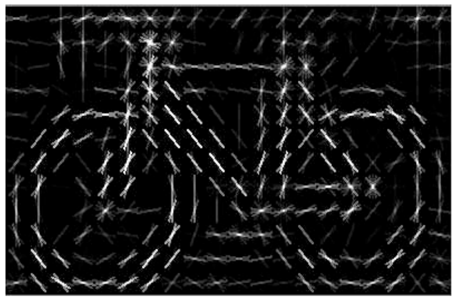
ECCV 2012 Tutorial

Part 2
Explicit Embeddings I
Kernel Feature Maps

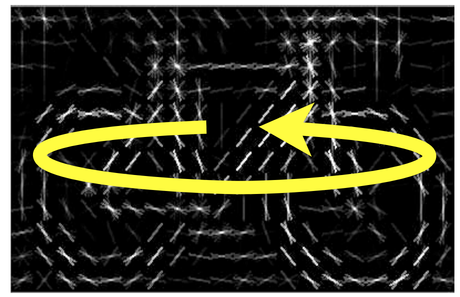
Andrea Vedaldi
University of Oxford

modelling of structure

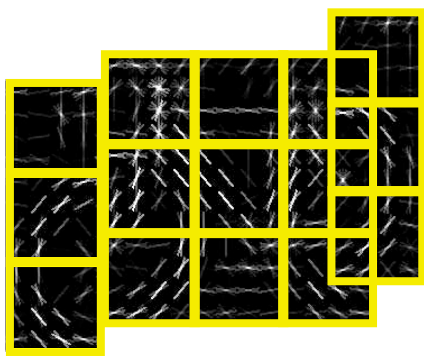
location



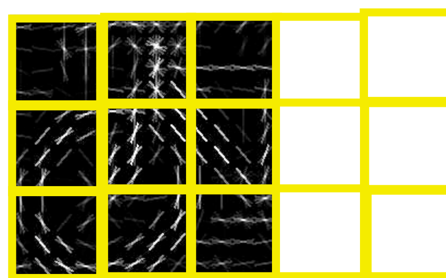
direction



deformation



truncation



[Vedaldi Zisserman 09]

massive datasets

Challenge

1000 classes

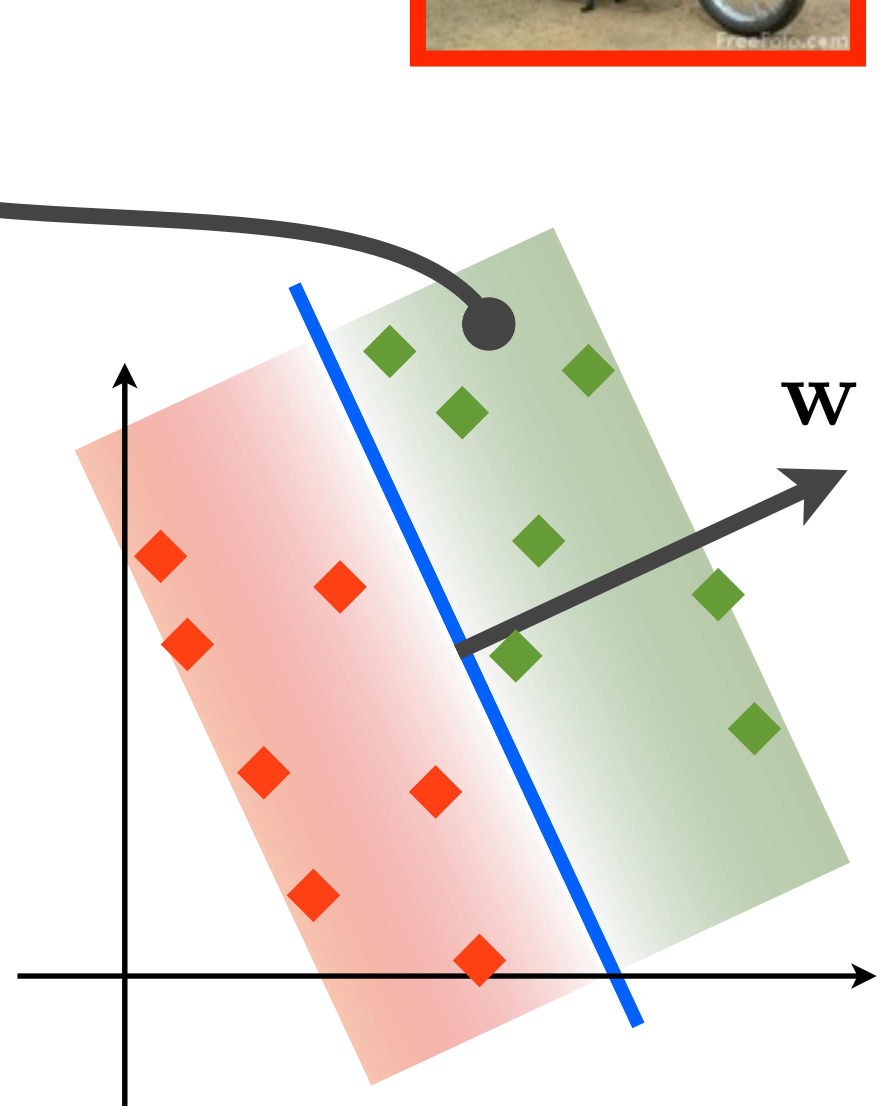
1.5M images

>50k-dim. descriptors

Very efficient learning

bicycle?

$\mathbf{x}$



binary classifier

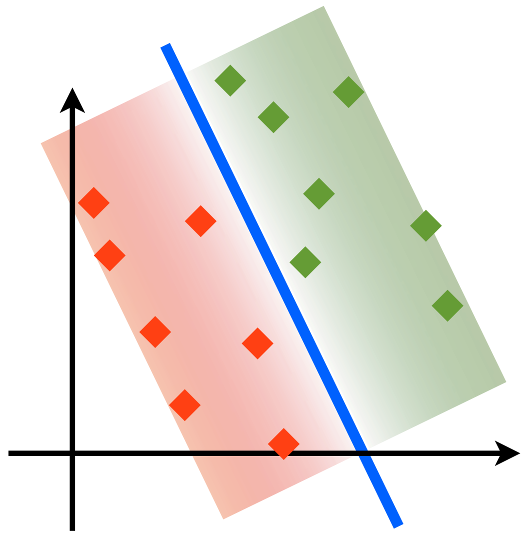
$$F(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

Linear vs non-linear kernels

Linear SVM

✔ fast

✘ restrictive

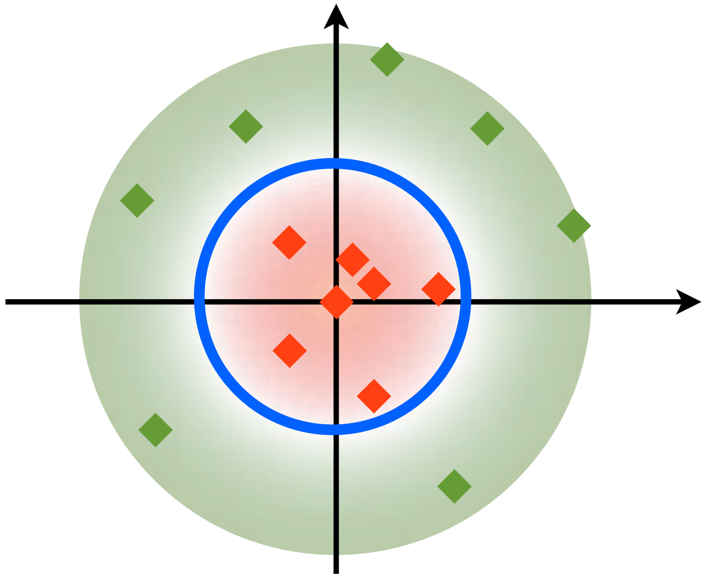


$$F(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

Non-linear SVM

✘ much slower

✔ powerful



$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

Non-linear kernel classifiers are slow

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

thousand bicycles

many more non-bicycle

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

feature map

$$K(\mathbf{x}, \mathbf{x}_i) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}_i) \rangle$$

$$F(\mathbf{x}) = \langle \mathbf{w}, \Psi(\mathbf{x}) \rangle \qquad \mathbf{w} = \sum_{i=1}^{N} \alpha_i \Psi(\mathbf{x}_i)$$

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

approximated feature map

$$K(\mathbf{x}, \mathbf{x}_i) \approx \langle \widehat{\Psi}(\mathbf{x}), \widehat{\Psi}(\mathbf{x}_i) \rangle$$

$$F(\mathbf{x}) = \langle \mathbf{w}, \widehat{\Psi}(\mathbf{x}) \rangle \qquad \mathbf{w} = \sum_{i=1}^{N} \alpha_i \widehat{\Psi}(\mathbf{x}_i)$$

Example: *homogeneous kernel map*

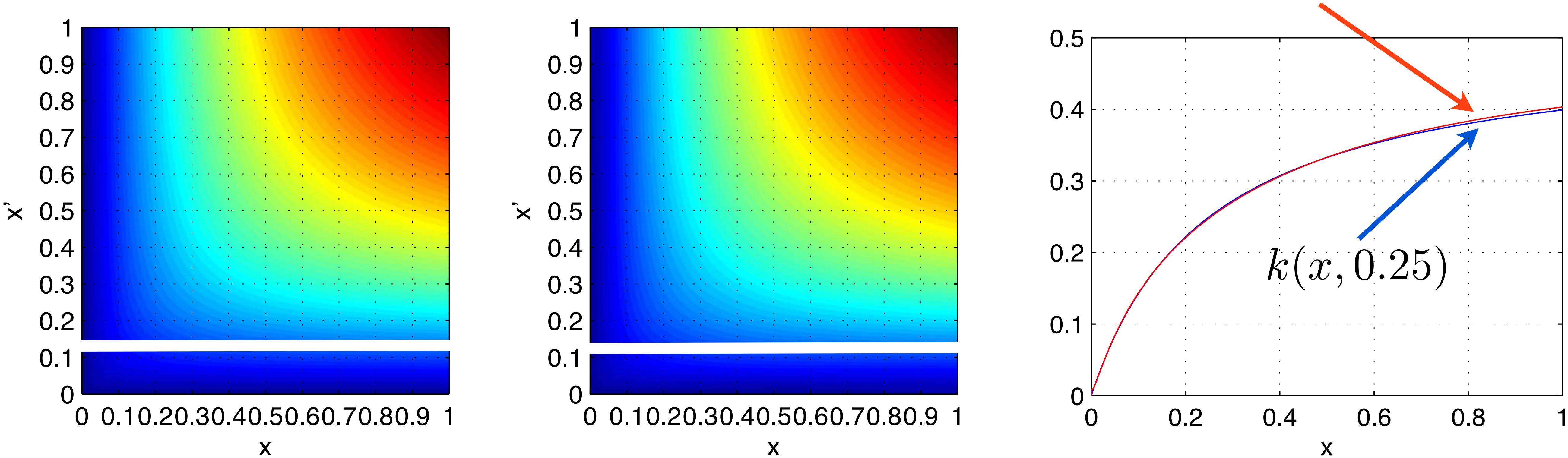
χ^2 kernel
Excellent for bag-of-words, ...

$$k(x,x') = \frac{2xx'}{x+x'}$$

Homogeneous kernel map
Closed form, simple, small

$$\Phi(x) = \sqrt{x}\begin{bmatrix} 0.8 \\ 0.6\cos(0.6\ \log x) \\ 0.6\sin(0.6\ \log x) \end{bmatrix}$$

$k(x,x')$ $\quad\langle \Phi(x), \Phi(x') \rangle$ $\quad\langle \Phi(x), \Phi(0.25) \rangle$



[Vedaldi Zisserman 10,11]

