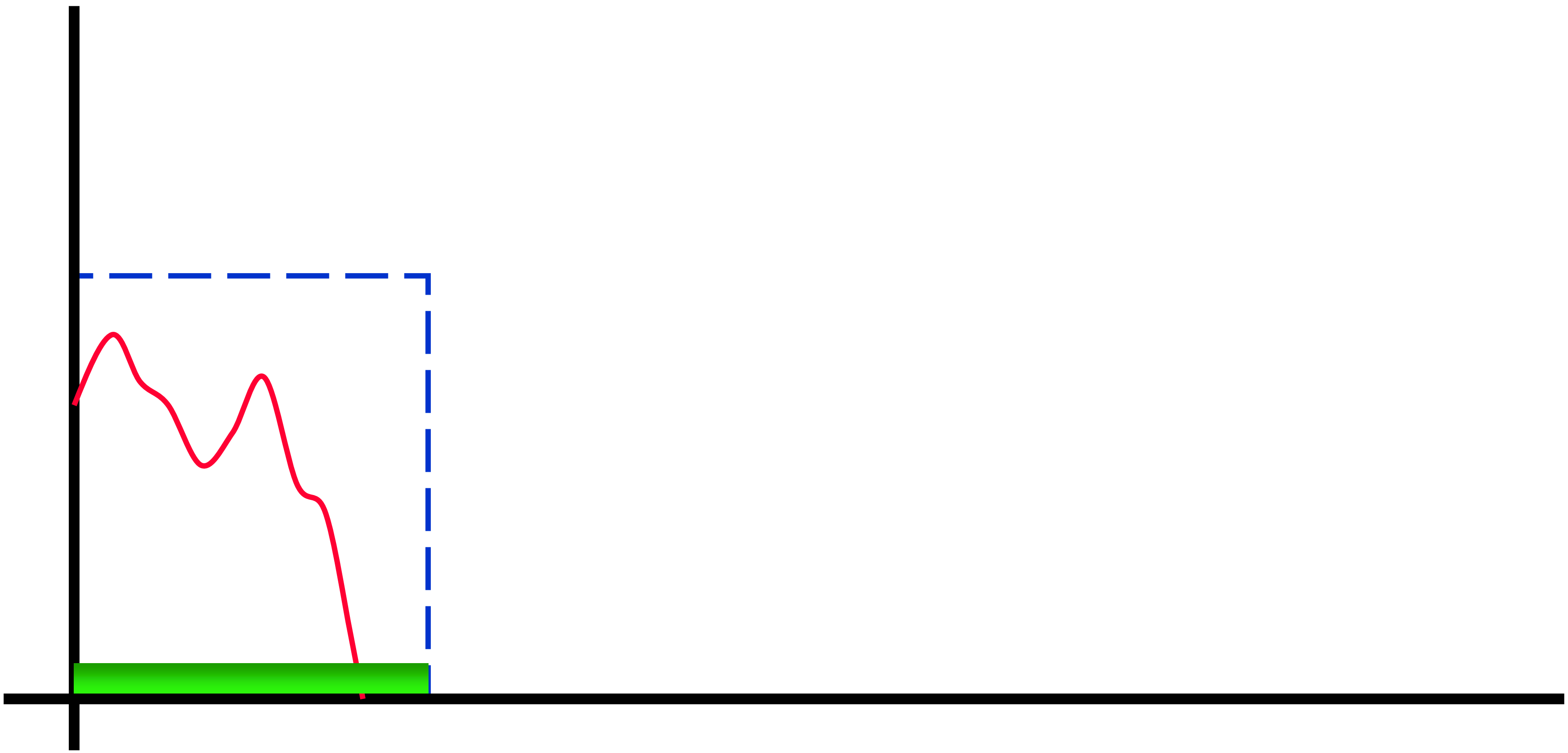
Crude quantization at extremely high rate

