Efficient Additive Kernels via Explicit Feature Maps

Andrea Vedaldi Andrew Zisserman — Oxford University

Overview

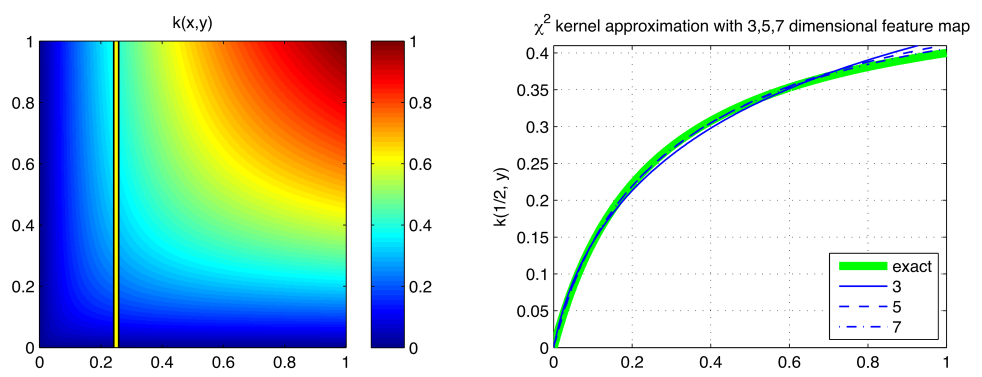
Linear kernel *vs* **additive** kernel:

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

Linear kernels yield fast training/testing of classifiers and compact representations. However, additive kernels are more accurate. We propose a **simple and efficient closed form data preprocessing step** that enables using an additive kernel as if it was linear.

The idea is to approximate the function $k(x,y)$ as the product of two small vectors $\widehat{\Psi}(x), \widehat{\Psi}(y)$. For instance for the χ^2 kernel:

$$k(x,y) = 2\frac{xy}{x+y} \approx \langle \widehat{\Psi}(x), \widehat{\Psi}(y) \rangle.$$



$\widehat{\Psi}(x)$ is given in closed form. For instance, a 3x approximation is given by:

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

The coefficients are obtained in closed form for all kernels and approximation orders.

Theory

- **Closed form feature maps** $\Psi(\mathbf{x})$ for all common additive kernels $K(\mathbf{x},\mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$ (X², intersection, ...).

	Hellinger's	Intersection	X²
$k(x,y) =$	$\sqrt{xy}$	$\min\{x,y\}$	$2\frac{xy}{x+y}$
$\mathcal{K}(\omega) =$	1	$e^{-\lvert\omega\rvert/2}$	$\operatorname{sech}(\omega/2)$
$\kappa(\lambda) =$	$\delta(\lambda)$	$\frac{2}{\pi}\frac{1}{1+4\lambda^2}$	$\operatorname{sech}(\pi\lambda)$

- **Finite approximations** with full characterization of the approximation error.

Practice

To kernelize a linear `algo(x)`:

1. Download VLFeat toolbox http://www.vlfeat.org
2. Preprocess data
```
psix = vl_homkermap(x, .5, 1, 'chi2') ;
```
3. Run `algo(psix)`.

Additive and homogeneous kernels

- **additive** decomposition: $K(\mathbf{x},\mathbf{y}) = \sum_{l=1}^{D} k(\mathbf{x}_l,\mathbf{y}_l)$
- $k: \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is itself a PD kernel
- k is **homogeneous**: $\forall c \geq 0: \quad k(cx,cy) = ck(x,y)$.

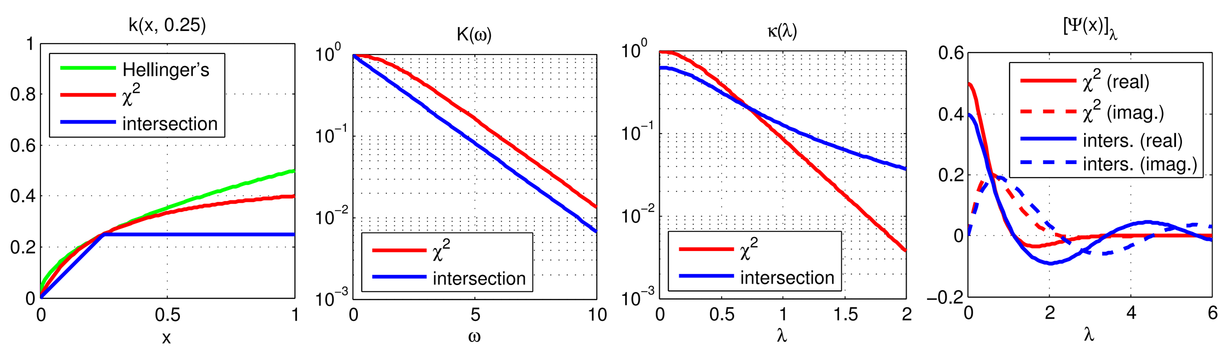
For any homogeneous kernel, by setting $c = \sqrt{xy}$ one obtains