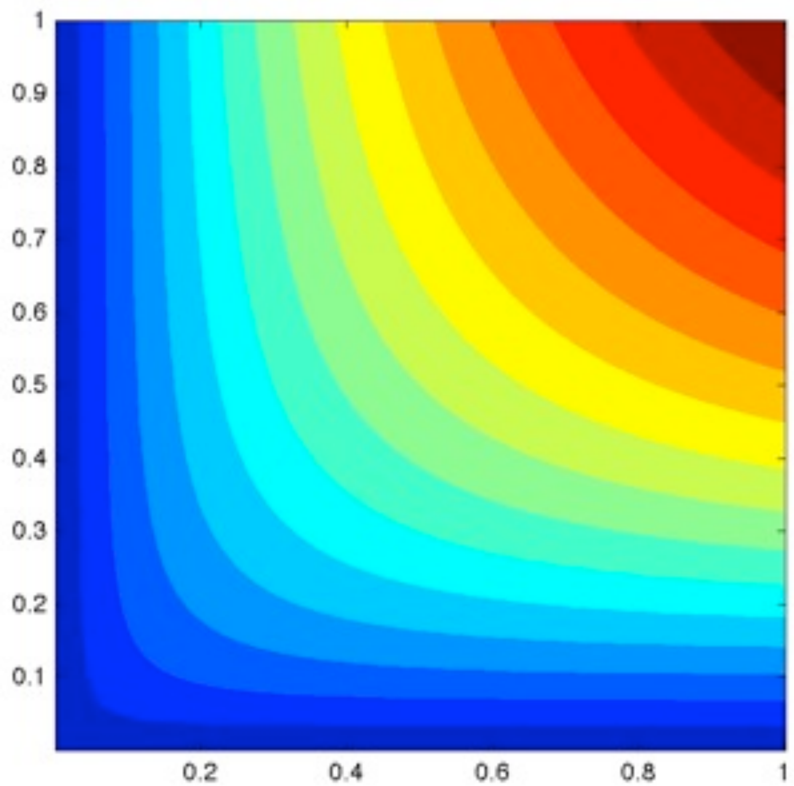
MATLAB code for Chi2 kernel

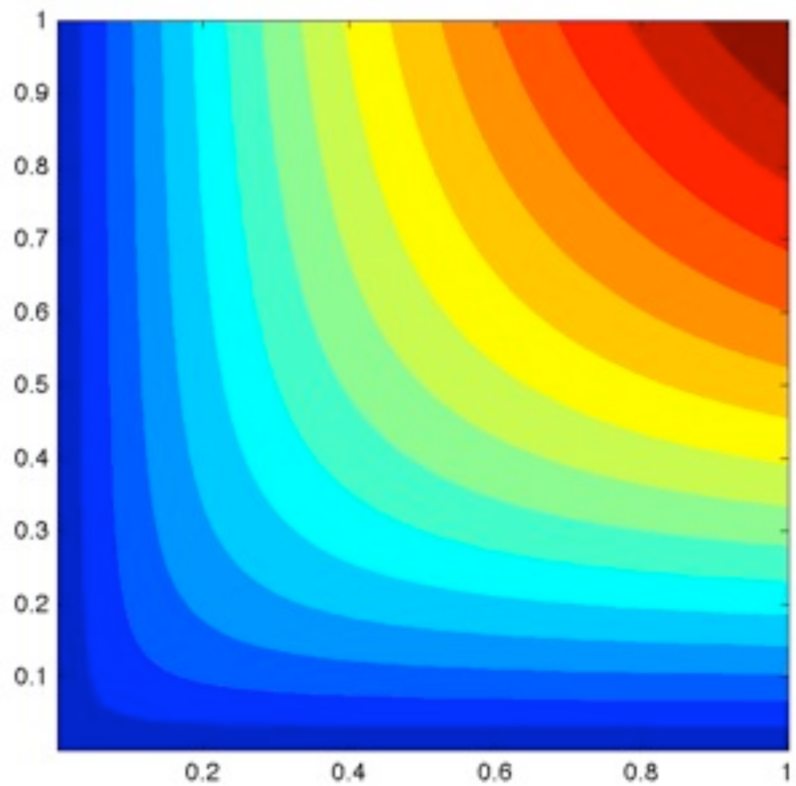
```
x = .01:.01:1 ;
for i = 1:100
  for j = 1:100
    K(i,j) = ...
      2*x(i)*x(j)/(x(i)+x(j));
  end
end
```

With the hom. kernel feature map

```
x = .01:.01:1 ;
psi = vl_homkermap(x,1) ;
K = psi'*psi ;
```

VLFeat Toolbox
http://www.vlfeat.org





Example: *homogeneous kernel map*

training time

Caltech-101 category recognition

#1,500

1 h 5 m

4× speedup

DaimlerChrylser pedestrian recognition

#20,000

1/2 h 14 s

100× speedup

Trecvid 2009 video indexing

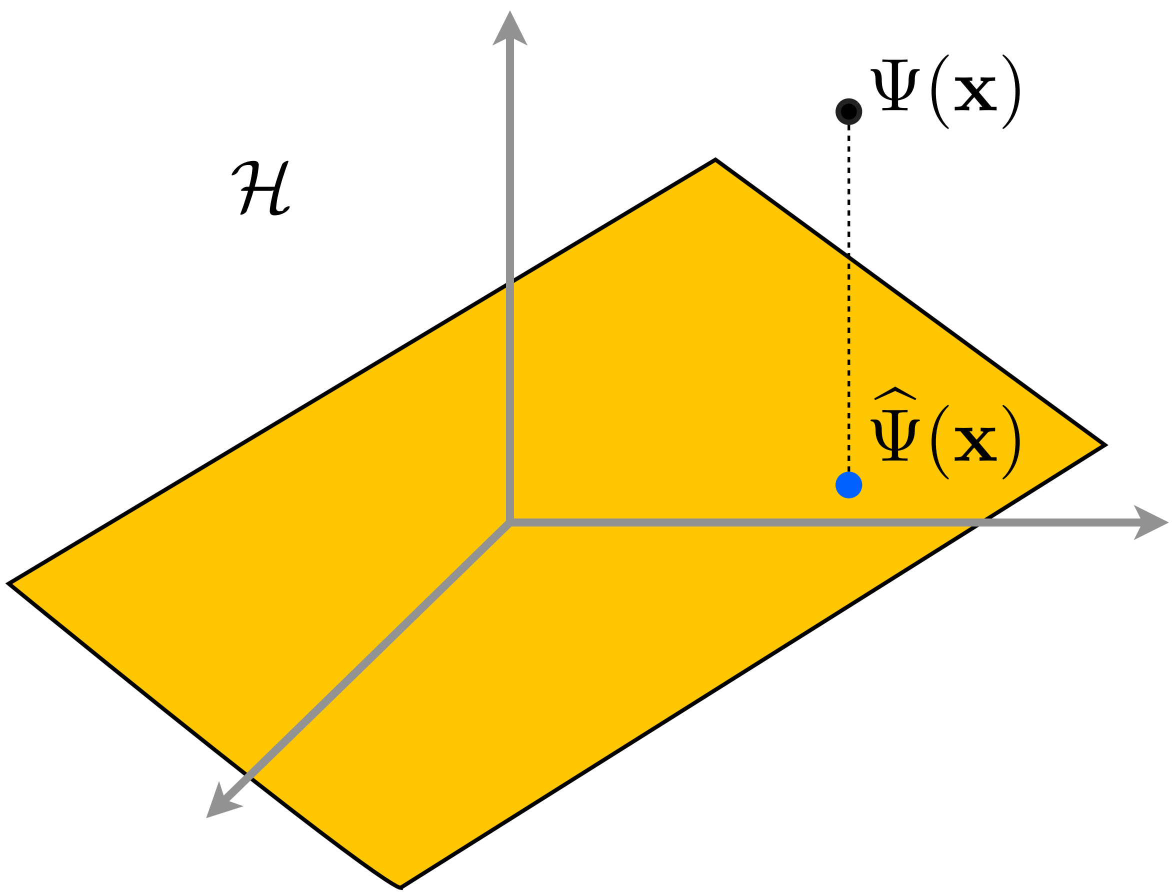
#70,000

> 1 h 22.6 s

160× speedup

Finite-dimensional embeddings by projection

Exact feature		**Approximated feature**
$\Psi(\mathbf{x})$	projection →	$\widehat{\Psi}(\mathbf{x})$
exact but inf. dim.		approx. but compact: small + dense or sparse



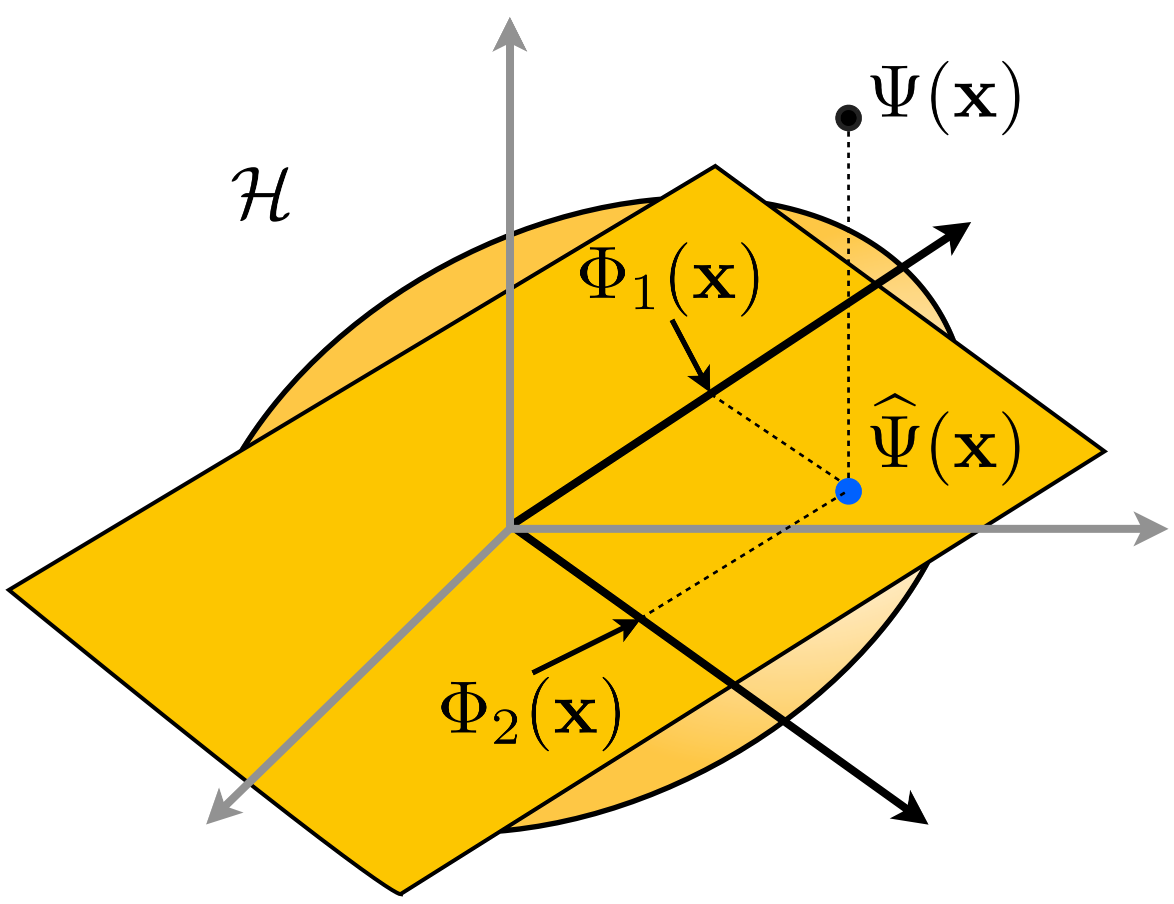
Exact feature space

(reproducing kernel Hilbert space)

$$K(\mathbf{x}, \mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle_{\mathcal{H}}$$

Data distribution

$p(\mathbf{x})$



PCA in feature space

Top D eigenfunctions of the kernel

$$\int_{\mathcal{X}} K(\mathbf{x}, \mathbf{z}) u_i(\mathbf{z}) p(\mathbf{z})\, d\mathbf{z} = \kappa_i^2 u_i(\mathbf{x})$$

Coordinate functions

Projection on orthonormal PCA basis

$$\Phi_i(\mathbf{x}) = \kappa_i u_i(\mathbf{x})$$

Approximate feature

$$\widehat{\Psi}(\mathbf{x}) \cong \Phi(\mathbf{x}) = \begin{bmatrix} \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_D(\mathbf{x}) \end{bmatrix}$$

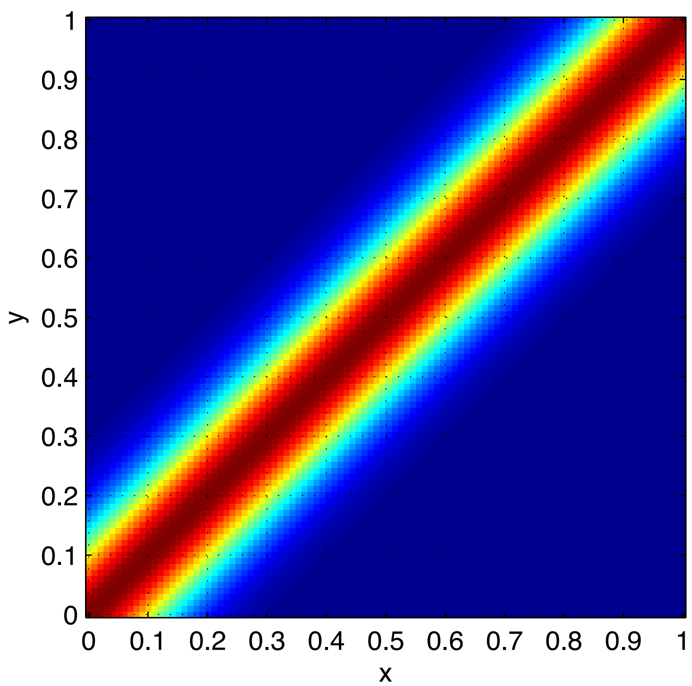
[Williams Seeger 01]

Nyström: stationary kernels

Stationary kernel

Translation invariant

$$\forall \mathbf{t} \in \mathbb{R}^d : K(\mathbf{x} + \mathbf{t}, \mathbf{x}' + \mathbf{t}) = K(\mathbf{x}, \mathbf{x}')$$

