Advanced Machine Learning & Perception

Instructor: Tony Jebara

Topic 7

- Standard Kernels
- Generalization and Regularization
- Reproducing Kernel Hilbert Space and Kernel Properties
- Unusual Kernels
- String Kernels
- Probabilistic Kernels
- Probability Product Kernels

Kernels

- Inner product of mapped inputs: k(X,X') = f(X)Tf(X')
- Example Kernels:

p=# of twists in SVM decision boundary

P-th Order Polynomial Kernel: $k\left(x,y\right) = \left(x^T y + 1\right)^p$

RBF Kernel (infinite!): $k\left(x,y\right) = \exp\left(-{}^{1}\!/_{2\sigma^2} \left\|x-y\right\|^2\right)$

- Kernels replace inner products in SVMs, allow nonlinearity:

$$L_D : \max \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \quad \textit{subject to} \ \ \alpha_i \in \left[0, C\right]$$

$$f\left(x\right) = sign\left(\sum_i \alpha_i y_i k\left(x, x_i\right) + b\right)$$

- Solution is still a QP, uses Gram matrix K (positive definite)

$$K = \begin{bmatrix} k\left(x_1,x_1\right) & k\left(x_1,x_2\right) & k\left(x_1,x_3\right) \\ & k\left(x_2,x_2\right) & k\left(x_2,x_3\right) \\ & & k\left(x_3,x_3\right) \end{bmatrix} \qquad K_{ij} = k\left(x_i, x_j\right)$$

Standard Kernels over Vectors

•All take two input vectors x and y and produce a scalar:

$$k(x,y) = (x^T y + 1)^p$$
$$k(x,y) = \exp\left(-{}^{1}\!/_{2\sigma^2} \|x-y\|^2\right)$$
$$k(x,y) = \tanh(\kappa x^T y - \delta)$$

String Kernels

- Inputs are two strings: X = “aardvark”
  X' = “accent”
- Want $k(X,X') = \text{scalar value} = \phi(X)^T\phi(X')$
- One choice for features $\phi(X)$ is the number of times each substrings of length 1, 2 and 3 appears in X

$$\phi(X) = [\#a\ \#b\ \#c\ \#d\ \#e\ \#f\ \ldots\ \#y\ \#z\ \#aa\ \#ab\ \#ac\ \ldots]$$

$$= [3\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ \ldots\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ \ldots\ ]$$

$$\phi(X') = [1\ \ 0\ \ 2\ \ 0\ \ 1\ \ 0\ \ \ldots\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ \ldots\ ]$$

- Can do this efficiently via dynamic programming

String Kernels

•Want $k(X,X') = \text{scalar value} = \phi(X)^T\phi(X')$

•Explicit Kernel can be slow because of many substrings s in our final vocabulary

•$\phi_s(X)$ = number of times substring s appears in X

•$\phi_a(\text{aardvark}) = 3$ $\phi_{ar}(\text{aardvark}) = 1$

•Implicit Kernel is more efficient

$k(X,X')$: for each substring s of X, count how often s appears in X′

…via dynamic programming in time proportional to the product of the lengths of X and X′. This computes the dot product much more quickly!

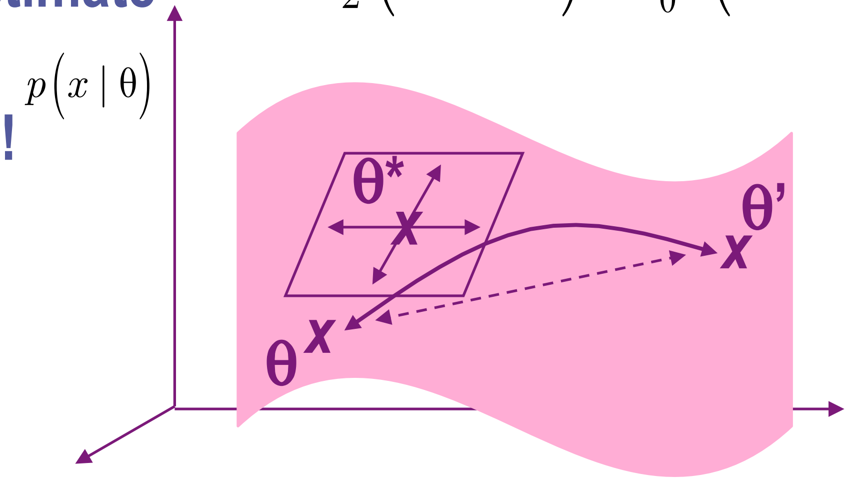
Fisher Kernels

Fisher Kernel: approximate distance between two generative models on statistical manifold using Kullback Leibler

$$KL\left(p \,\|\, p'\right) = \int p \log \frac{p}{p'} dx$$

Approximate KL by quadratic form local tangent space at ML estimate

$$KL \approx \tfrac{1}{2}\left(\theta - \theta'\right)^T I_{\theta^*}\left(\theta - \theta'\right)$$



affinity from distance= kernel !!!
via Fisher Info & gradients

$$K\left(\chi, \chi'\right) \approx U_x I_{\theta^*}^{-1} U_{x'}$$

$$U_\chi = \nabla_{\theta^*} \log p\left(\chi \mid \theta\right)$$

Fisher Info $I_{\theta^*}\left(i, j\right) = \int p\left(\chi \mid \theta^*\right) \frac{\partial \log p\left(\chi \mid \theta\right)}{\partial \theta_i} \frac{\partial \log p\left(\chi \mid \theta\right)}{\partial \theta_j} d\chi$