$$k(x,y) = \sqrt{xy}\, k\left(\sqrt{\frac{x}{y}}, \sqrt{\frac{y}{x}}\right) = \sqrt{xy}\,\mathcal{K}\left(\log\frac{y}{x}\right)$$

where we call

$$\mathcal{K}(\omega) = \kappa\left(e^{-\omega/2}, e^{\omega/2}\right), \quad \omega = \log\frac{y}{x}$$

the **kernel signature**. This is a *scalar* function that fully characterizes the kernel.



γ-homogeneous variants

A kernel is γ-homogeneous if $k(cx,cy) = c^{\gamma}k(x,y)$. Given any signature $\mathcal{K}(\omega)$ we get a γ-homogeneous kernel by:

$$k(x,y) = (xy)^{\frac{\gamma}{2}}\mathcal{K}\left(\log\frac{y}{x}\right)$$

The linear kernel is $\gamma = 2$. Empirically, using $\gamma < 1$ can improve bag-of-words models.

Kernel normalization

In applications it is useful to normalize a kernel so that $K(\mathbf{x},\mathbf{x}) = 1$. For γ-homogeneous additive kernels this amounts to normalizing $\mathbf{x}$ in l^{γ} norm.

RBF variants

Our method can be combined with [Rahimi and Recht 07] to obtain feature maps for RBF-homogeneous kernels, such as the exponential X² kernel $\exp(-d^2_{\chi^2}(\mathbf{x},\mathbf{y})/(2\sigma^2))$.

From signatures to feature maps

Exact feature maps

For any homogeneous kernel there exists [Hein and Bousquet 2005] a measure $\kappa(\lambda)\,d\lambda$ such that

$$\mathcal{K}(\omega) = \int e^{-i\omega\lambda}\kappa(\lambda)\,d\lambda$$

I.e. $\kappa(\lambda)\,d\lambda$ is the Fourier transform of the signature $\mathcal{K}(\omega)$.

This yields a feature map for any homogeneous kernel:

$$[\Psi(x)]_\lambda = e^{-i\lambda\log x}\sqrt{x\kappa(\lambda)}$$

Verification:

$$\int [\Psi(x)]^*_\lambda [\Psi(y)]_\lambda d\lambda = \int e^{-i\lambda\log(y/x)}\kappa(\lambda)\,d\lambda = K(x,y).$$

For common kernels the computation is in *closed form*.

Approximate compact feature maps

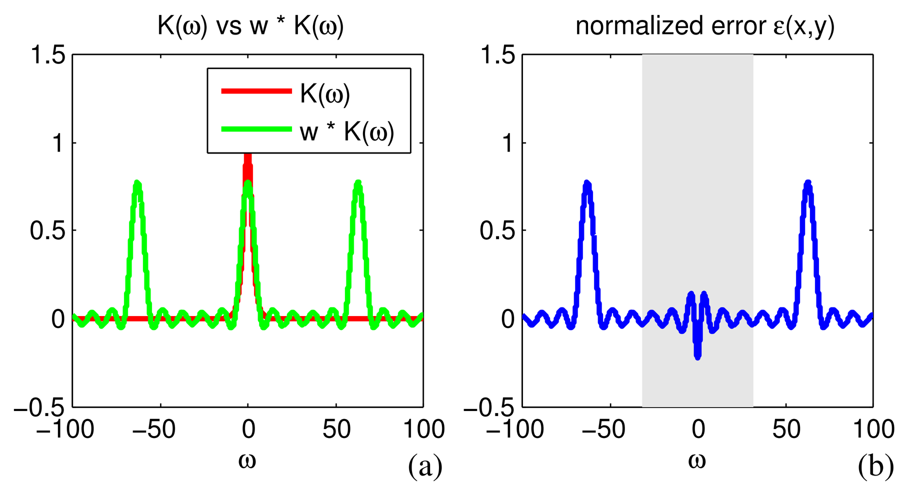
A finite approximated feature map $[\widehat{\Psi}(\mathbf{x})]_j$ can be obtained by taking $2n+1$ samples from $[\Psi(\mathbf{x})]_\lambda$.