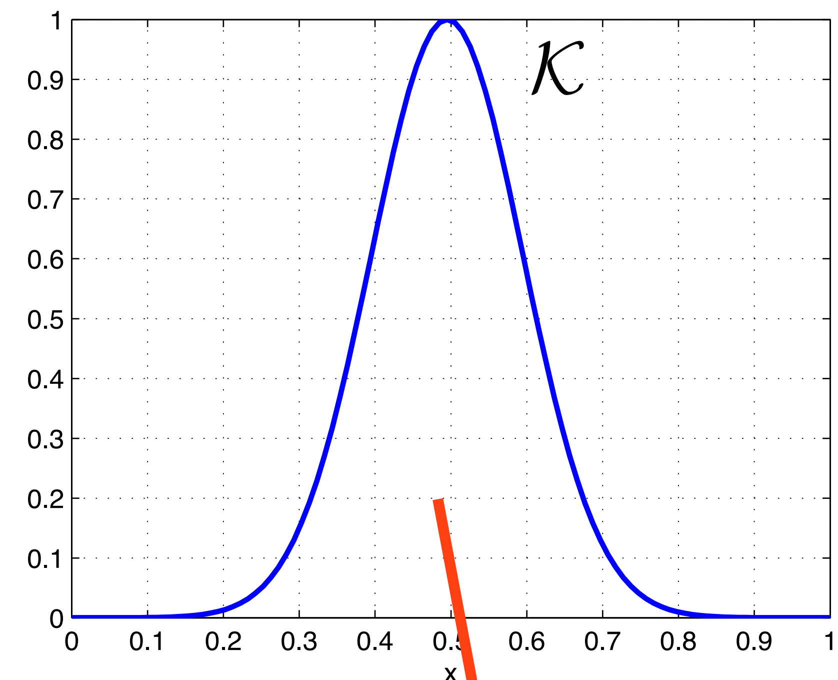


$$K(x, x')$$

Signature / profile

1D positive definite function

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{K}(\mathbf{x}' - \mathbf{x})$$



Fourier transform

Eigenfunctions = Sinusoids

$$\int_{\mathbb{R}^d} \mathcal{K}(\mathbf{x} - \mathbf{z}) e^{-\mathbf{i}\langle \boldsymbol{\omega}, \mathbf{z} \rangle} \, d\mathbf{z} = \kappa_{\boldsymbol{\omega}}^2 e^{-\mathbf{i}\langle \boldsymbol{\omega}, \mathbf{x} \rangle}$$

$$\boxed{\Phi_{\boldsymbol{\omega}}(\mathbf{x}) = \kappa_{\boldsymbol{\omega}} e^{-\mathbf{i}\langle \boldsymbol{\omega}, \mathbf{x} \rangle}}$$

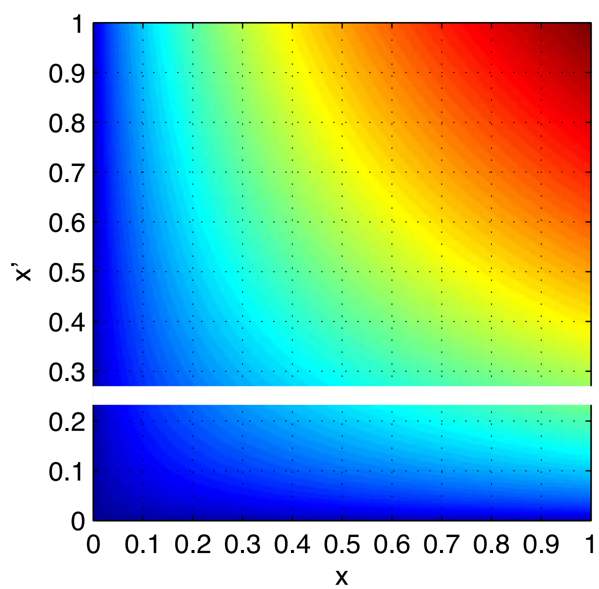
Additive *homogeneous* kernels

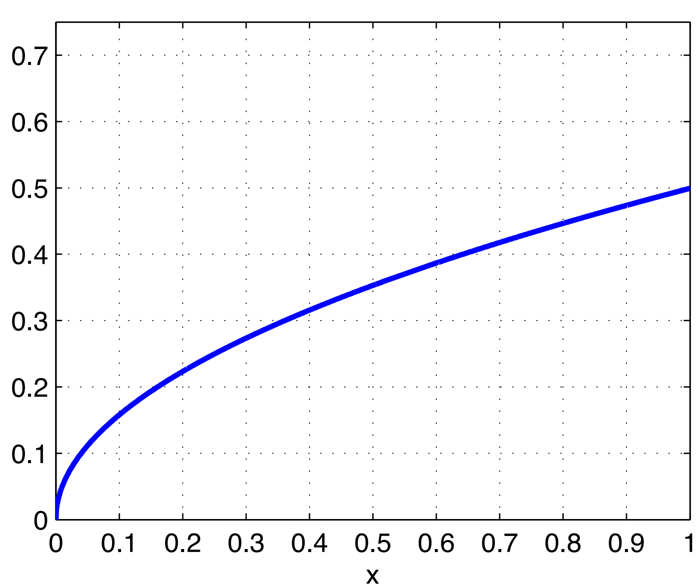
Additive kernel
Sum of 1D kernels

$$K(\mathbf{x}, \mathbf{x}') = \sum_{l=1}^{D} k(x_l, x'_l)$$

Hellinger

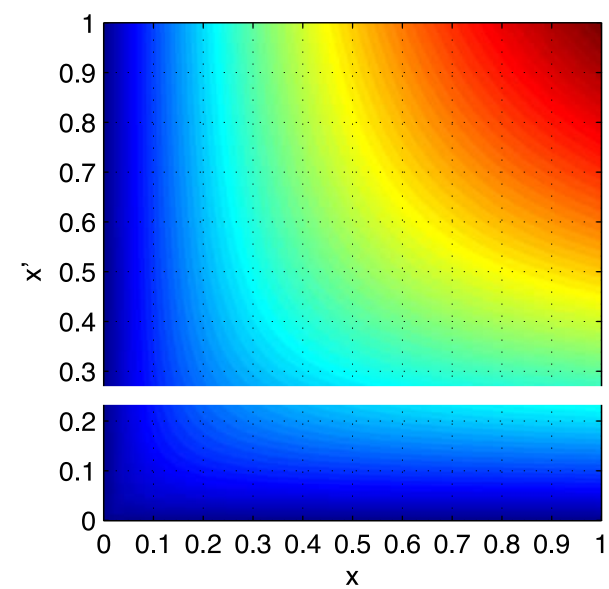
$$k(x, x') = \sqrt{xx'}$$

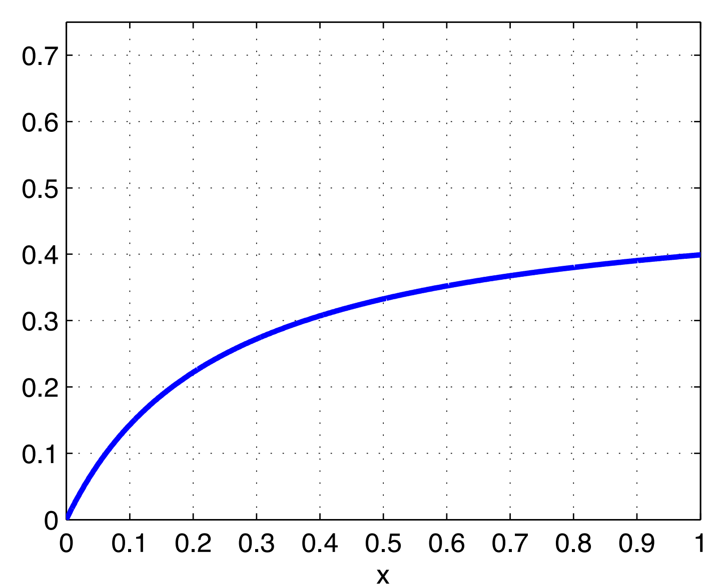




χ^2

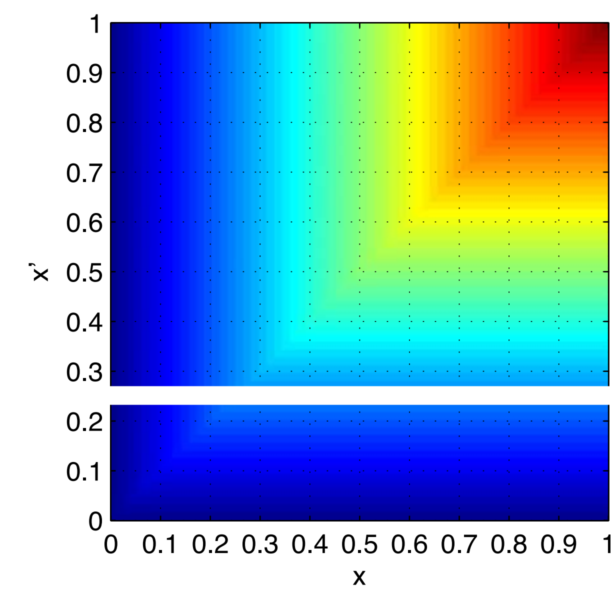
$$\frac{2xx'}{x + x'}$$

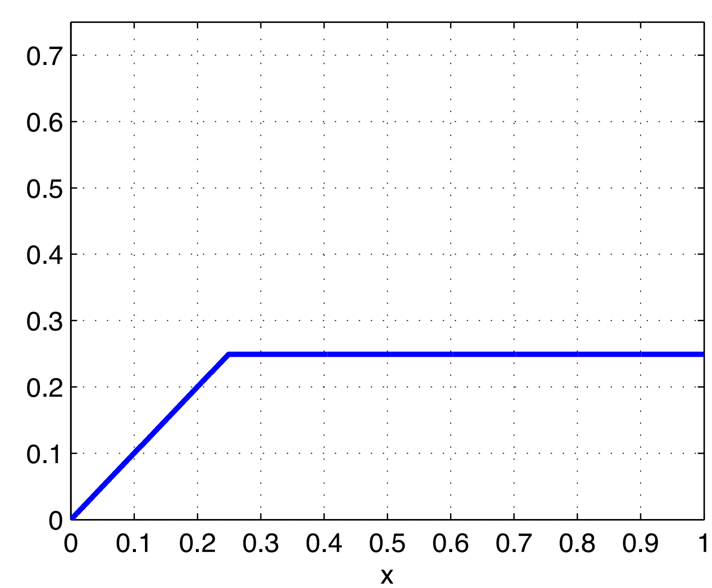




intersection

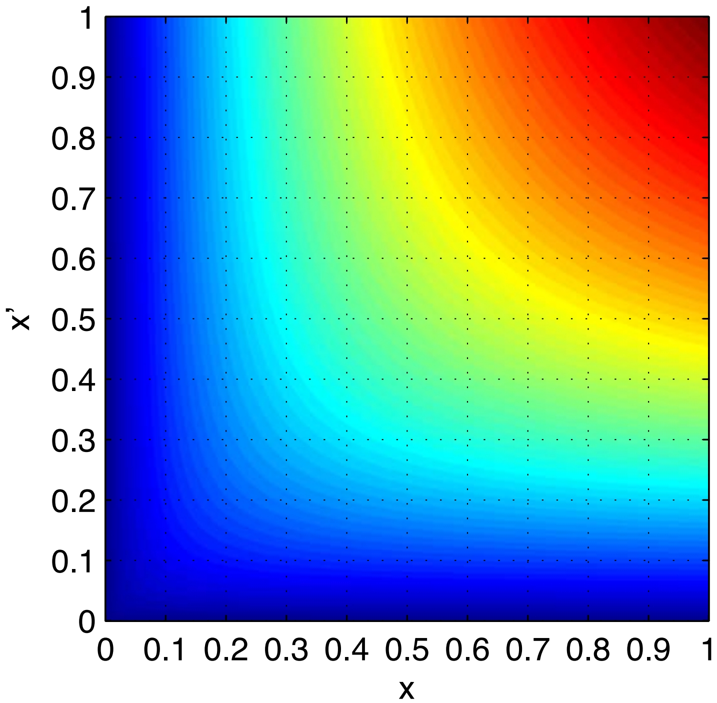
$$\min\{x, x'\}$$



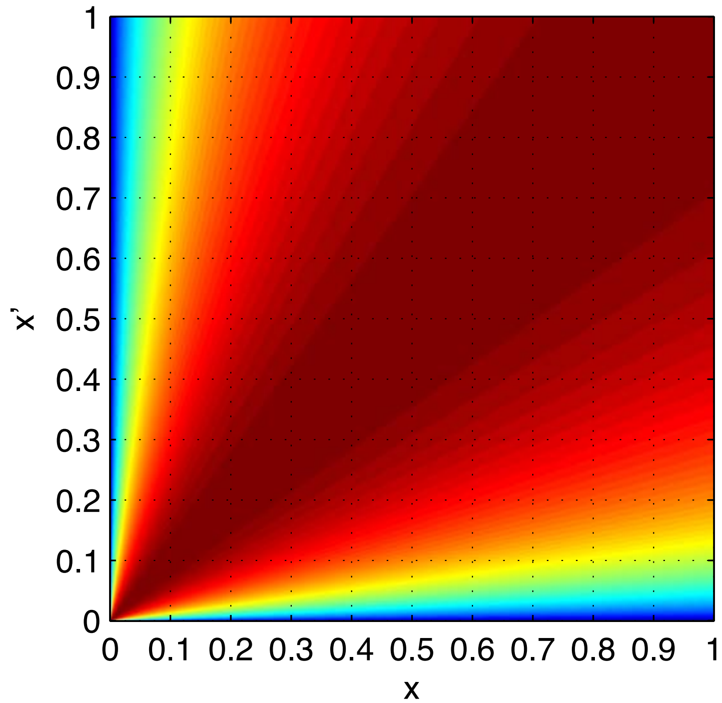


The trick

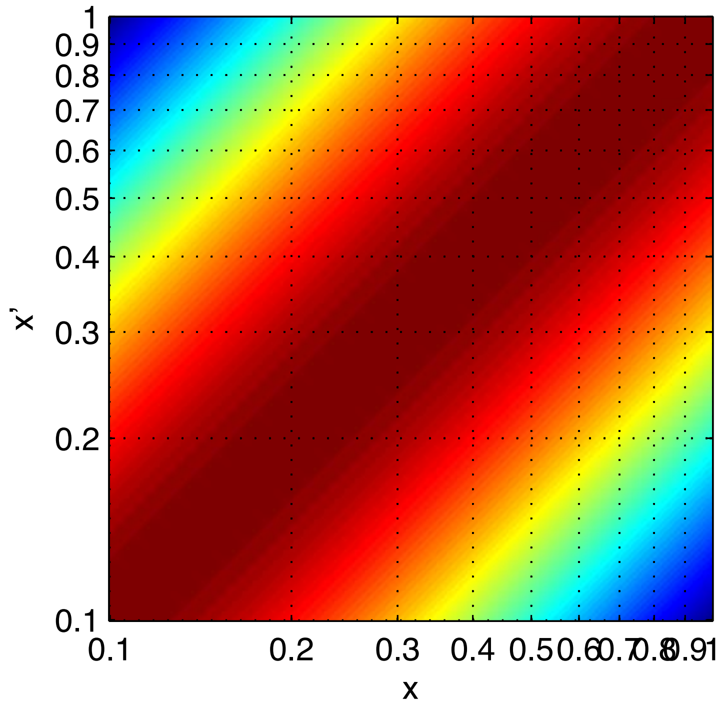
$$k(x, x')$$



$$\frac{k(x, x')}{\sqrt{xx'}}$$



$$\log x$$



Homogeneous kernel

Multiplicative constant pops out

$$\forall c \geq 0 : k(cx, cx') = ck(x, x')$$

Signature / profile

Up to a factor and a logarithm

$$k(x, x') = \sqrt{xx'}\, \mathcal{K}(\log x - \log x')$$

$$\boxed{\Phi_\omega(x) = \kappa_\omega \sqrt{x}\, e^{-\mathbf{i}\langle \omega, \log x \rangle}}$$