A scalar analogue



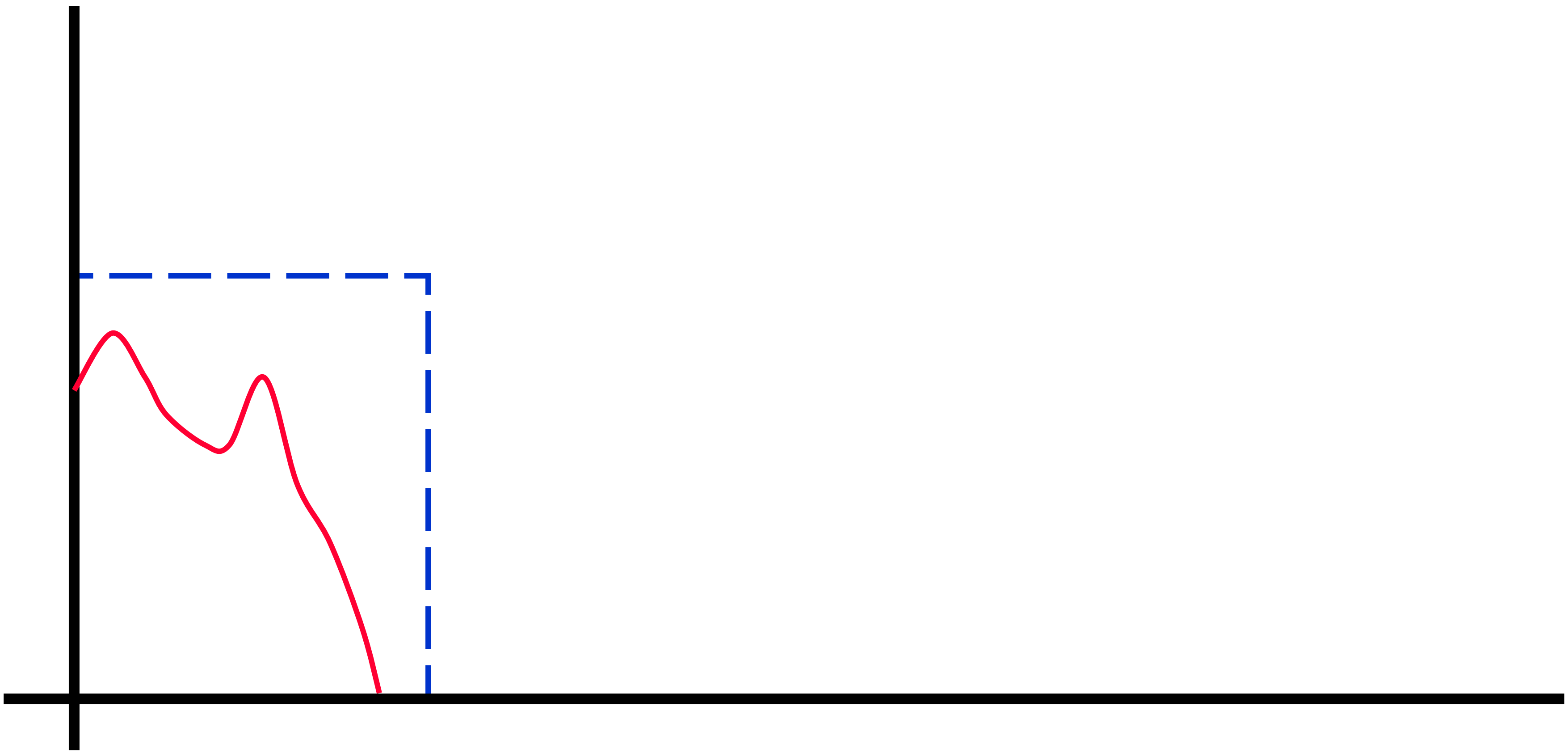
Crude quantization at extremely high rate + low-pass filter

A scalar analogue



Crude quantization at extremely high rate + low-pass filter

A scalar analogue



Crude quantization at extremely high rate + low-pass filter = 1-bit CD player ("Bitstream")

Accelerating spectral computations

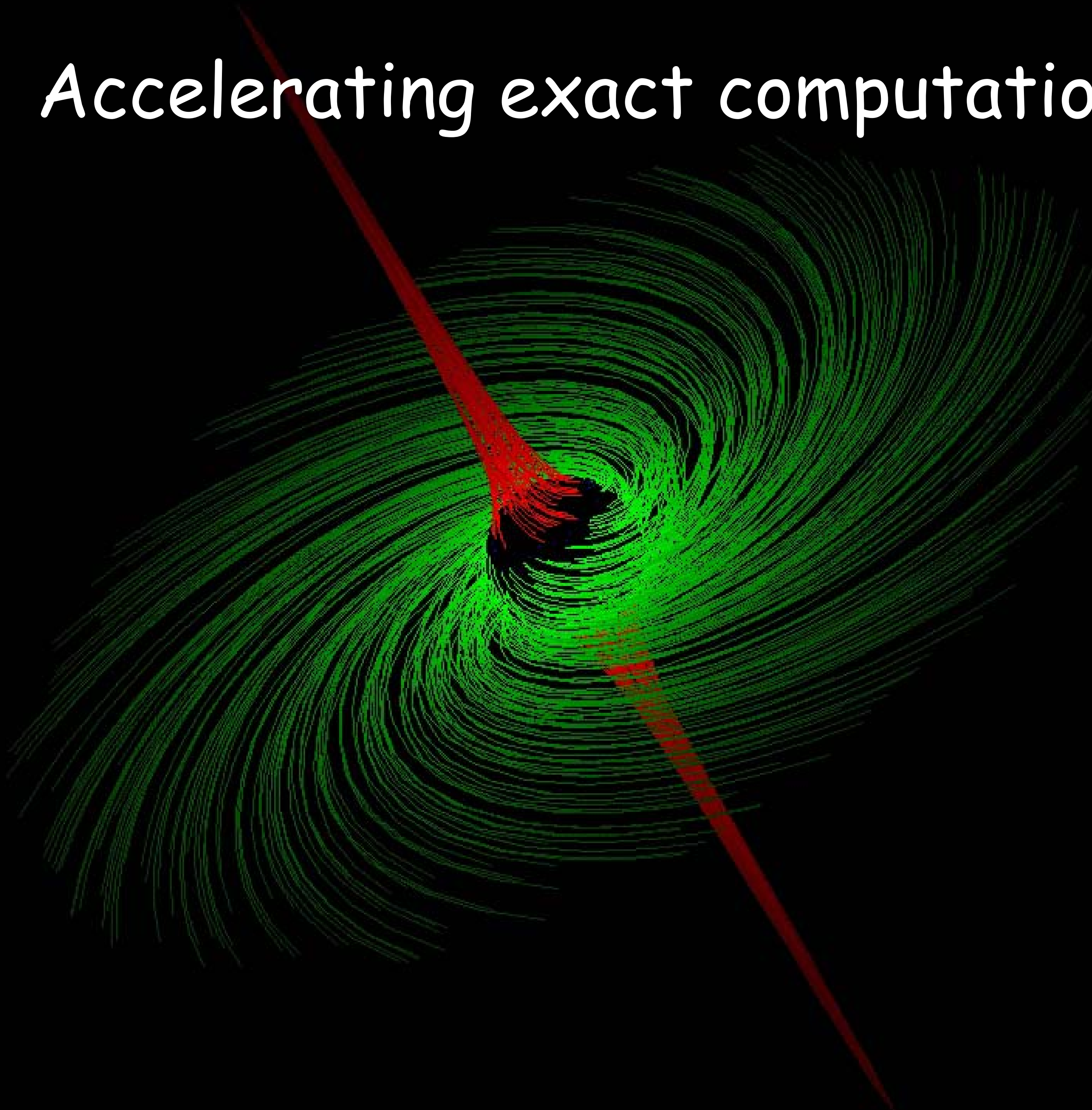
- By injecting sparsification/quantization “noise” we can accelerate spectral computations:
  - Fewer/simpler arithmetic operations
  - Reduced memory footprint