Other Divergences

Since the Kullback Leibler Divergence is asymmetric and tricky…

Find other divergences: $D(p,q) \geq 0, D(p,q) = 0 \; iff \; p = q$

Kullback-Leibler:
$$KL(p,q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Jensen-Shannon:
$$JS(p,q) = \tfrac{1}{2} KL\left(p, \tfrac{p+q}{2}\right) + \tfrac{1}{2} KL\left(q, \tfrac{p+q}{2}\right)$$

Symmetrized KL:
$$SKL(p,q) = \tfrac{1}{2} KL(p,q) + \tfrac{1}{2} KL(q,p)$$

Hellinger Divergence:
$$H(p,q) = \tfrac{1}{4} \sqrt{\int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx}$$

… makes a nice kernel:
$$H(p,q) = \sqrt{k(p,p) - 2k(p,q) + k(q,q)}$$

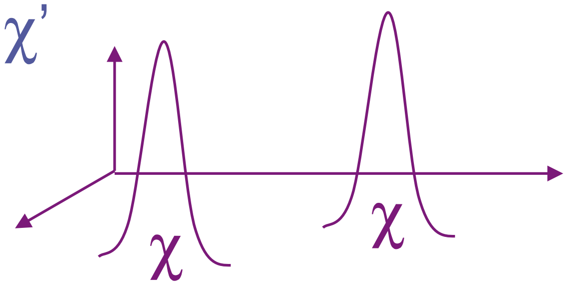
Probability Product Kernels

Computing the kernel: given inputs χ and χ'

1) Estimate Densities (I.e. ML):

$$\chi \to p(x) = p(x \mid \theta)$$

$$\chi' \to p'(x) = p(x \mid \theta')$$



2) Compute Bhattacharyya Affinity: $K(\chi,\chi') = \int \sqrt{p}\,\sqrt{p'}\,dx$

Probability Product Kernel: $K(\chi,\chi') = \int (p)^{\rho}(p')^{\rho}\,dx$

Bhattacharyya Kernel: $\rho = \frac{1}{2}$

Expected Likelihood Kernel: $\rho = 1$

$$K(\chi,\chi') = E_p[p'] = E_{p'}[p]$$

Gaussian Product Kernels

Gaussian Mean: (any ρ)
(continuous)
If μ=χ μ'=χ' get RBF Kernel
Fisher here is linear

$$K(\chi,\chi') = \int N^{\rho}(x \mid \mu)\, N^{\rho}(x \mid \mu')\, dx$$
$$\propto \exp\left(-\left\|\mu-\mu'\right\|^2 / \tilde{\sigma}\right)$$

Gaussian Covariance: (any ρ)

$$K(\chi,\chi') = \int N^{\rho}(x \mid \mu,\Sigma)\, N^{\rho}(x \mid \mu',\Sigma')\, dx$$
$$\propto \left|\Sigma\right|^{-\rho/2} \left|\Sigma'\right|^{-\rho/2} \left|\Sigma^{\dagger}\right|^{1/2} \exp\left(-\tfrac{\rho}{2}\mu^T\Sigma^{-1}\mu - \tfrac{\rho}{2}\mu'^T\,\Sigma'^{-1}\,\mu' + \tfrac{1}{2}\mu^{\dagger T}\Sigma^{\dagger}\mu^{\dagger}\right)$$
$$where\ \ \Sigma^{\dagger} = \left(\rho\Sigma^{-1} + \rho\Sigma'^{-1}\right)^{-1} \quad and\ \ \mu^{\dagger} = \rho\Sigma^{-1}\mu + \rho\Sigma'^{-1}\mu'$$

Fisher here is quadratic
But, how do we get a non-degenerate covariance from 1 data point?