[Vedaldi Zisserman 10, 11]

Hellinger	X^2	intersection
$k(x, x') = \sqrt{xx'}$	$\dfrac{2xx'}{x + x'}$	$\min\{x, x'\}$
$\mathcal{K}(\lambda) = 1$	$e^{-\lvert\lambda\rvert/2}$	$\operatorname{sech}(\lambda/2)$
$\kappa_\omega^2 = \delta(\omega)$	$\dfrac{2}{\pi(1 + 4\omega^2)}$	$\operatorname{sech}(\pi\omega)$
$\Phi_\omega(x) = \sqrt{x}$	$\sqrt{\dfrac{2x}{\pi(1 + 4\omega^2)}} e^{-\mathbf{i}\,\omega \log x}$	$\sqrt{x \operatorname{sech}(\pi\omega)} e^{-\mathbf{i}\,\omega \log x}$

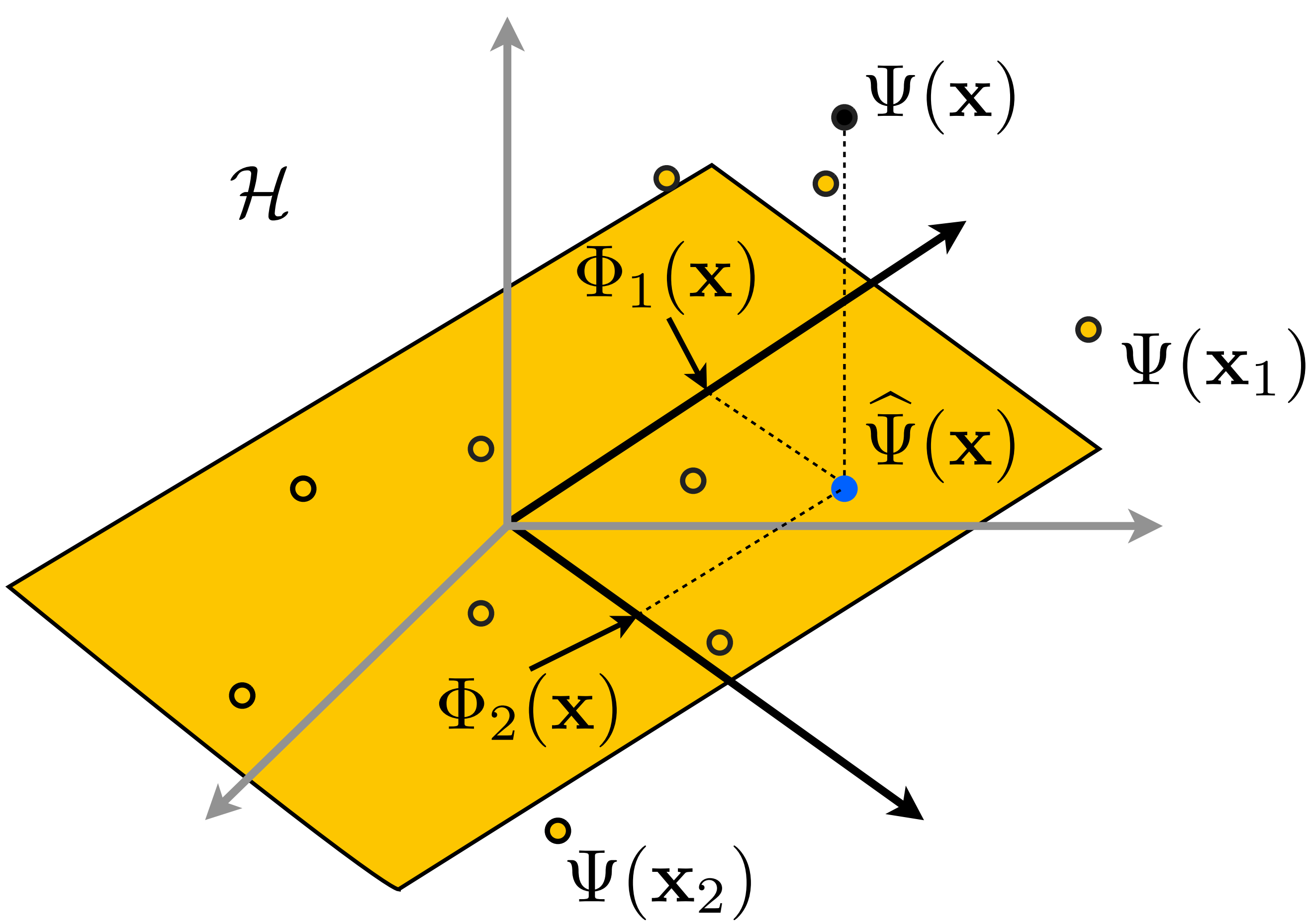
Empirical Nyström's approximation

Exact feature space
(e.g. reproducing kernel Hilbert space)

$$K(\mathbf{x}, \mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle_{\mathcal{H}}$$

Data distribution

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$



PCA in feature space
Find the top D eigenfunctions of the kernel

$$\frac{1}{n}\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j) u_i(\mathbf{x}_j) = \kappa_i^2 u_i(\mathbf{x}_i)$$

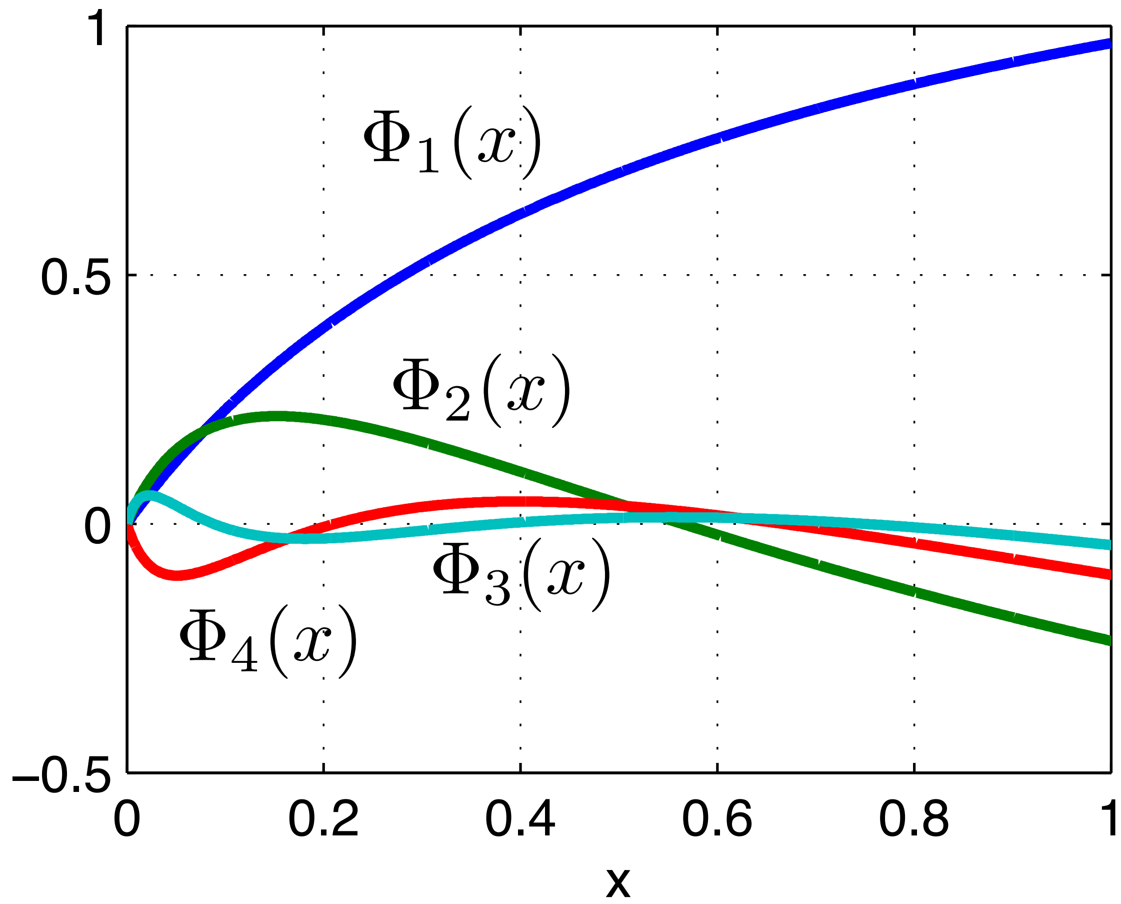
Coordinate functions
Projection on orthonormal PCA basis

$$\Phi_i(\mathbf{x}) = \kappa_i u_i(\mathbf{x})$$

$$= \underbrace{(n\kappa_i)^{-1} \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_j) u_i(\mathbf{x}_j)}_{\text{encoding a new point } \mathbf{x} \text{ requires projecting it on all the data sample}}$$

encoding a new point **x**
requires projecting it
on all the data sample

Additive kernel PCA (addKPCA)



$$\Phi_i(x) = (n\kappa_i)^{-1} \sum_{i=1}^{n} k(x, x_j) u_i(x_j)$$

[Perronnin *et al.* 10]

- For any additive kernel
  - Empirical Nyström's approximation can be applied component-wise
  - The coordinate functions **can be tabulated**

Beyond additive kernels

- RBF / Gaussian / exponential kernel
  Random Fourier features [Rahimi Recht 07]

$$Q(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}'\|^2\right)$$

$$\Phi_{\mathrm{RBF}}(\mathbf{x}) = D^{-\frac{1}{2}}\left[\cos\langle\omega_1,\mathbf{x}\rangle \quad \ldots \quad \cos\langle\omega_D,\mathbf{x}\rangle \quad \sin\langle\omega_1,\mathbf{x}\rangle \quad \ldots\right]^\top$$

random Gaussian vectors

- **Generalized**: RBF + Chi2 distance

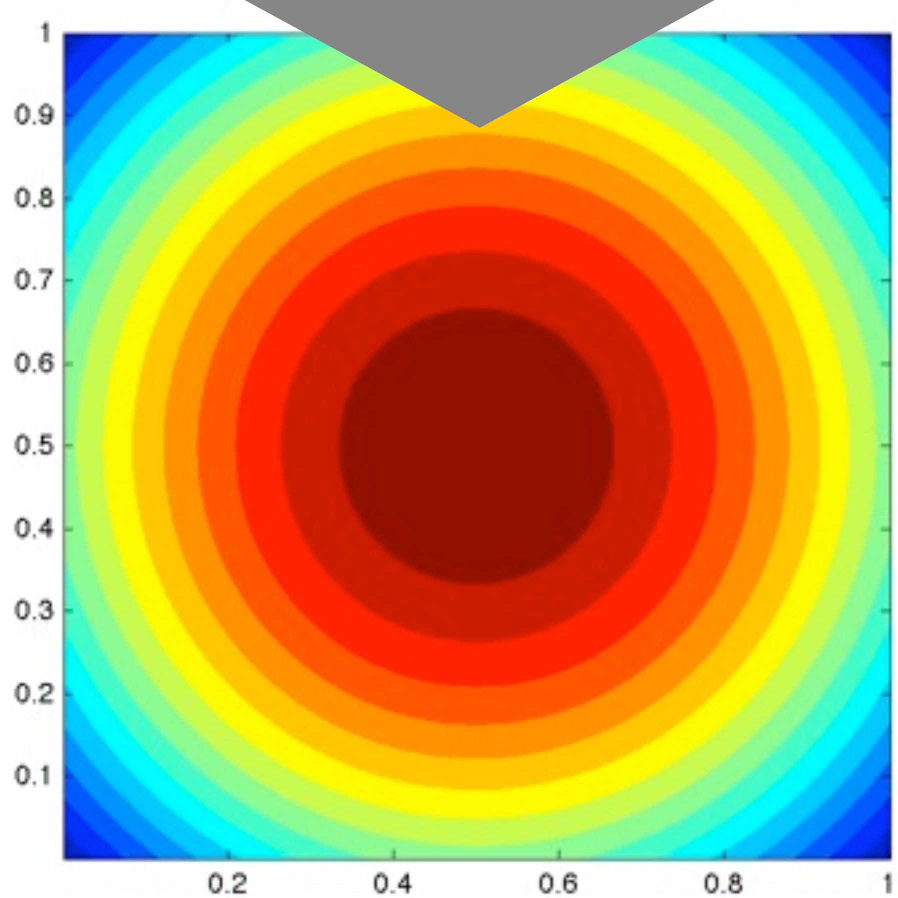
$$Q(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{D}\frac{2(x_i-x_i')^2}{x_i+x_i'}\right)$$

$$\Phi_{\mathrm{GRBF}}(\mathbf{x}) = D^{-\frac{1}{2}}\left[\cos\langle\omega_1,\Phi(\mathbf{x})\rangle \quad \ldots \quad \cos\langle\omega_D,\Phi(\mathbf{x})\rangle \quad \sin\langle\omega_1,\Phi(\mathbf{x})\rangle \quad \ldots\right]^\top$$