- Amount of “noise” that can be tolerated increases with redundancy in data

- L2 error can be quadratically better than “Nystrom”

- Useful even for exact computations

Accelerating exact computations

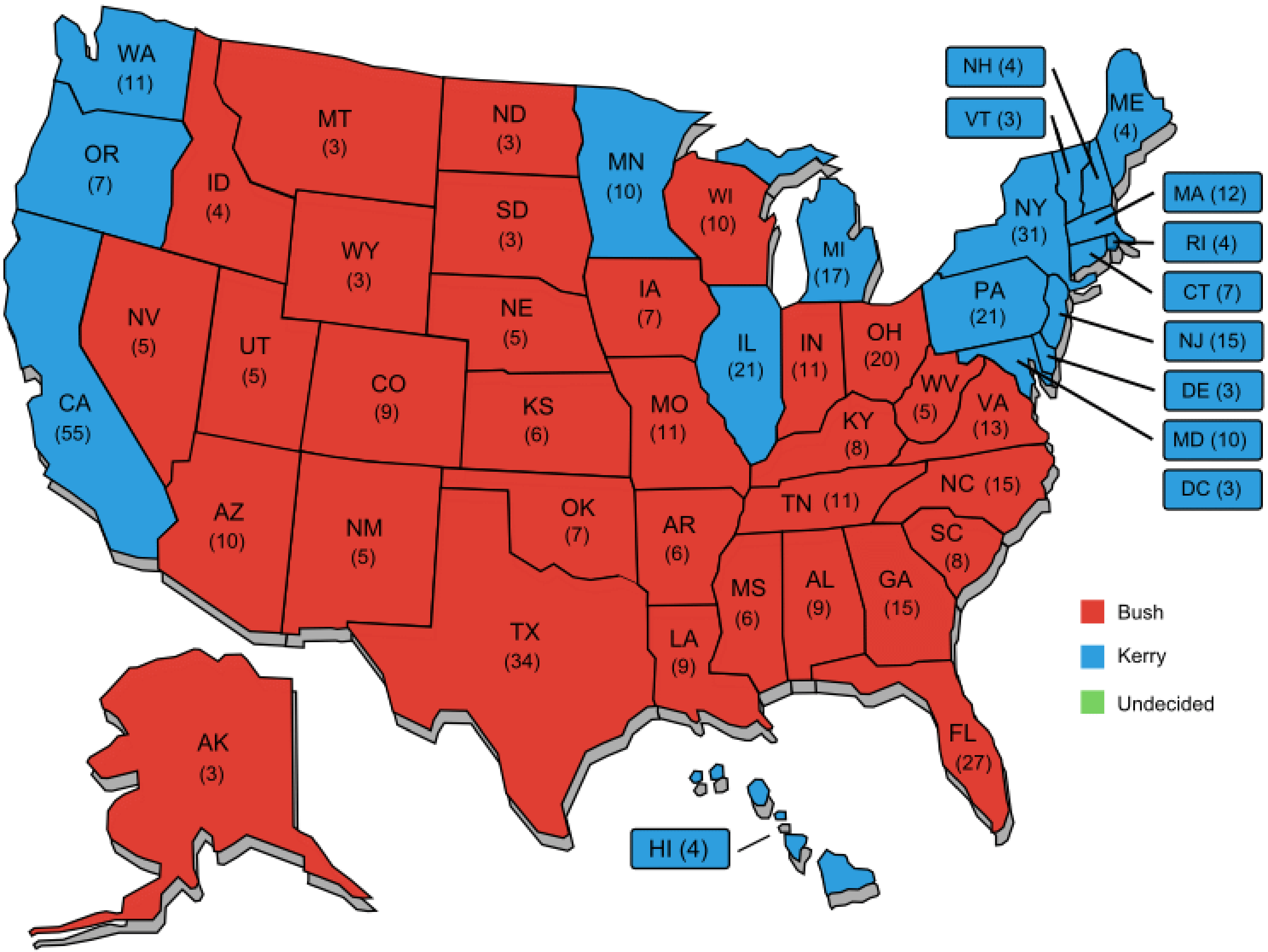