Multinomial Product Kernels

Bernoulli:
(binary)

$$p(x \mid \theta) = \prod_{d=1}^{D} \gamma_d^{x_d} (1 - \gamma_d)^{1-x_d}$$

$$K(\chi, \chi') = \int (p(x))^{\rho} (p'(x))^{\rho} \, dx$$

$$= \prod_{d=1}^{D} \left[(\gamma_d \gamma_d')^{\rho} + (1 - \gamma_d)^{\rho} (1 - \gamma_d')^{\rho} \right]$$

Multinomial:
(discrete)

$$p(x \mid \theta) = \prod_{d=1}^{D} \alpha_d^{x_d}$$

$$K(\chi, \chi') = \sum_{d=1}^{D} (\alpha_d \alpha_d')^{\rho}$$

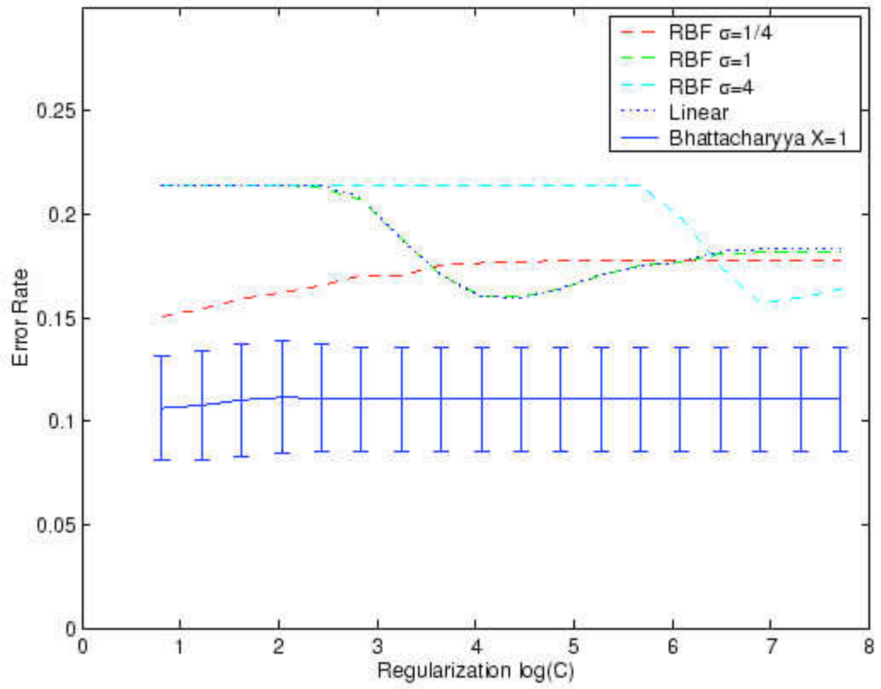
For multinomial counts (for N words):

$$K(\chi, \chi') = \left[\sum_{d=1}^{D} (\alpha_d \alpha_d')^{1/2} \right]^{N}$$

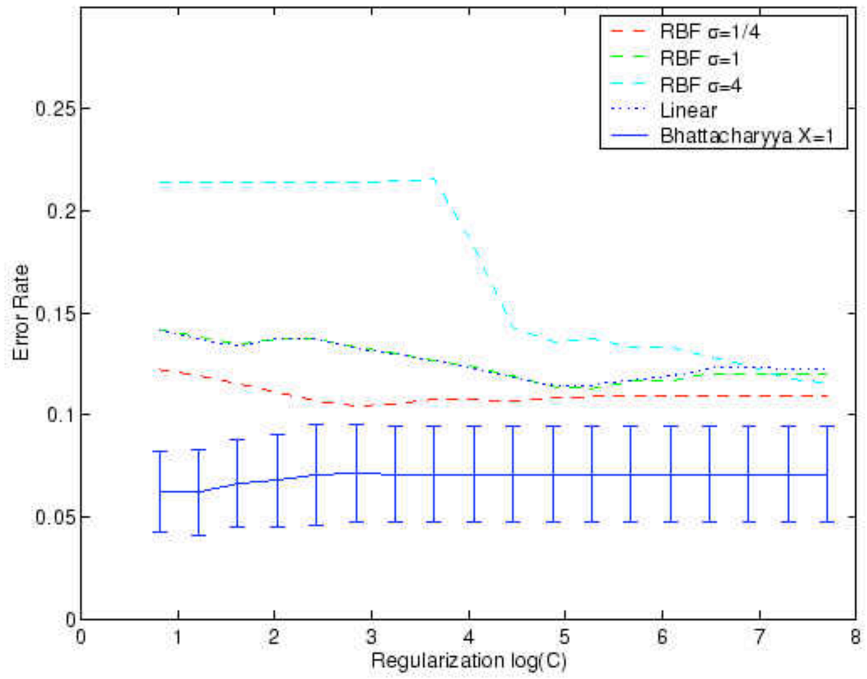
Fisher for Multinomial is linear

Multinomial Product Kernels

WebKB dataset: Faculty vs. student web page SVM kernel classifier
20-Fold Cross-Validation, 1641 student & 1124 faculty, ...
Use Bhattacharyya Kernel on multinomial (word frequency)