Example: *Random Fourier features*

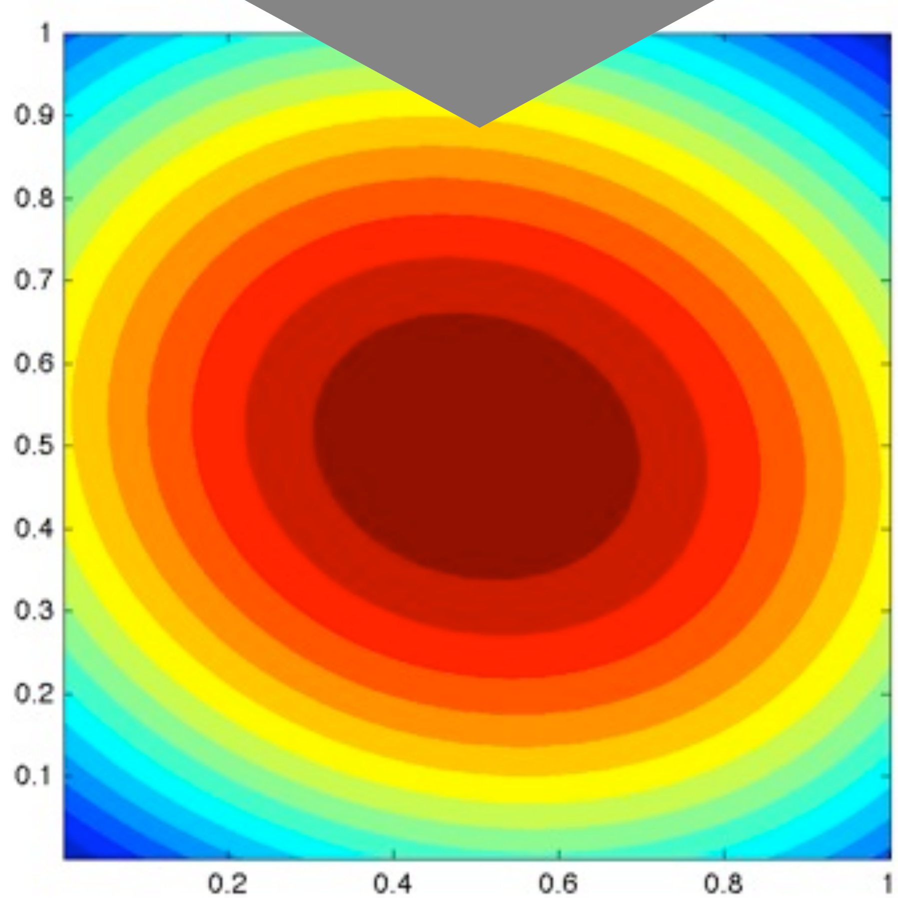
RBF kernel

```
[x1,x2] = meshgrid(range) ;
x = [x1(:) x2(:)]' ;
xp = [0.5;0.5] ;
for i=1:n*n
  sqd1 = (x(1,i) - xp(1))^2 ;

  sqd2 = (x(2,i) - xp(2))^2 ;

  K(i) = exp(-0.5*(sqd1 + sqd2)) ;
end
```



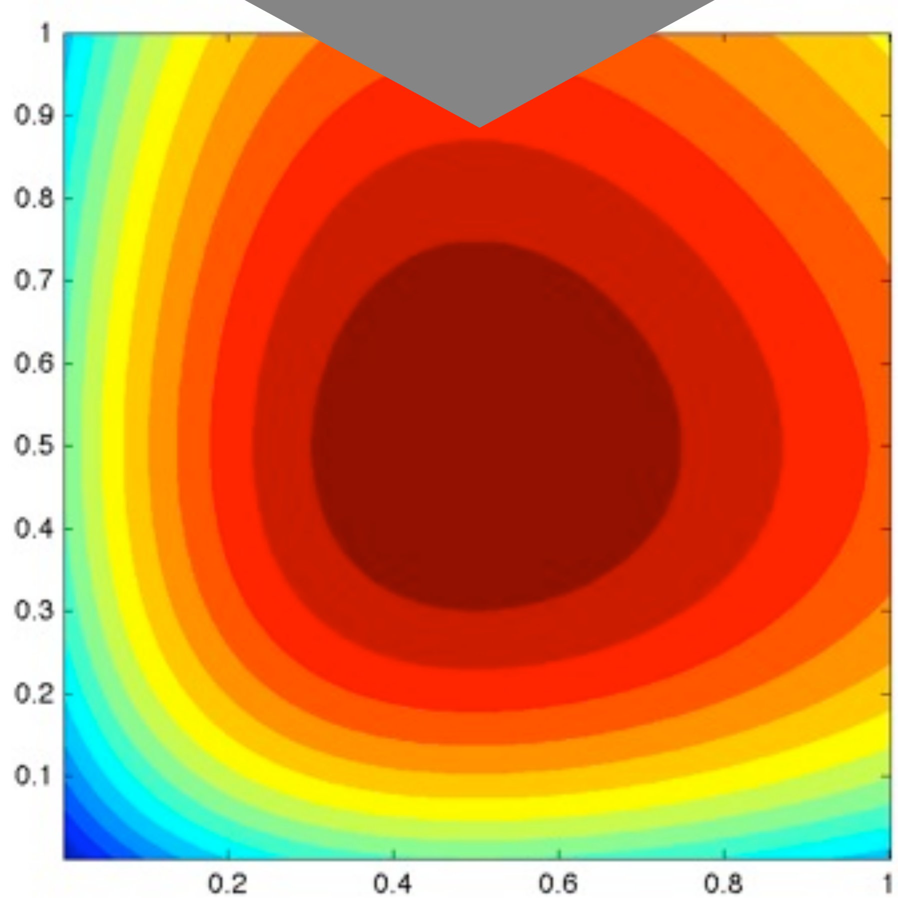
RBF w/Random Fourier Features

```
omega = randn(10,2) ;


psi = [cos(omega*x) ;
       sin(omega*x)] / sqrt(10) ;
psip = [cos(omega*xp) ;
        sin(omega*xp)] / sqrt(10) ;
K = psi'*psip ;
```



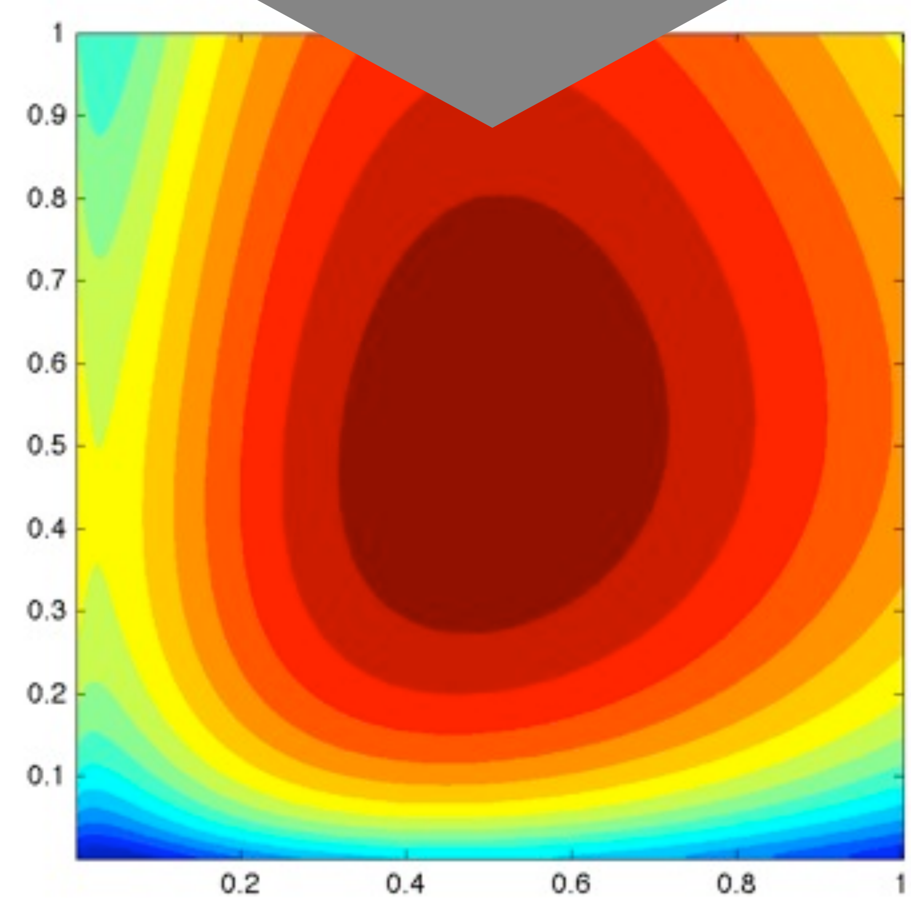
Chi2-RBF kernel

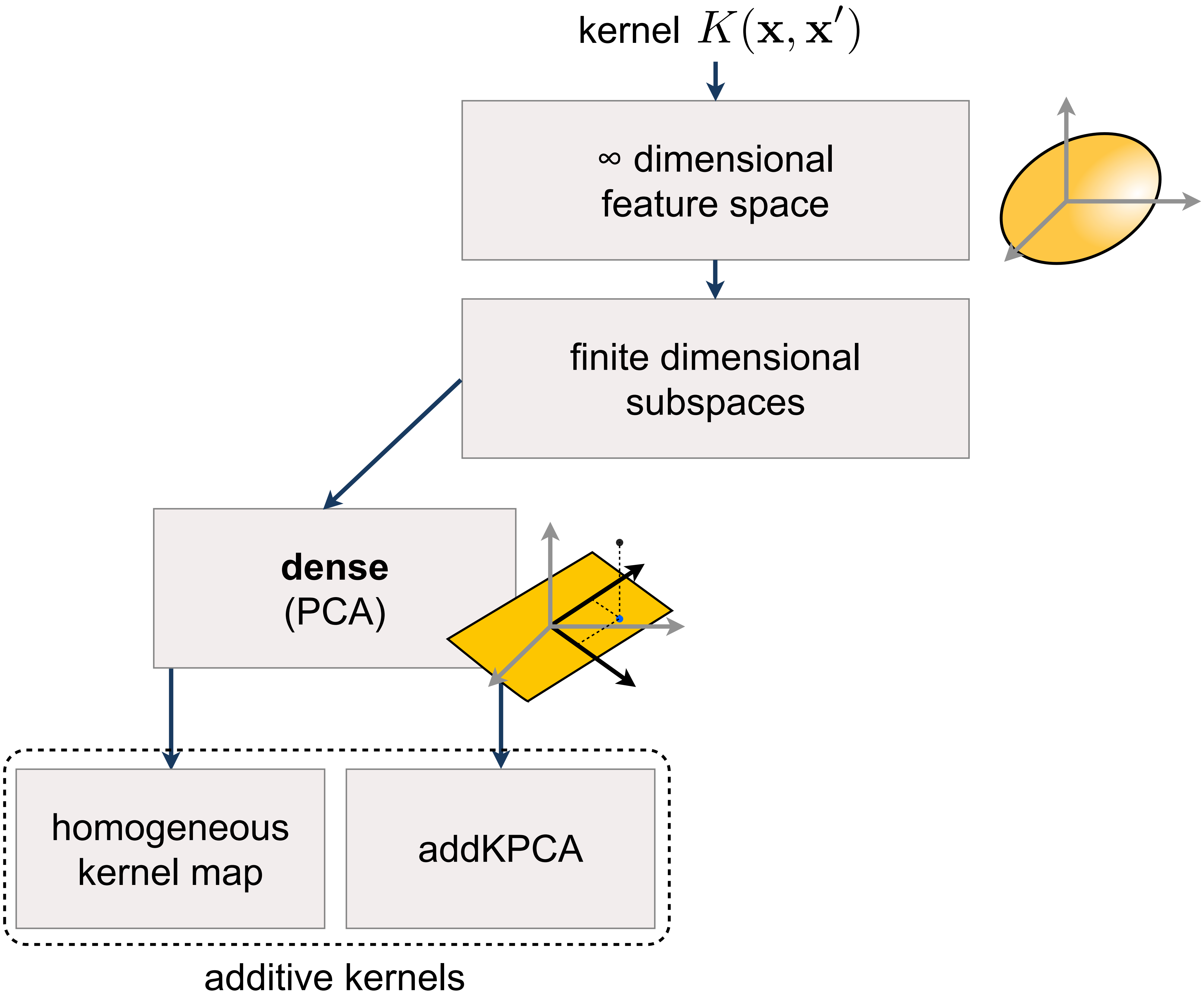
```
[x1,x2] = meshgrid(range) ;
x = [x1(:) x2(:)]' ;
xp = [0.5;0.5] ;
for i=1:n*n
  sqd1 = (x(1,i) - xp(1))^2 /
(x(1,i)+xp(1)) ;
  sqd2 = (x(2,i) - xp(2))^2 /
(x(2,i)+xp(2)) ;
  K(i) = exp(-0.5*(sqd1 + sqd2)) ;
end
```