Kernels

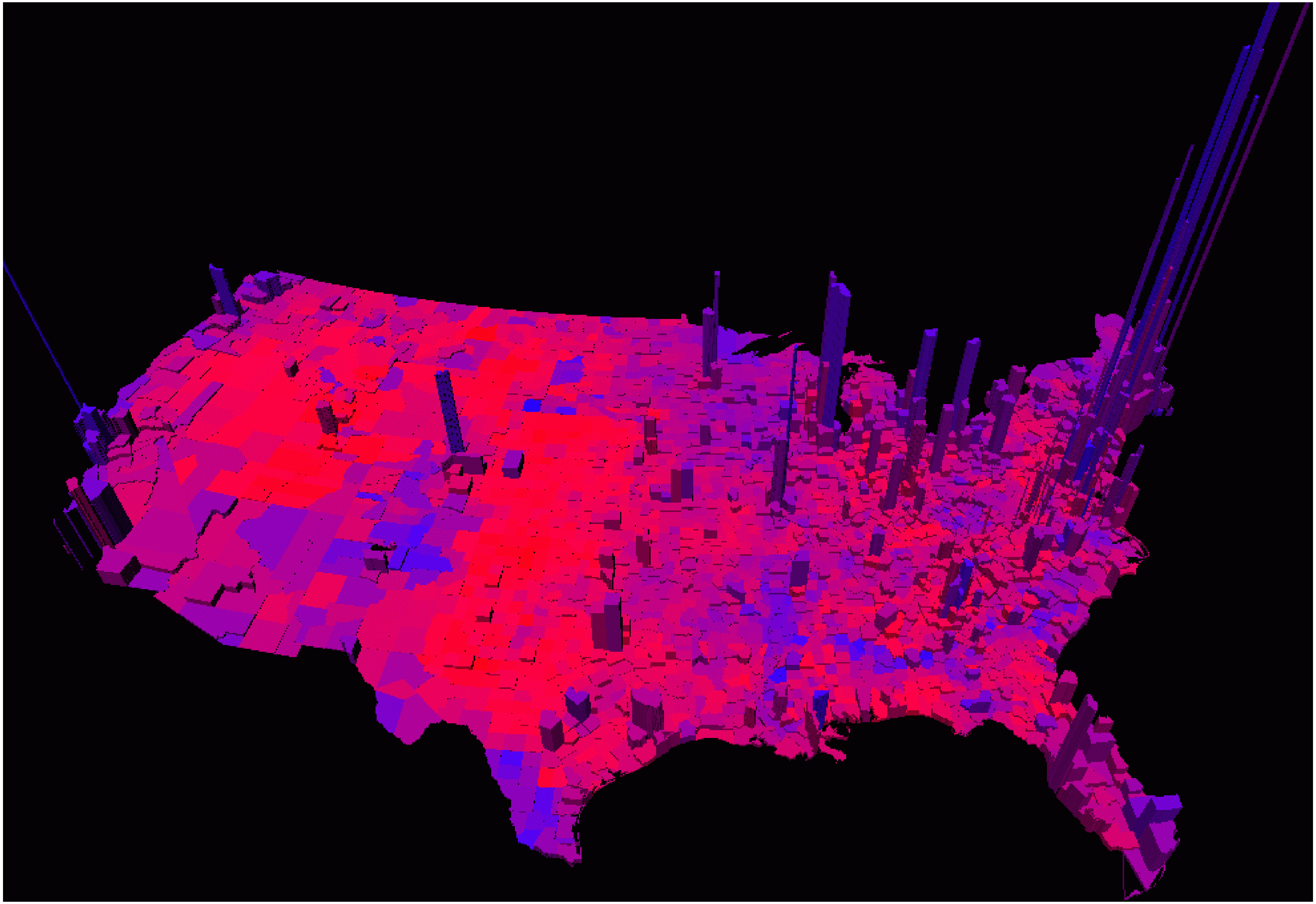
Kernels & Support Vector Machines

- Red and Blue pointclouds
- Which linear separator (hyperplane)?
- Maximum margin
- Optimal can be expressed by inner products with (a few) data points