Training Set Size = 77



Training Set Size = 622

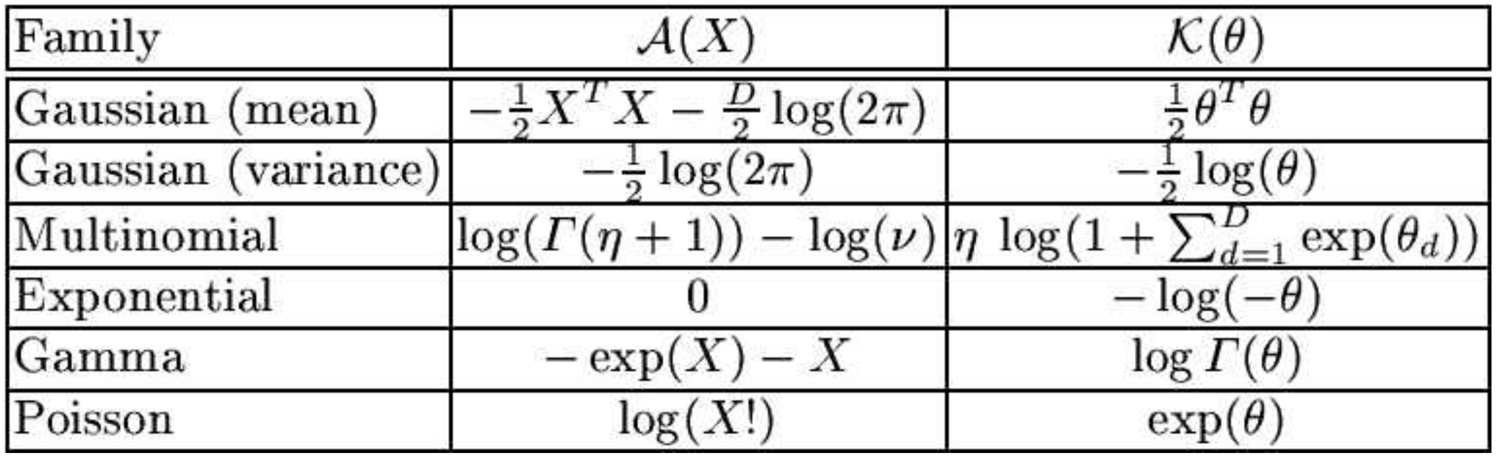
Exp. Family Product Kernels

Exponential Family: Many Properties, includes Gaussian Mean, Covariance, Multinomial, Binomial, Poisson, Exponential, Gamma, Bernoulli, Dirichlet, …

$$p\left(x \mid \theta\right) = \exp\left(A\left(x\right) + T\left(x\right)^{T}\theta - K\left(\theta\right)\right)$$

Maximum likelihood is straightforward: $\frac{\partial}{\partial\theta}K\left(\theta\right) = \frac{1}{n}\sum_{n} T\left(x_{n}\right)$

All have above form but different A(x), T(x), convex K(θ)



Family	$\mathcal{A}(X)$	$\mathcal{K}(\theta)$
Gaussian (mean)	$-\frac{1}{2}X^TX - \frac{D}{2}\log(2\pi)$	$\frac{1}{2}\theta^T\theta$
Gaussian (variance)	$-\frac{1}{2}\log(2\pi)$	$-\frac{1}{2}\log(\theta)$
Multinomial	$\log(\Gamma(\eta+1)) - \log(\nu)$	$\eta\ \log(1+\sum_{d=1}^{D}\exp(\theta_d))$
Exponential	0	$-\log(-\theta)$
Gamma	$-\exp(X) - X$	$\log\Gamma(\theta)$
Poisson	$\log(X!)$	$\exp(\theta)$

Exp. Family Product Kernels

Compute the Bhattacharyya Kernel for the e-family:

$$p(x \mid \theta) = \exp\left(A(x) + T(x)^T \theta - K(\theta)\right)$$

Analytic solution for e-family:

$$\begin{aligned} K(\chi, \chi') &= \int p(x \mid \theta)^{1/2} p(x \mid \theta')^{1/2} dx \\ &= \exp\left(K\left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta'\right) - \tfrac{1}{2}K\left(\tfrac{1}{2}\theta\right) - \tfrac{1}{2}K\left(\theta'\right)\right) \end{aligned}$$

Only depends on convex cumulant-generating function K(θ)

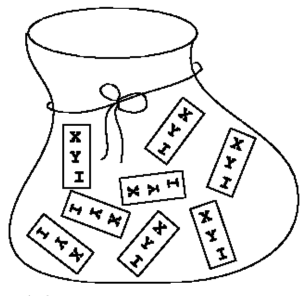
Meanwhile, Fisher Kernel is always linear in sufficient stats…

$$U_x = \nabla_{\theta^*} \log p(\chi \mid \theta) = T(\chi) - \nabla_{\theta^*} K(\theta)$$

$$U_\chi I_{\theta^*}^{-1} U_{\chi'} = \left(T(\chi) - g\right)^T I_{\theta^*}^{-1} \left(T(\chi') - g\right)$$