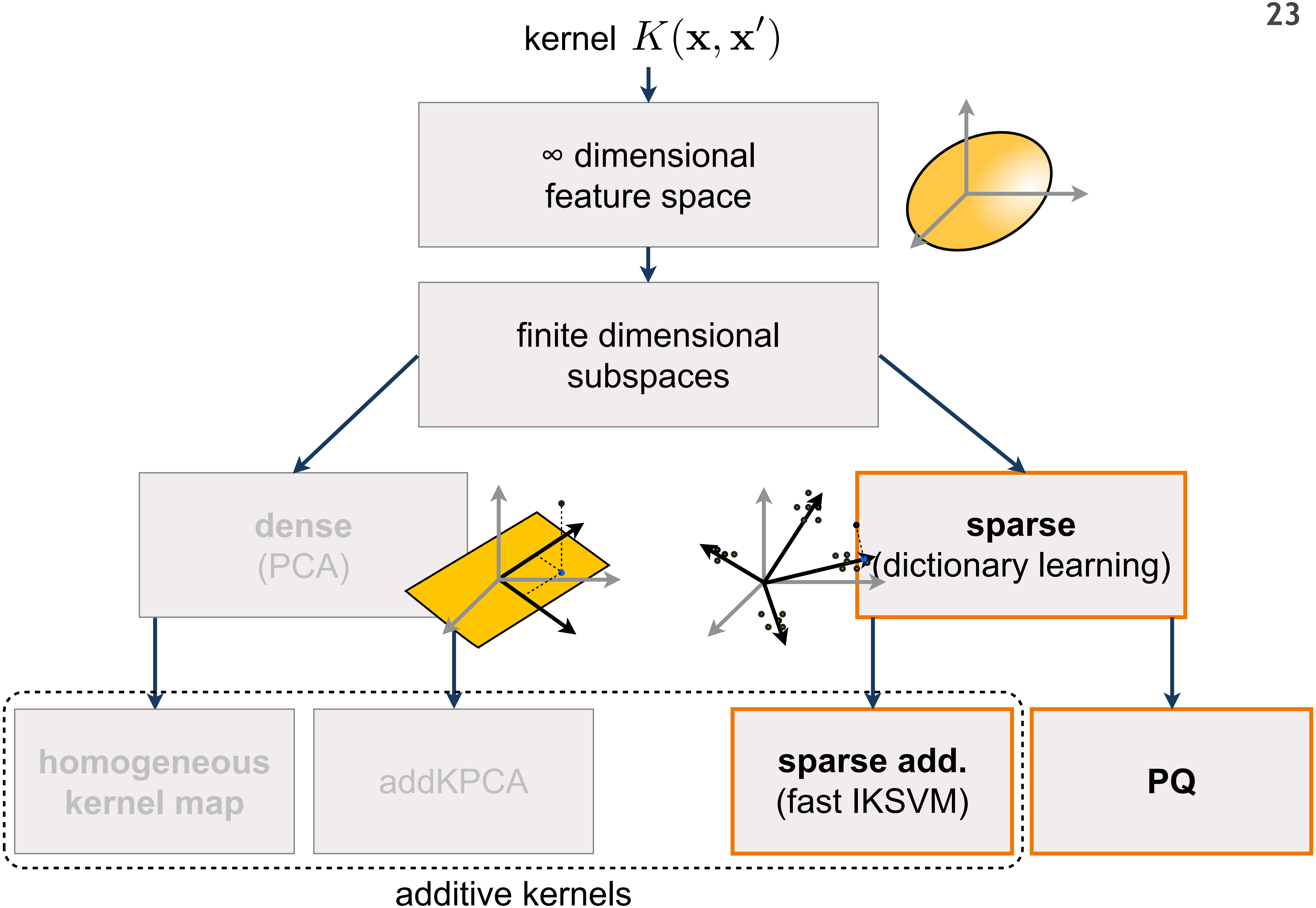
Chi2-RBF w/Random Fourier Features

```
omega = randn(10,6) ;
x = vl_homkermap(x,1) ;
xp = vl_homkermap(xp,1) ;
psi = [cos(omega*x) ;
       sin(omega*x)] / sqrt(10) ;
psip = [cos(omega*xp) ;
        sin(omega*xp)] / sqrt(10) ;
K = psi'*psip ;
```





kernel $K(\mathbf{x}, \mathbf{x}')$
∞ dimensional feature space
finite dimensional subspaces
dense
(PCA)
homogeneous kernel map
addKPCA
additive kernels



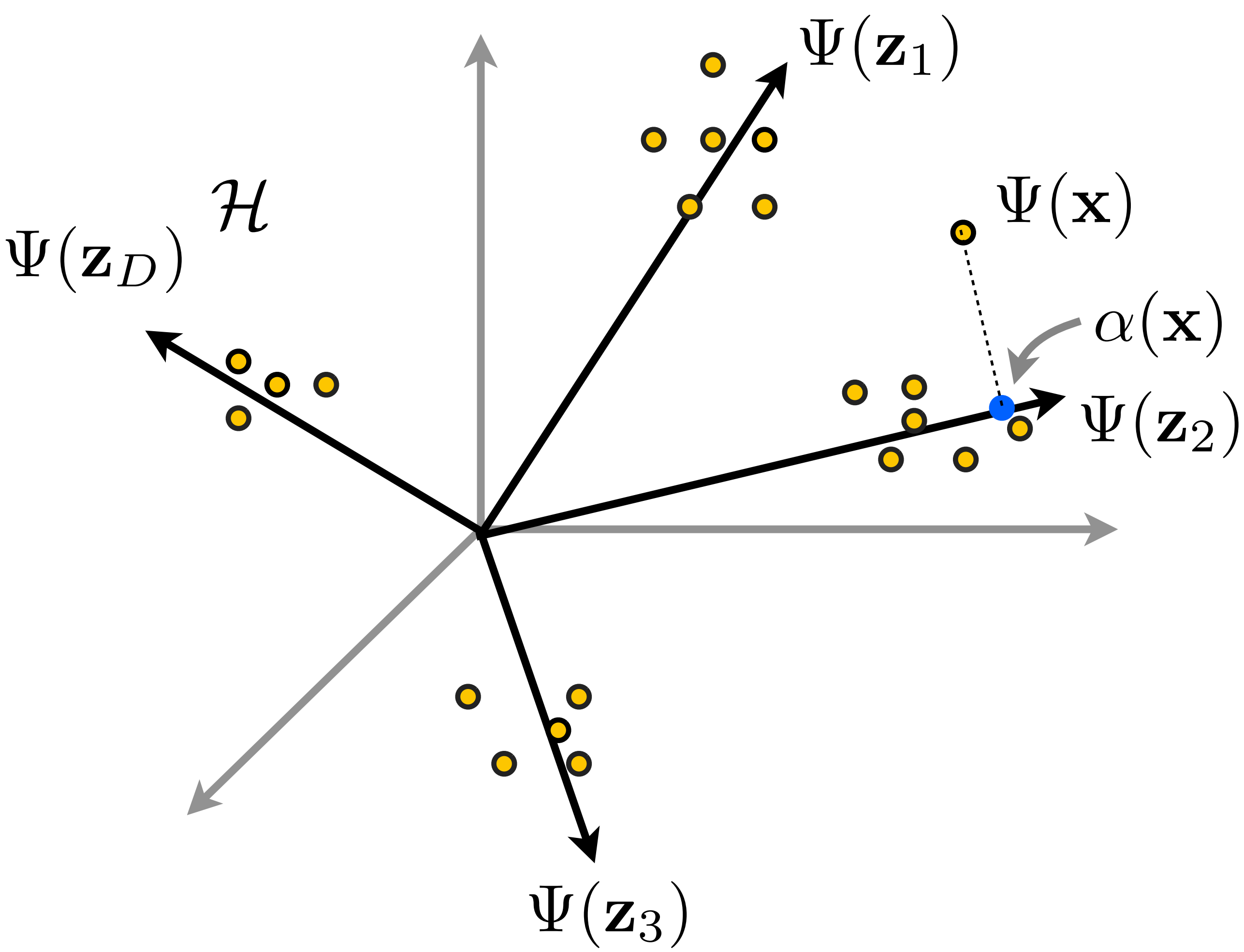
kernel $K(\mathbf{x}, \mathbf{x}')$
∞ dimensional feature space
finite dimensional subspaces
dense (PCA)
sparse (dictionary learning)
homogeneous kernel map
addKPCA
sparse add. (fast IKSVM)
PQ
additive kernels

Exact feature space
(e.g. reproducing kernel Hilbert space)

$$K(\mathbf{x}, \mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle_{\mathcal{H}}$$

Data distribution

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$



Sparse projection in feature space

large basis

$$Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_D\}$$

Find best P-sparse approx. to **x**

$$\Psi(\mathbf{x}) \approx \sum_{i=1}^{D} \Psi(\mathbf{z}_i) \Phi_i(\mathbf{x}) \qquad \Phi(\mathbf{x}) = \begin{bmatrix} 0 \\ \alpha(\mathbf{x}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

✔ Arbitrarily good *P*-sparse approximation

- **non-diagonal** inner product

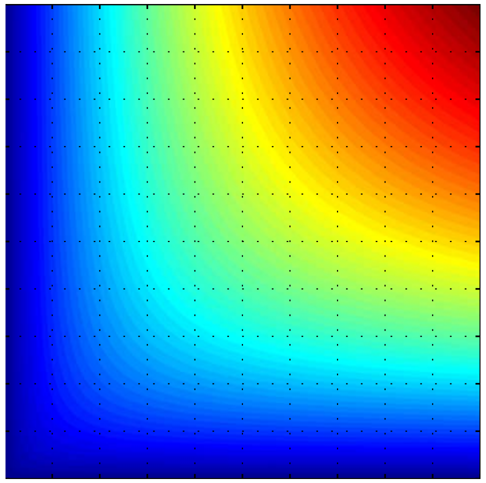
$$K(\mathbf{x}, \mathbf{x}') \approx \Phi(\mathbf{x})^\top K_{ZZ} \Phi(\mathbf{x}')$$

~~[*Kernel Matching Pursuit*, Vincente Bengion 02]~~

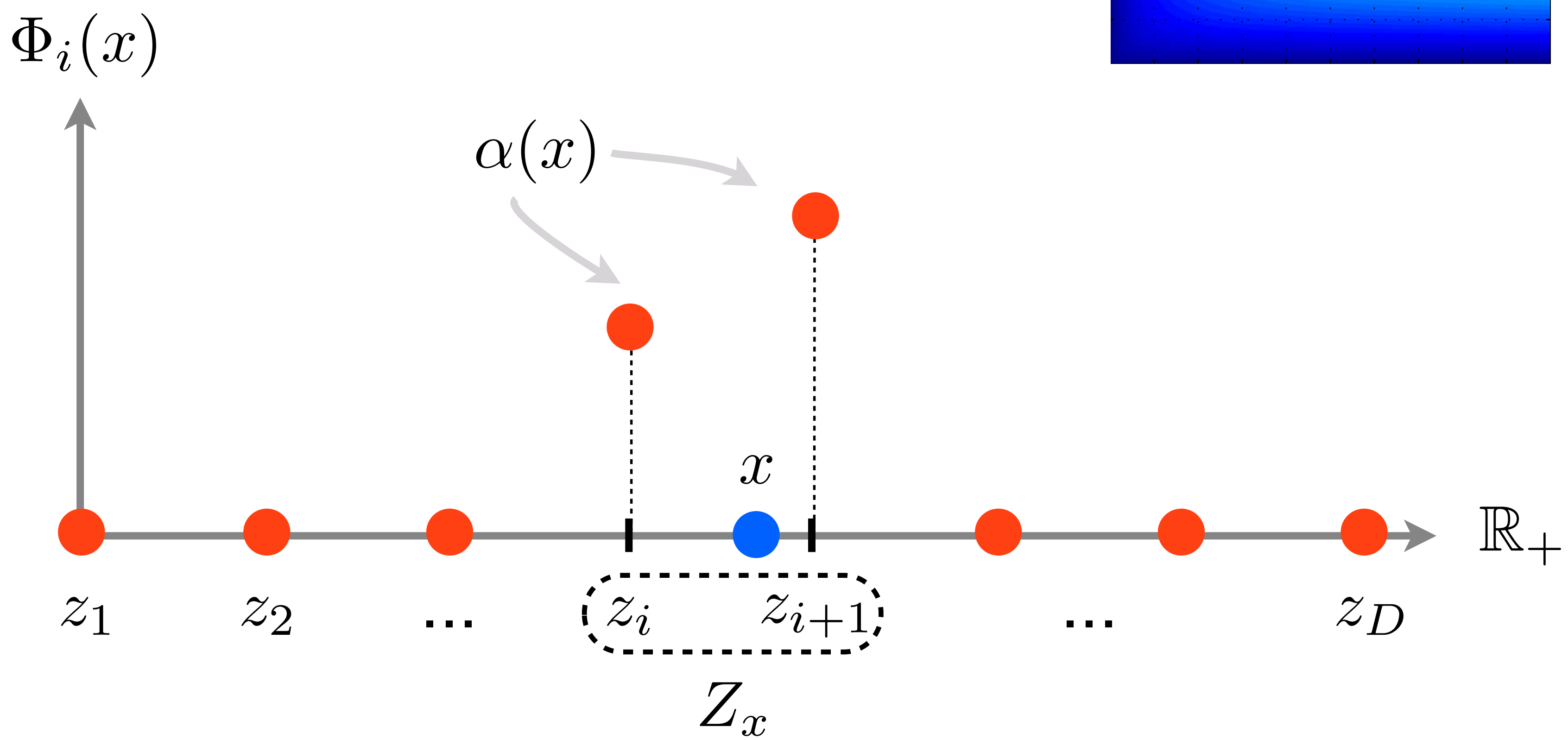
[Snelson Ghaharamani 07]
[Vedaldi Zisserman 12]