Not always linearly separable



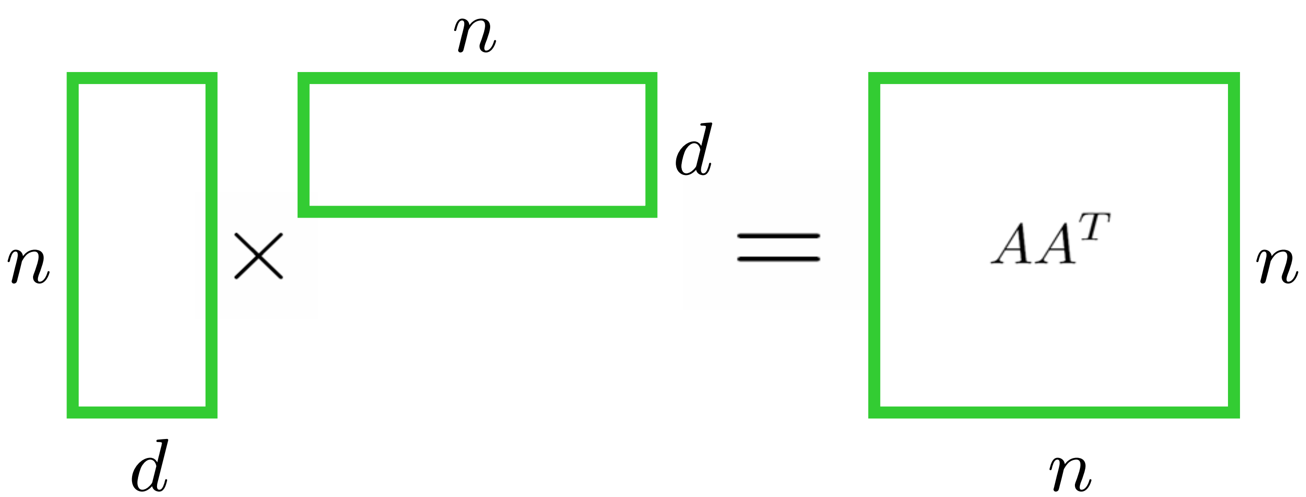
Population density



Kernel PCA

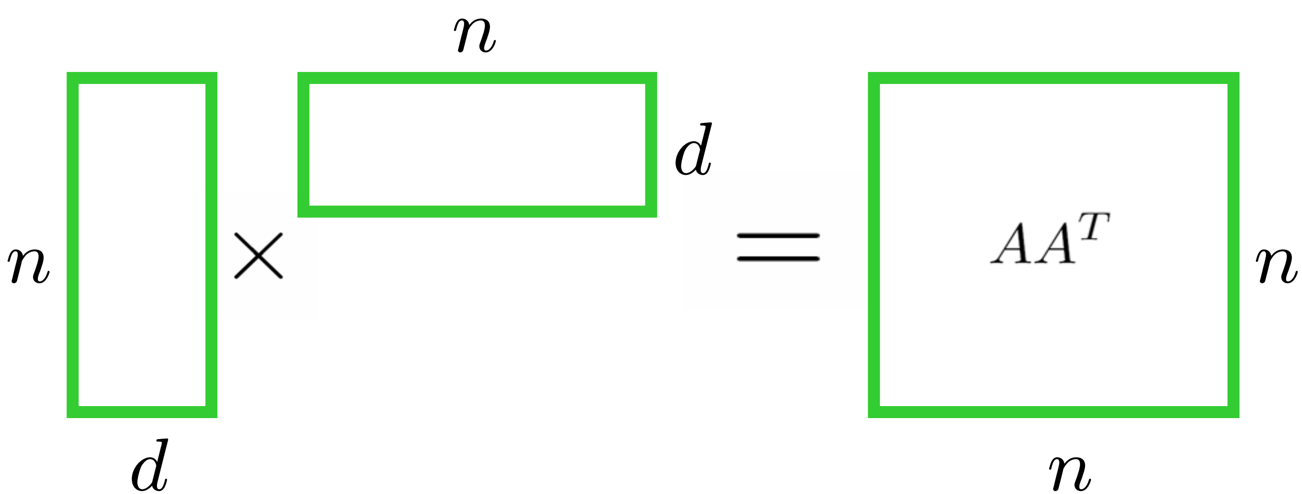
Kernel PCA

We can also compute the SVD via the spectrum of



Kernel PCA

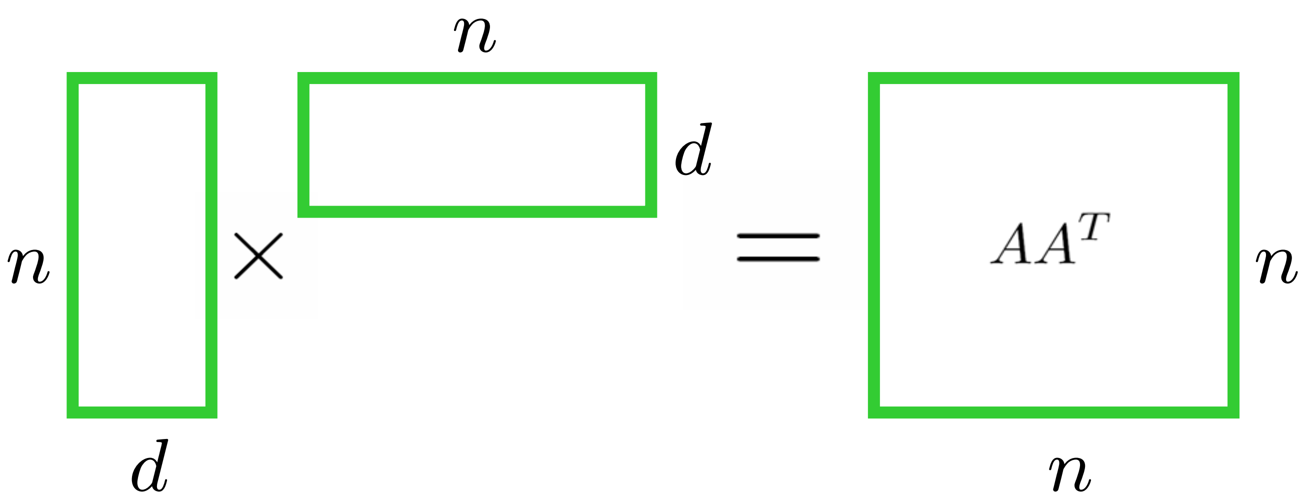
We can also compute the SVD via the spectrum of