Error analysis

The error of the finite representation can be analyzed by using the Fourier sampling theory. The **normalized approximation error** is given by the formula:

$$\epsilon(x,y) = \frac{\widehat{k}(x,y)}{\sqrt{xy}} - \frac{k(x,y)}{\sqrt{xy}} = ((\delta - w) * \mathcal{K})\left(\log\frac{y}{x}\right)$$

The error depends only on the ratio $\omega = \log(y/x)$. The **distortion function** $w(\omega)$ is a periodic sinc:



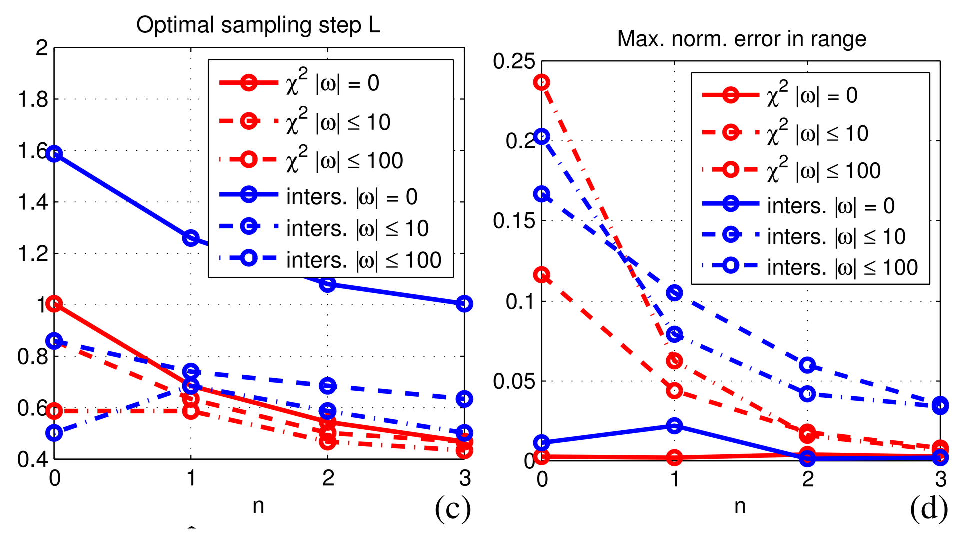
Conditions for a good approximation

The error is small for $|\omega| < \pi/L$ if

- $\mathcal{K}(\omega) \approx 0$ for $|\omega| > \pi/L$;
- $\kappa(\lambda) \approx 0$ for $|\lambda| > (n+1/2)L$.

Some consequences

- The X² kernel is easier to approximate than the intersection kernel as the fall-off in the λ domain is faster.
- The *normalized* error must be large for $|x| \gg |y|$, but in this case the *absolute* error is small (e.g. there is no error for $|x| = 0$).



Typically $L = 1/2$ and $n = 1$ are sufficient to get optimal performances in applications!

Experiments

Caltech-101

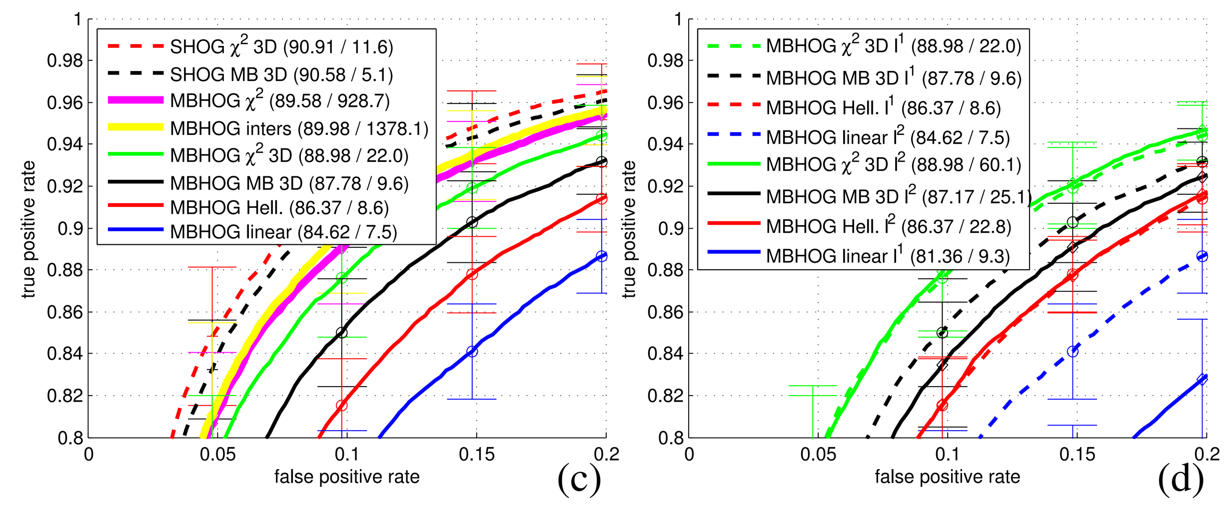
15 training and 15 testing images, PHOW features.