Gaussian Product Kernels

Instead of a single χ & χ'
Construct p & p' from many χ & χ'
I.e. use bag of vectors



$$\{\chi_1,\ldots,\chi_N\} \to p(x)$$

$$\{\chi_1',\ldots,\chi_{N'}'\} \to p'(x)$$

$$\mu = E\{\chi_i\} \qquad \Sigma = E\left\{(\chi_i-\mu)(\chi_i-\mu)^T\right\}$$

$$\sim N(x \mid \mu, \Sigma)$$

$$\sim N(x \mid \mu', \Sigma')$$

$$K(\chi,\chi') = |\Sigma|^{-\rho/2}|\Sigma'|^{-\rho/2}|\Sigma^\dagger|^{1/2}\exp\left(-\tfrac{\rho}{2}\mu^T\Sigma^{-1}\mu - \tfrac{\rho}{2}\mu'^T\Sigma'^{-1}\mu' + \tfrac{1}{2}\mu^{\dagger T}\Sigma^\dagger\mu^\dagger\right)$$

Kernelized Gaussian Product

Bhattacharyya affinity on Gaussians on bags $\{\chi,\ldots\}$ & $\{\chi',\ldots\}$

Invariant: to order of tuples in each bag

But too simple: overlap of two Gaussian distributions on images

Need more detail than mean and covariance of pixels…

Use Kernel Trick <u>again</u> when computing Gaussian mean & covariance

$$\kappa\left(\chi,\chi'\right)=\phi\left(\chi\right)^T\phi\left(\chi'\right)$$

Never compute outerproducts, use kernel, I.e. infinite RBF:

$$\kappa\left(\chi,\chi'\right)=\exp\left(-\tfrac{1}{2\sigma^2}\left\|\chi-\chi'\right\|^2\right)$$

Compute mini-kernel between each pixel in a given image…

Gives kernelized or augmented Gaussian μ and Σ via Gram

Kernelized Gaussian Product

Previously: $\mu = E\left\{\chi\right\}$ $\quad\Sigma = E\left\{\left(\chi-\mu\right)\left(\chi-\mu\right)^T\right\}$

Now have:

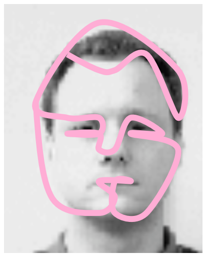
$$\mu = E\left\{\phi\left(\chi\right)\right\} \qquad \Sigma = E\left\{\left(\phi\left(\chi\right)-\mu\right)\left(\phi\left(\chi\right)-\mu\right)^T\right\}$$

Still invariant to order of pixels

Compute Hilbert Gaussian's mean & covariance of each image

bag or image is N x N pixel Gram matrix using kappa kernel

Use kernel PCA to regularize infinite dimensional RBF Gaussian

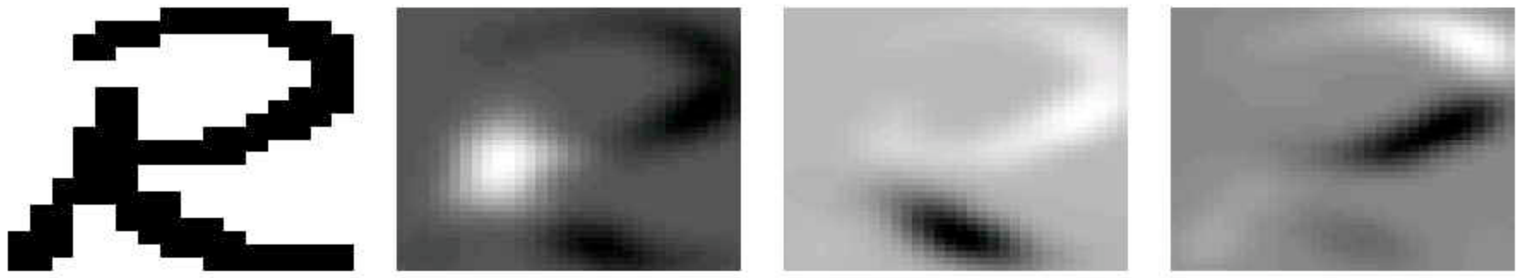


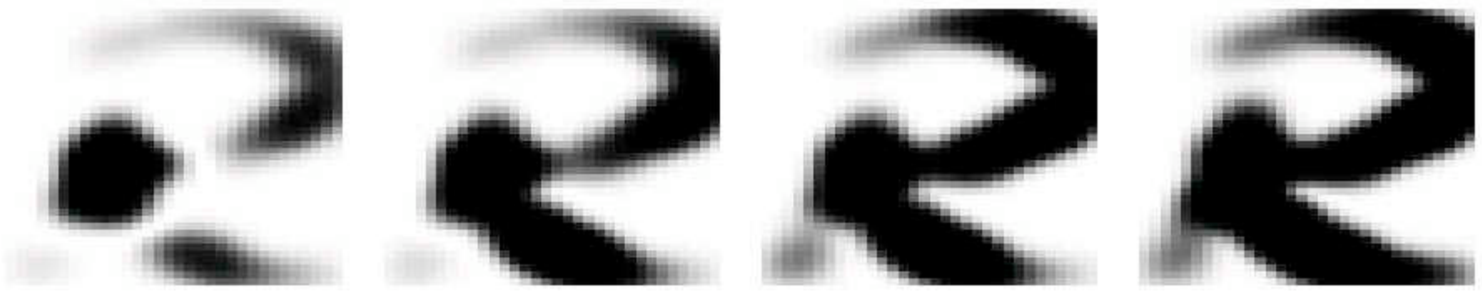
$$\begin{bmatrix} \kappa\left(x_1,x_1\right) & \dots & \kappa\left(x_1,x_N\right) \\ \vdots & & \vdots \\ \kappa\left(x_N,x_1\right) & \dots & \kappa\left(x_N,x_N\right) \end{bmatrix} \quad \begin{bmatrix} \kappa\left(x_1,x_1\right) & \dots & \kappa\left(x_1,x_M\right) \\ \vdots & & \vdots \\ \kappa\left(x_M,x_1\right) & \dots & \kappa\left(x_M,x_M\right) \end{bmatrix}$$