- Each entry in AA^T is the inner product of two inputs

Kernel PCA

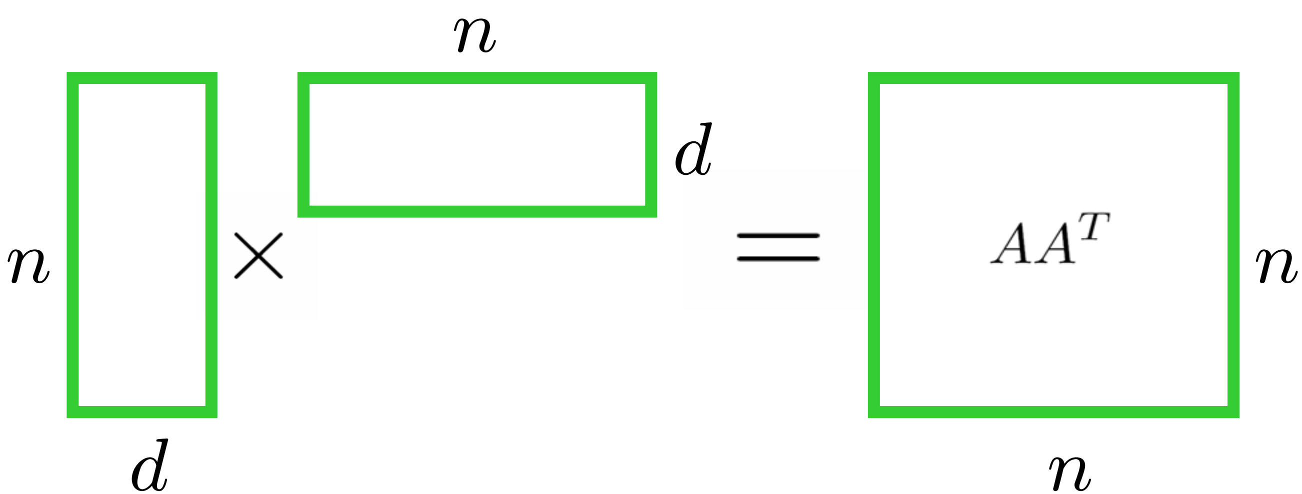
We can also compute the SVD via the spectrum of



- Each entry in AA^T is the inner product of two inputs
- Replace inner product with a kernel function

Kernel PCA

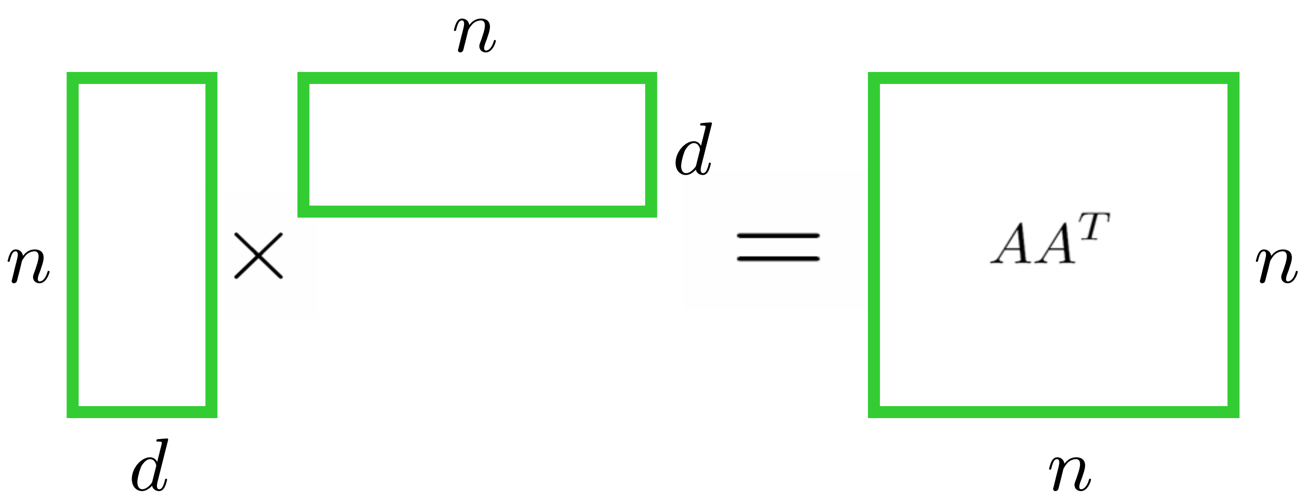
We can also compute the SVD via the spectrum of



- Each entry in AA^T is the inner product of two inputs
- Replace inner product with a kernel function
- Work implicitly in high-dimensional space