linear kernel acc.	linear kernel time	Hellinger's kernel acc.	Hellinger's kernel time
49.0±1.5	29.2±0.9	63.7±1.9	19.9±0.4

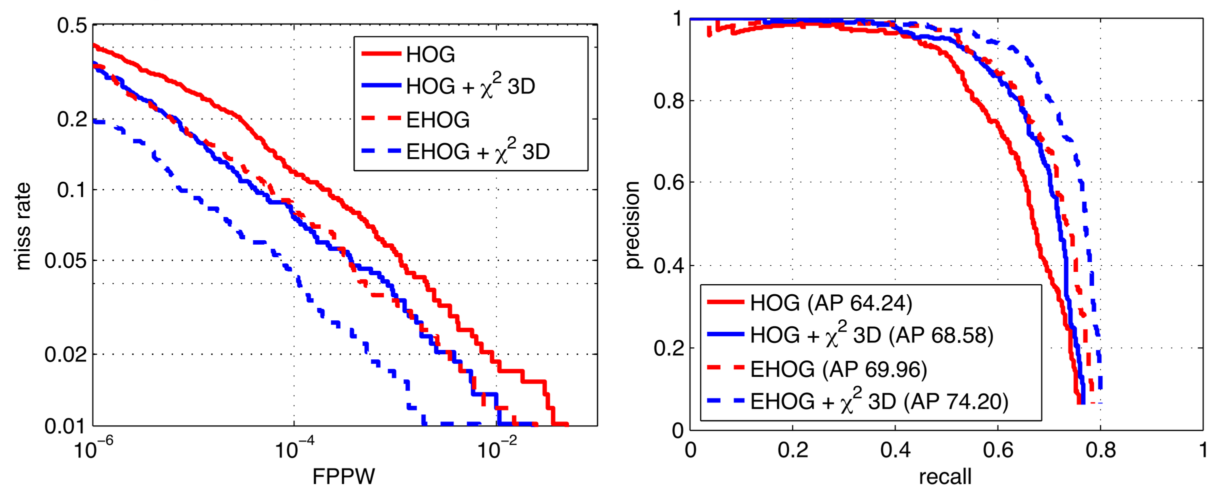
mthd.	dm.	χ^2 kernel acc.	χ^2 kernel time	inters. kernel acc.	inters. kernel time
kernel	–	64.2±1.7	388.4±8.7	62.2±1.8	354.7±24.4
appr.	1	62.4±1.6	**20.7**±0.3	62.0±1.4	22.9±0.7
appr.	3	64.2±1.5	58.4±7.2	63.9±1.2	66.5±2.3
appr.	5	64.0±1.6	101.3±0.7	64.0±1.7	105.8±6.5
appr-γ	3	**65.8**±1.5	54.7±6.2	65.7±1.5	52.6±7.7
MB	1	–	–	55.9±0.9	26.9±0.8
MB	3	–	–	60.5±1.3	25.5±1.2
MB	5	–	–	61.3±1.1	22.1±3.3

The 3×D approximated feature map is as good or better than the exacts kernels. The γ-homogeneous variant ($\gamma=1/2$) performs better still.

DaimlerChrysler pedestrians



INRIA pedestrians



By using the feature map it is trivial to extend a HOG detector to use a X² kernel. Scanning can then use convolution as usual and is very efficient.

Comparison with [Maji and Berg 09]

[Maji and Berg 09] propose an approximate feature map for the intersection kernel. Our method:

✓ works with all homogeneous kernels, not just the intersection kernel,

✓ spreads the error more uniformly and yields kernels exactly normalized,

✓ usually saturates performance with just 3×D features,

✗ yields a dense rather than a sparse expansion.