Sparse kernel maps: additive kernels

$$k(x,x') \approx \Phi(x)^\top K_{ZZ}\Phi(x') = \begin{bmatrix}1 & 0 & 0 & \dots & 0\end{bmatrix}$$



$$\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix}$$



Fast Intersection Kernel

$$\alpha(x) = \begin{bmatrix} i+1-x \\ x-1 \end{bmatrix}$$

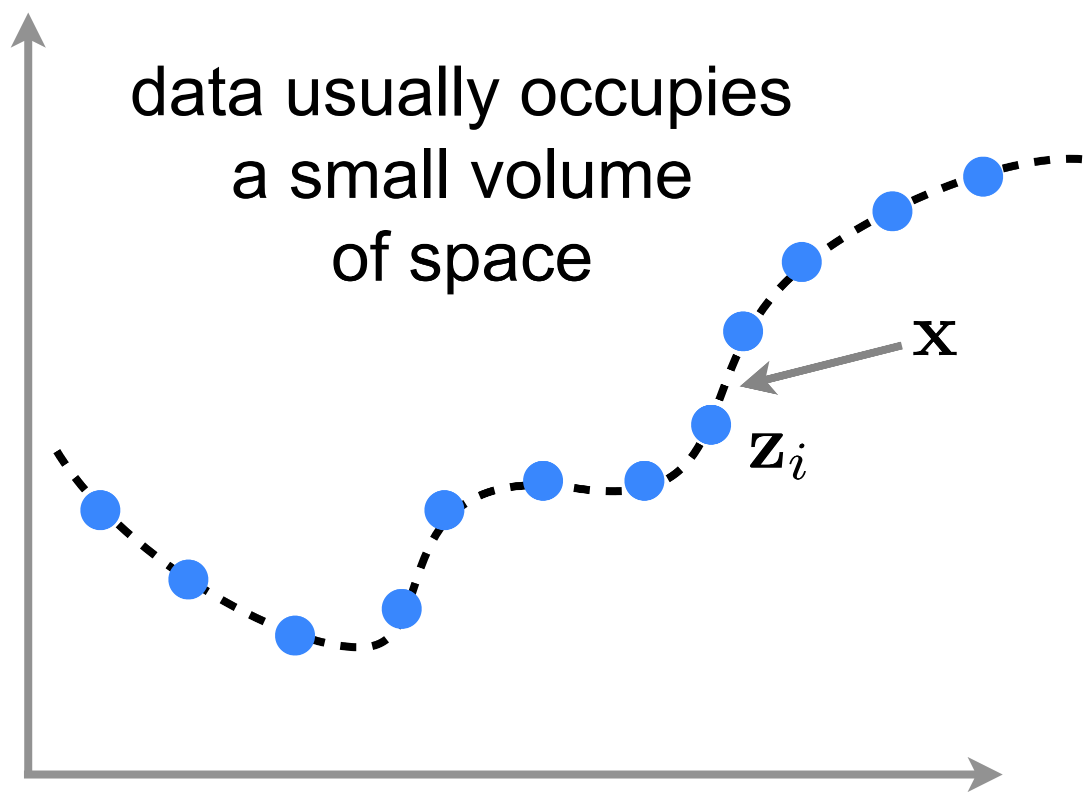
[Maji Berg 09]

Chi2 kernel

$$\alpha(x) = \frac{2(1+2i)x}{(i+x)(1+i+x)} \begin{bmatrix} i+1-x \\ x-1 \end{bmatrix}$$

[Vedaldi Zisserman 12]

Quantisation for compression

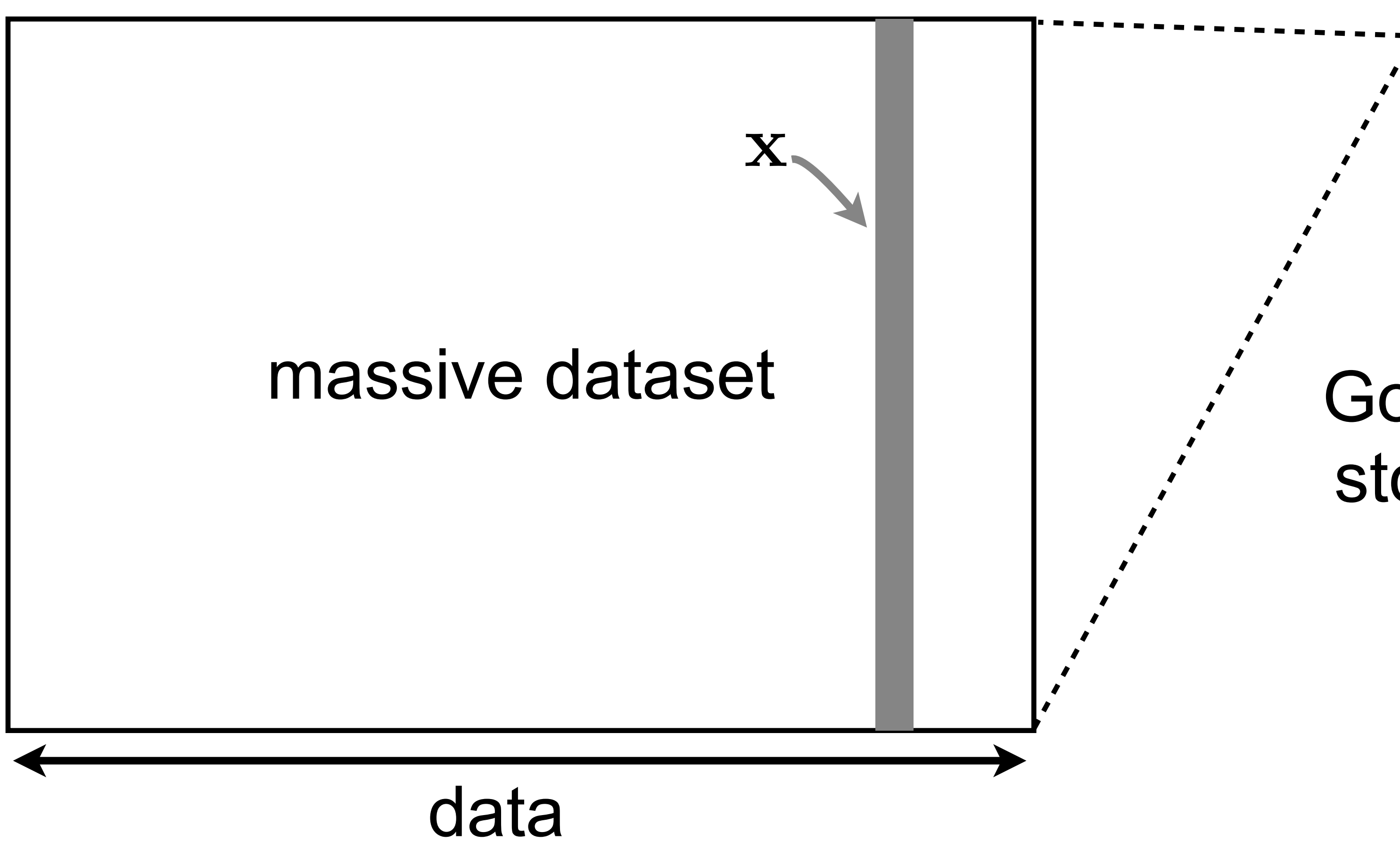


- $\mathbf{z}_i$ **codewords**

$Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_D\}$ **codebook**

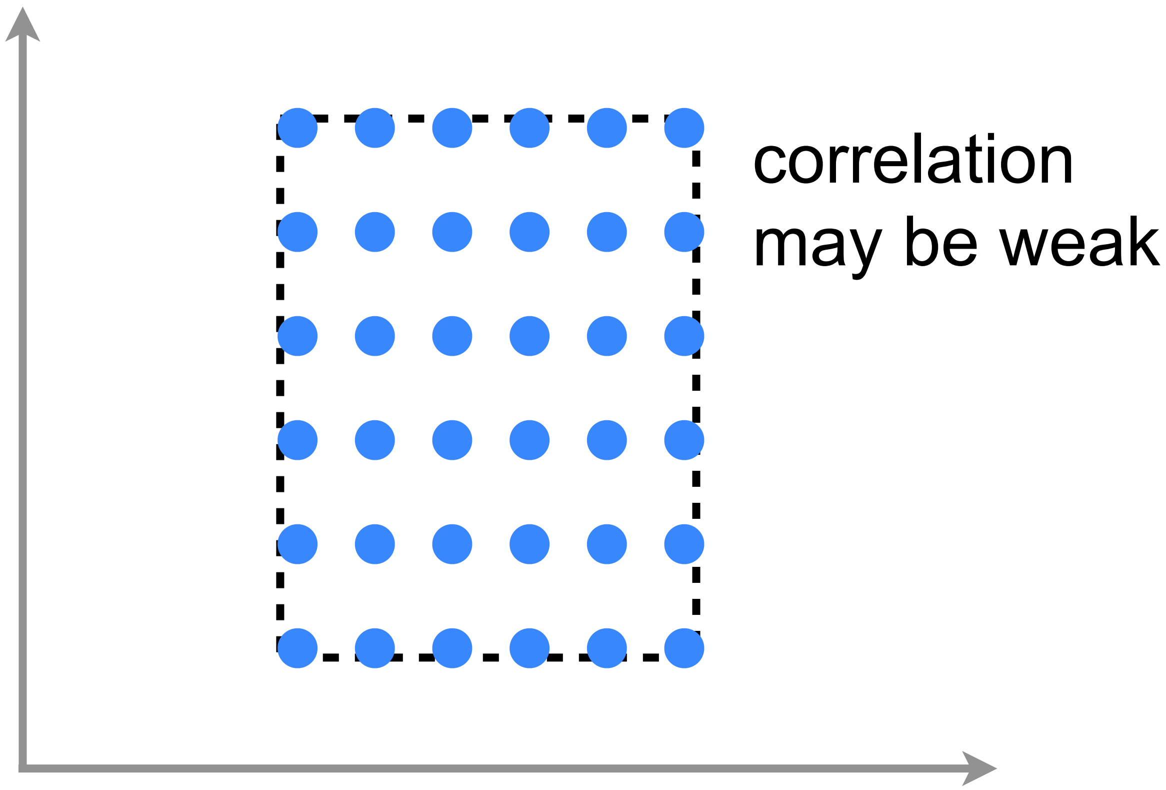
$\mathbf{x} \approx \mathbf{z}_{q(\mathbf{x})}$ **quantisation**: represent a data point by the closest codeword

saving: store only the index *q*, using $\log_2(D)$ bits



Good for sending information but ...
storing the coodebook is ultimately
as large as the data!

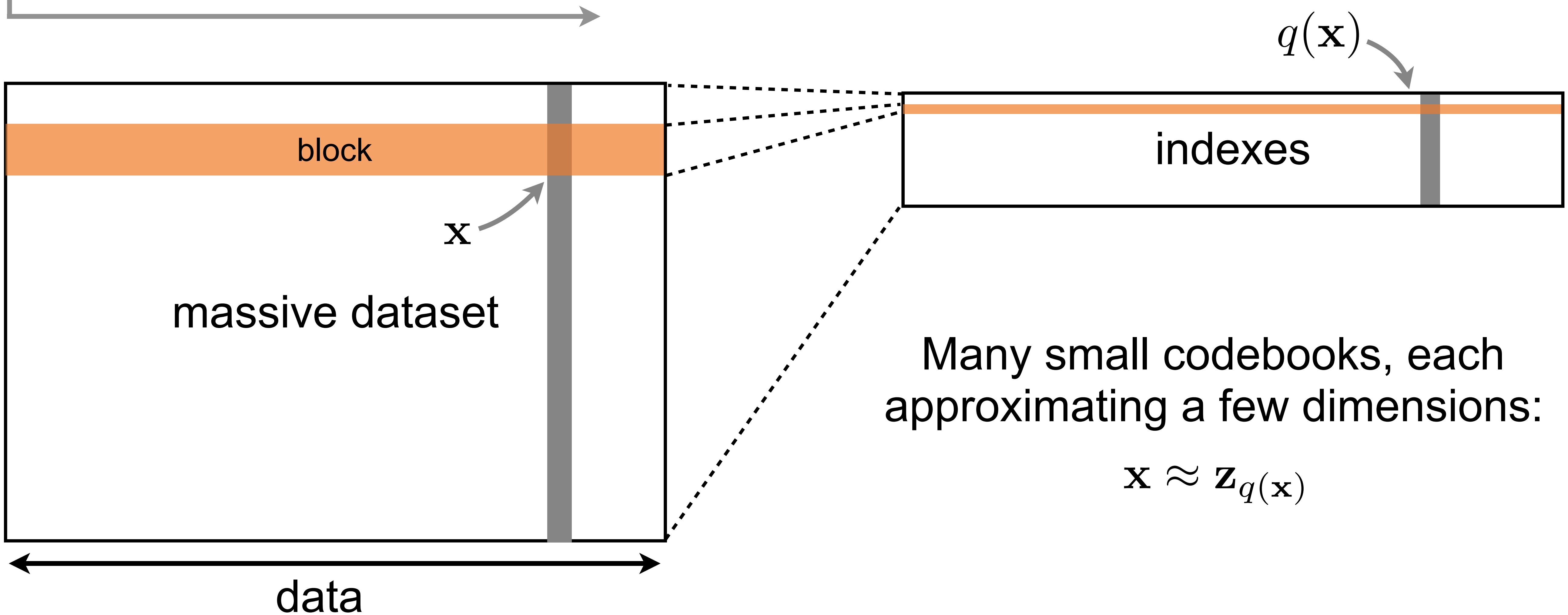
Product Quantisation (PQ)