Kernel PCA

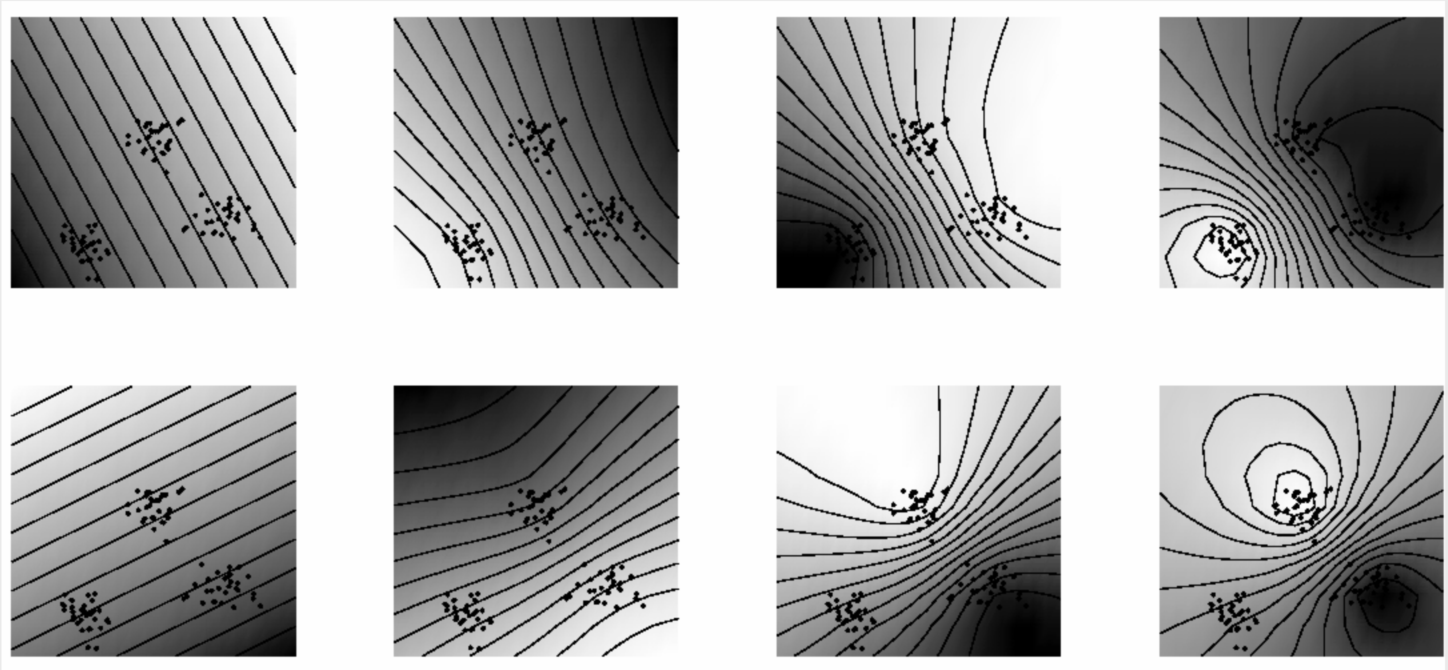
We can also compute the SVD via the spectrum of



- Each entry in AA^T is the inner product of two inputs
- Replace inner product with a kernel function
- Work implicitly in high-dimensional space
- Good linear separators in that space

From linear to non-linear PCA

$||X-Y||^p$ kernel illustrates how the contours of the first 2 components change from straight lines for p=2 to non-linea for p=1.5, 1 and 0.5.