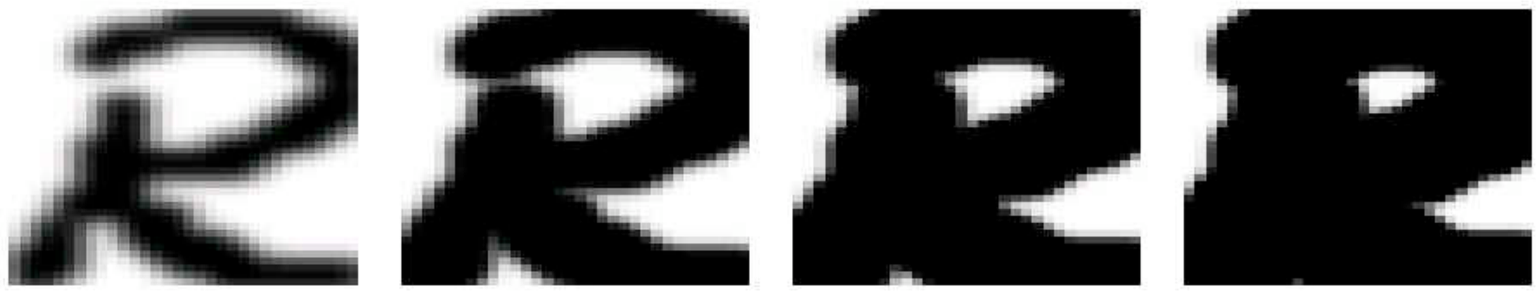
$$K\left(\phi\left(\chi\right)\middle\|\phi\left(\chi'\right)\right) \propto \left|\Sigma\right|^{-\rho/2}\left|\Sigma'\right|^{-\rho/2}\left|\Sigma^{\ddagger}\right|^{1/2}\exp\left(-\tfrac{\rho}{2}\mu^T\Sigma^{-1}\mu-\tfrac{\rho}{2}\mu'^T\Sigma'^{-1}\mu'+\tfrac{1}{2}\mu^{\ddagger T}\Sigma^{\ddagger}\mu^{\ddagger}\right)$$