Idea

Apply quantisation to blocks

[Gray Neuhoff 98, Jégou et al. 11, Sánchez and Perronnin 11]



Many small codebooks, each approximating a few dimensions:

$$\mathbf{x} \approx \mathbf{z}_{q(\mathbf{x})}$$

- **Data reconstruction** = codebook × sparse code

$$\mathbf{x} \approx \mathbf{z}_{q(x)} = \begin{bmatrix} \mathbf{z}_1 & \dots & \mathbf{z}_D \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = Z\Phi(\mathbf{x})$$

$q(\mathbf{x})$-th entry

- **Kernel reconstruction** = sparse feature map

$$K(\mathbf{x}, \mathbf{x}') \approx \Phi(\mathbf{x})^\top Z^\top Z \Phi(\mathbf{x}') = \Phi(\mathbf{x})^\top K_{ZZ} \Phi(\mathbf{x}')$$

Compute quickly many **inner products**

[Jégou et al. 11]

immediate expansion

delayed expansion

$$\{\mathbf{x}_i^\top \mathbf{x}\}_{i=1}^N \qquad \{\underbrace{(\Phi(\mathbf{x}_i)^\top Z^\top)}_{\text{collect in a large matrix } M}(Z\Phi(\mathbf{x}))\}_{i=1}^N \qquad M\Phi(\mathbf{x})$$

collect in a large matrix M

big saving!

Accumulate quickly many vectors

[Vedaldi Zisserman 12]

immediate expansion

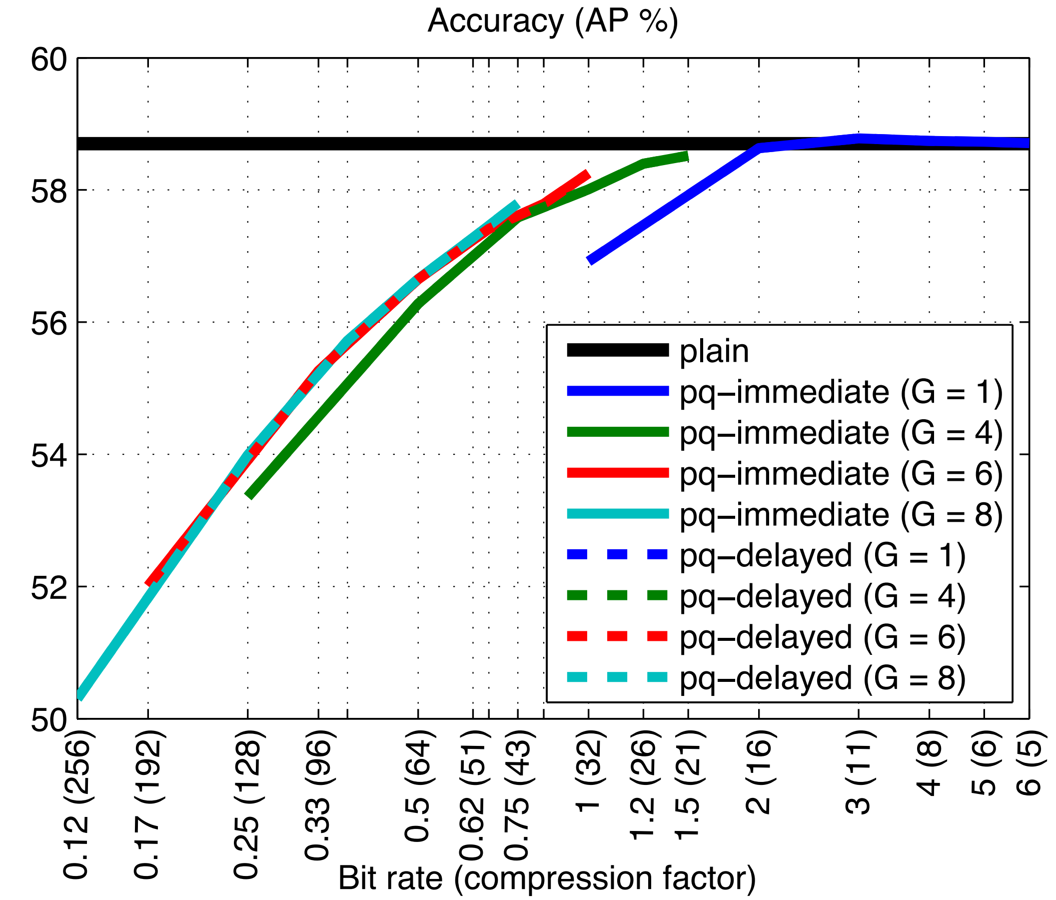
delayed expansion

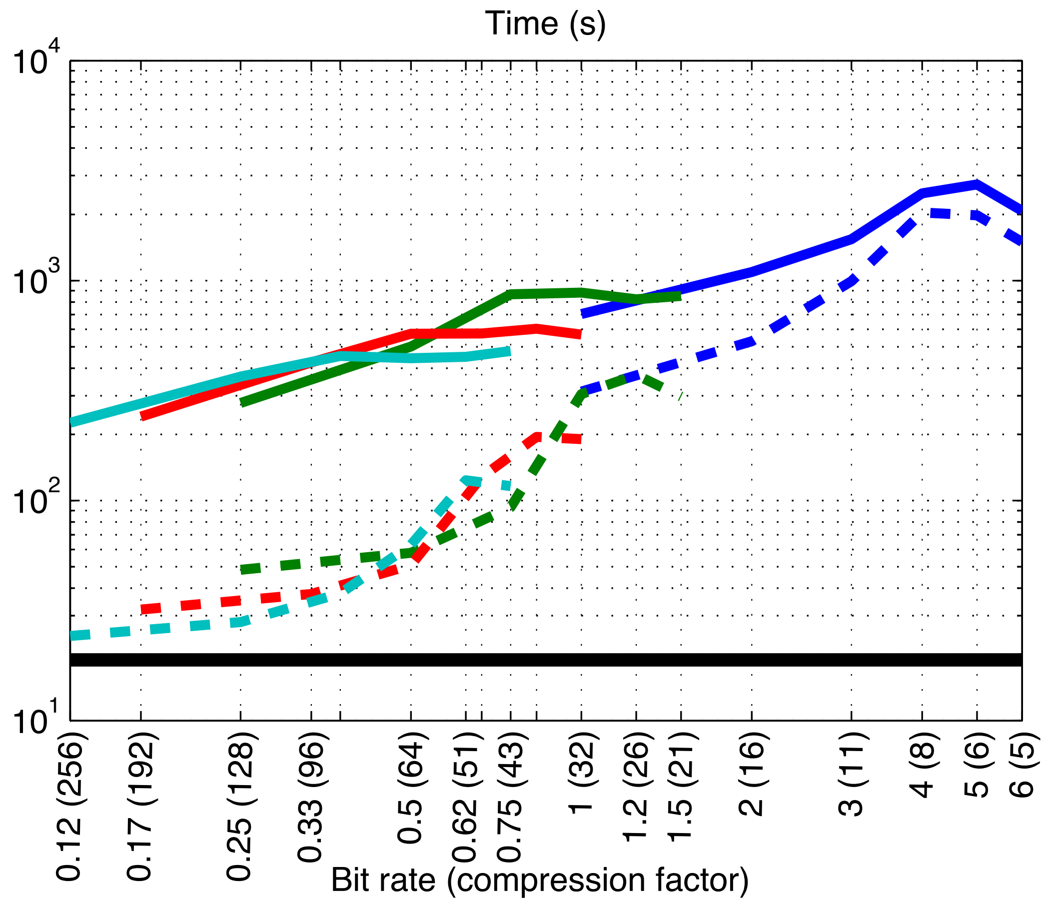
$$\sum_{i=1}^N \alpha_i \mathbf{x}_i \qquad \sum_{i=1}^N \alpha_i Z\Phi(\mathbf{x}_i) \qquad Z\sum_{i=1}^N \alpha_i \Phi(\mathbf{x}_i)$$

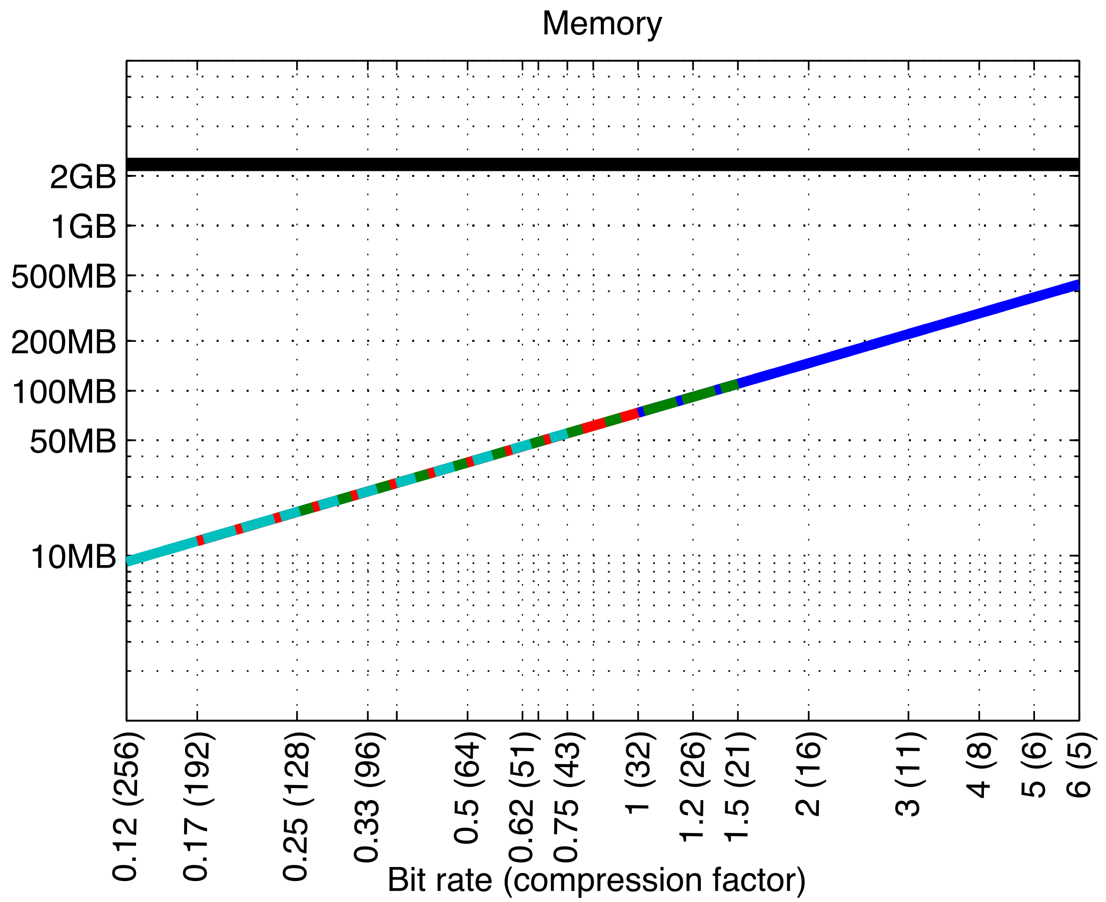
collet outside

Example: *Accelerating a cutting plane solver*

- **PASCAL VOC 2007**
  - 40,960-dimensional descriptors
  - ~ 0.5 Mb per image
  - 2GB of data
- **Training with PQ**
  - up to 100x memory reduction
  - up to 10 times faster







- **Feature maps**
  - Explicit linear embeddings reproducing a kernel
  - Allow tremendous speed-ups in learning
  - Particularly simple & efficient for additive kernels

- **Dense low-dimensional features**
  - Similar to PCA
  - *Homogeneous kernel map* (analytical for homogeneous additive)
  - *addKPCA* (empirical Nyström for additive)
  - *Random Fourier features* (Gaussian)
  - *Generalised Random Fourier Features* (Gaussian + additive)

- **Sparse high-dimensional features**
  - Similar to sparse coding
  - *Intersection kernel map* (sparse version)
  - *Product quantisation* for compression
  - Computation in the compressed domain
    - inner products and accumulation

Some references

- S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In Proc. ICCV, 2009.
- S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In Proc. CVPR, 2008.
- F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In Proc. CVPR, 2010.
- A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. PAMI, 34(3), 2012.
- A. Rahimi and B. Recht. Random features for large-scale kernel machines. In Proc. NIPS, 2007.
- H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. PAMI, 33(1), 2011.
- R. M. Gray and D. L. Neuhoff. Quantization. IEEE Trans. on Information Theory, 1998.