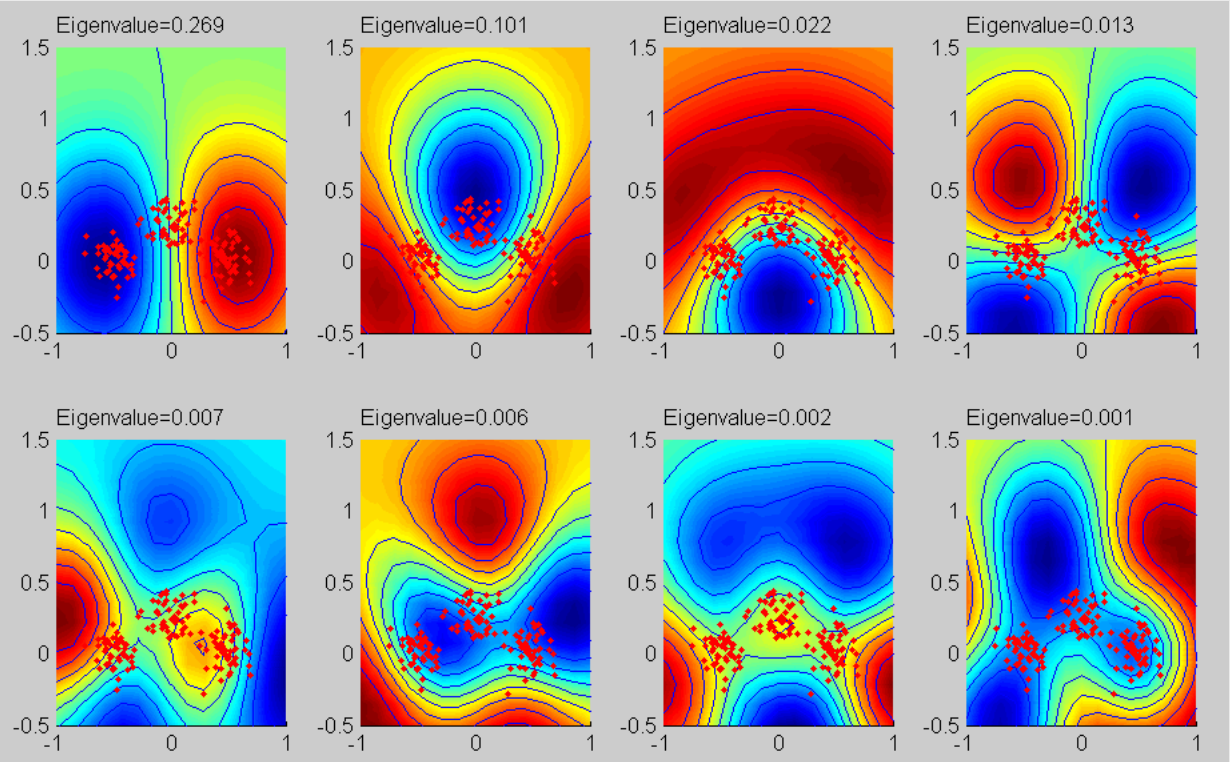


From Schölkopf and Smola, Learning with kernels, MIT 2002

Kernel PCA with Gaussian Kernel

KPCA with Gaussian kernels. The contours follow the cluster densities!

First two kernel PCs separate the data nicely.



Linear PCA has only 2 components, but kernel PCA has more, since the space dimension is usually large (in this case infinite)

KPCA in brief

Good News:

- Work directly with non-vectorial inputs

KPCA in brief

Good News:

- Work directly with non-vectorial inputs
- Very powerful:

  e.g. LLE, Isomap, Laplacian Eigenmaps [Ham et al. '03]

KPCA in brief

Good News:

- Work directly with non-vectorial inputs
- Very powerful:

e.g. LLE, Isomap, Laplacian Eigenmaps [Ham et al. '03]

Bad News:

n^2 kernel evaluations are too many….

KPCA in brief

Good News:

- Work directly with non-vectorial inputs
- Very powerful:

e.g. LLE, Isomap, Laplacian Eigenmaps [Ham et al. '03]

Bad News:

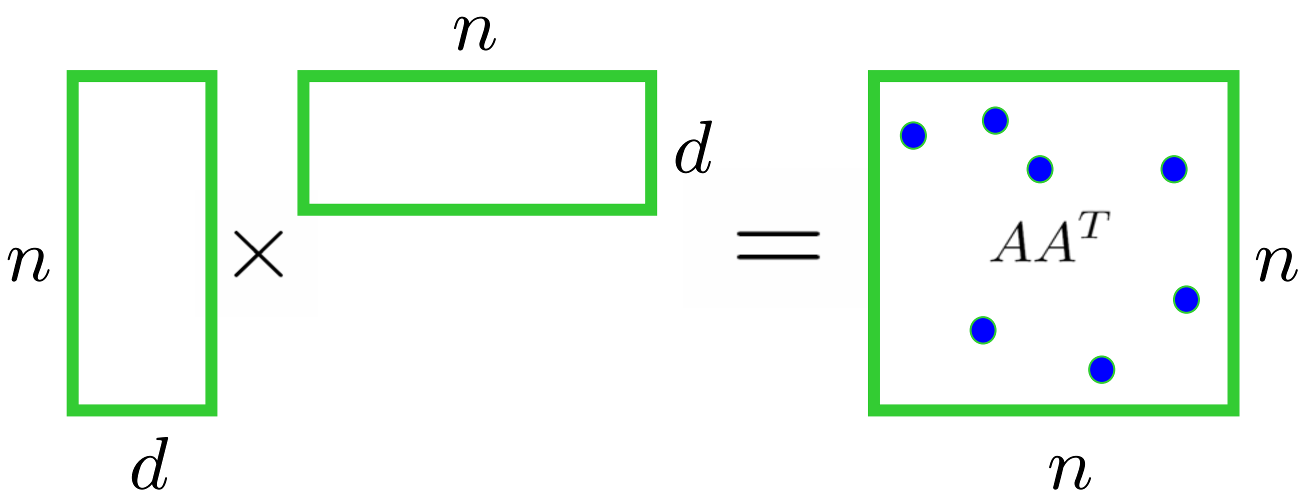
n^2 kernel evaluations are too many….

Good News:

[Shaw-Taylor et al. '03]

good generalization $\Longleftrightarrow$ rapid spectral decay

So, it's enough to sample...



- In practice, 1% of the data is more than enough
- In theory, we can go down to $n \times \mathrm{polylog}(n)$

Important Features are Preserved

Each feature/eigenvector of K has an associated eigenvalue.

Each eigenvalue measures the *importance* of the corresponding feature for reconstructing K.

- Let $B(t)$ be an orthonormal basis for those features which have eigenvalue at least t in K.

- Let $B_\perp(t)$ be an orthonormal basis for the complement of $B(t)$.

- Similarly for $\widehat{B}(t), \widehat{B}_\perp(t)$

Theorem 2 *For every* $\xi_1 > \xi_2$

$$\left| \widehat{B}^T(\xi_1) B_\perp(\xi_2) \right|_2 \leq \frac{|K - \widehat{K}|_2}{\xi_1 - \xi_2}$$

Open Problems

- How general is this “stability under noise”?
- For example, does it hold for

  Support Vector Machines?
- When can we prove such stability in a black-box fashion, i.e. as with matrices?
- Can we exploit if for data privacy?