Puts all dimensions (X,Y,I) on an equal footing

Kernelized Gaussian Product

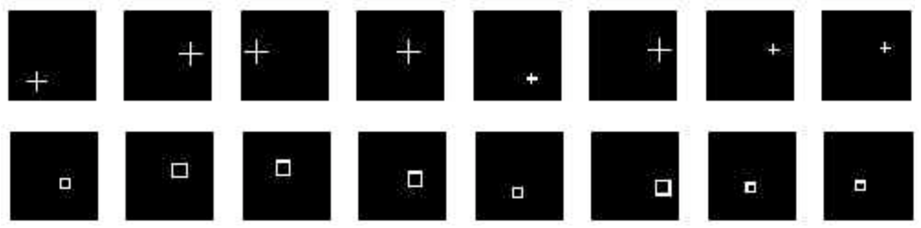

Letter ‘R’ with 3 KPCA Components of RBF Kernel


Reconstruction of Letter ‘R’ with 1-4 KPCA with RBF Kernel


Reconstruction of Letter ‘R’ with 3 KPCA with RBF Kernel + Smoothing

Kernelized Gaussian Product

100 40x40 monochromatic images of crosses & squares translating & scaling

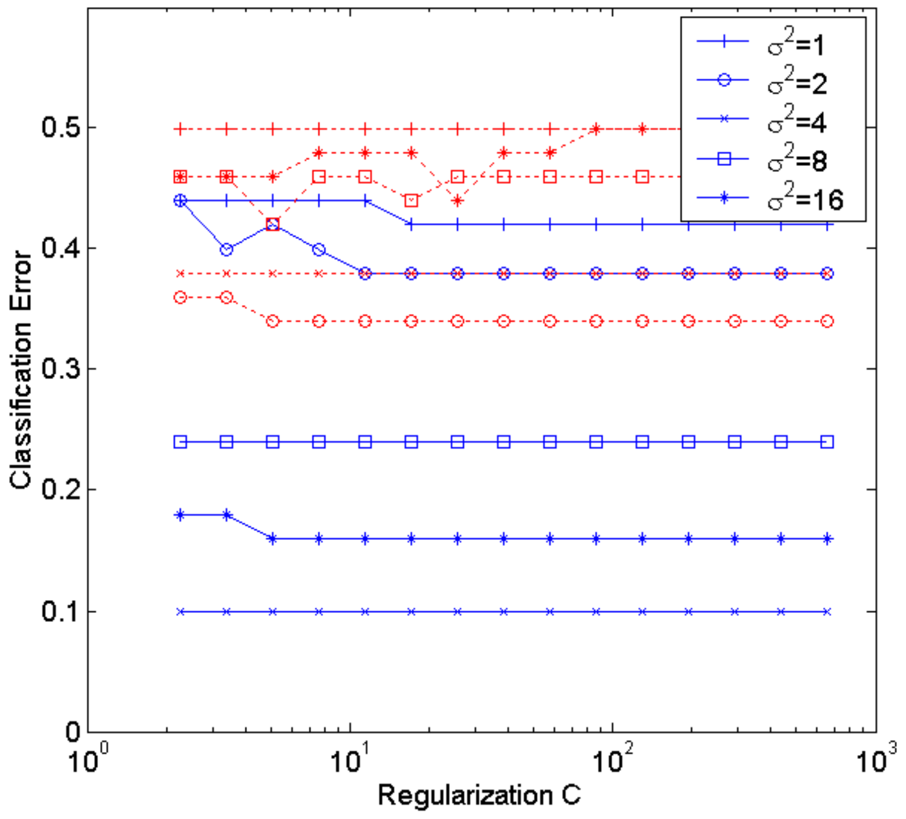


SVM: Train on 50, Test on 5

Fisher for Gaussian is Quadratic Kernel

RBF Kernel (red) 67% accuracy

Bhattacharyya (blue) 90% accuracy

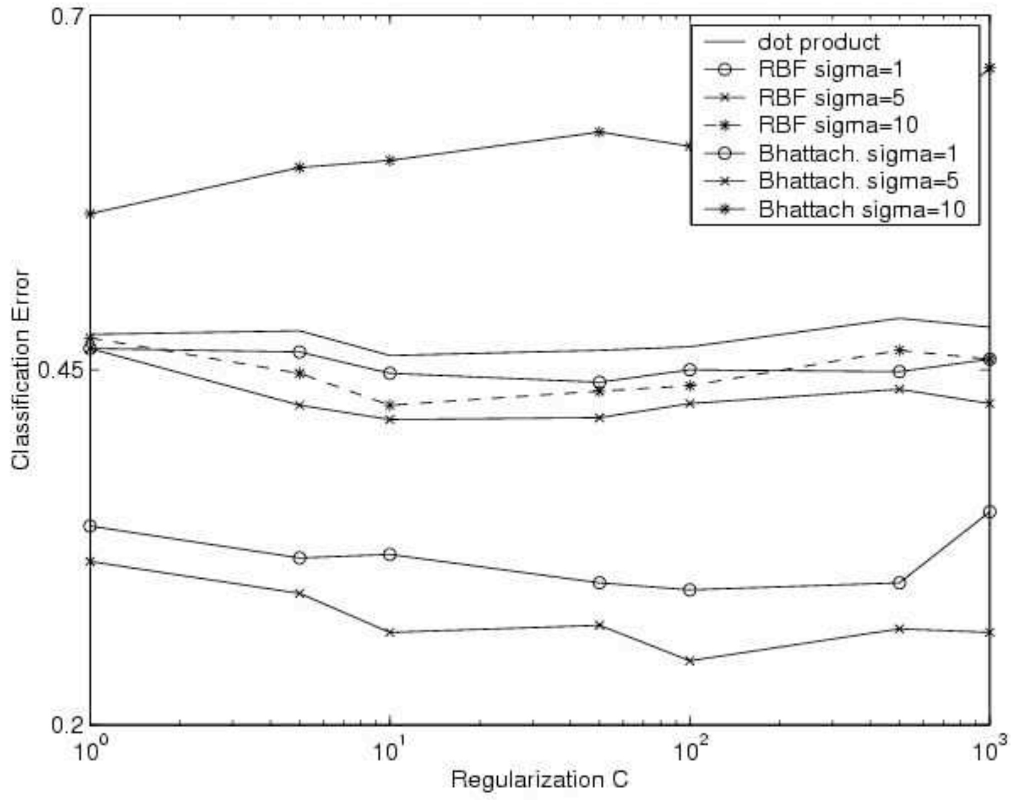


Kernelized Gaussian Product

SVM Classification of NIST digit images 0,1,…,9
Sample each image to get bag of 30 (X,Y) pixels
Train on random 120, test on random 80

bag-of-vectors Bhattacharyya outperforms standard RBF due to built-in invariance

Fisher Kernel for Gaussian is quadratic



Mixture Product Kernels

Beyond Exponential Family: Mixtures and Hidden Variables

Easier for ρ=1 Expected Likelihood kernel…

$$\chi \to p(x) = \sum_{m=1}^{M} p(m) p(x \mid m)$$

$$\begin{tikzcd} m \arrow[r] & x \end{tikzcd}$$

$$\chi' \to p'(x) = \sum_{n=1}^{N} p'(n) p'(x \mid n)$$

$$\begin{tikzcd} n \arrow[r] & x \end{tikzcd}$$

$$K(\chi, \chi') = \int p(x)\, p'(x)\, dx$$

$$= \sum_{m=1}^{M} \sum_{n=1}^{N} p(m)\, p'(n) \int p(x \mid m)\, p'(x \mid n)\, dx$$

$$= \sum_{m=1}^{M} \sum_{n=1}^{N} p(m)\, p'(n)\, K_{m,n}(\chi, \chi')$$

$$\begin{tikzcd} m \arrow[dr] & \\ & x \\ n \arrow[ur] & \end{tikzcd}$$

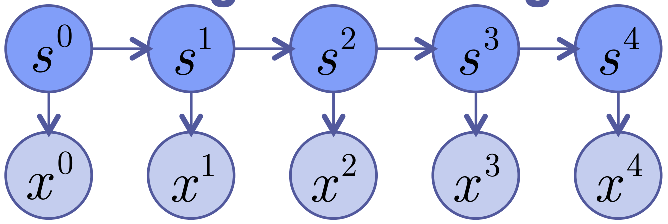
Use M*N subkernel evaluations from our previous repertoire

HMM Product Kernels

Hidden Markov Models: (sequences)

$$p(x) = \sum_S p(s_0)\, p(x_0 \mid s_0) \prod_{t=1}^{T} p(s_t \mid s_{t-1})\, p(x_t \mid s_t)$$

# of hidden configurations large



$$\#configs = |s|^T$$

Kernel: $K(\chi, \chi') = \int p(x)\, p'(x)\, dx$

$$= \sum_S \sum_U p(S)\, p'(U) \int p(x \mid S)\, p'(x \mid U)\, dx$$

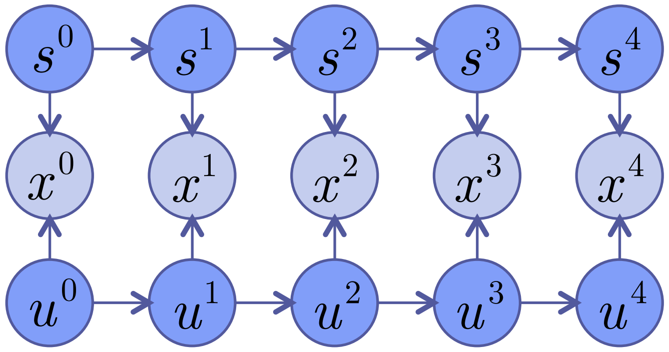
$$= \sum_S \sum_U p(S)\, p'(U)\, K_{S,U}(x, x')$$

Do we need to consider raw cross-product of hiddens? $O\left(|s|^T \times |u|^T\right)$

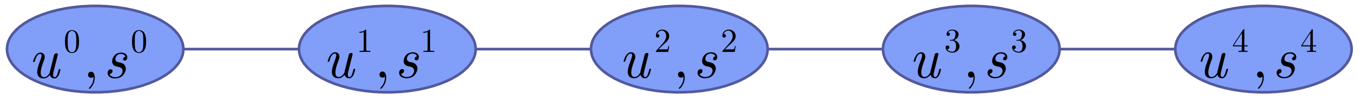
HMM Product Kernels

$$K(\chi,\chi') = \sum_S \sum_U \prod_t p(s_t \mid s_{t-1}) p'(u_t \mid u_{t-1}) \int p(x_t \mid s_t) p'(x_t \mid u_t) dx_t$$
$$= \sum_{S,U} \prod_t p(s_t \mid s_{t-1}) p'(u_t \mid u_{t-1}) K_{s_t,u_t}$$
$$= \sum_{S,U} p(s_0) p'(u_0) \psi(s_0,u_0) \prod_t p(s_t \mid s_{t-1}) p'(u_t \mid u_{t-1}) \psi(s_t,u_t)$$

Take advantage of structure in HMMs via Bayesian network



Only compute subkernels for common parents
Evaluate total of $O\left(T|s||u|\right)$ subkernels
Form +ve clique potential functions, sum via junction tree algorithm



Sampling Product Kernels

Sampling: approximate K $K(\chi,\chi') = \int p(x)\,p'(x)\,dx$

By definition, generative models can:

1) **Generate a Sample**
2) **Compute Likelihood of a Sample**

Thus, approximate probability product via sampling:

$$K(\chi,\chi') = K(p,p') = \frac{\beta}{N}\sum_{\substack{x_i \sim p(x)\\ i=1\ldots N}} p'(x_i) + \frac{1-\beta}{N'}\sum_{\substack{x_i' \sim p'(x)\\ i=1\ldots N'}} p(x_i')$$

Beta controls how much sampling from each distribution…