Advanced Machine Learning & Perception

Instructor: Tony Jebara

Topic 7

- Standard Kernels
- Unusual Kernels
- String Kernels
- Probabilistic Kernels
- Probability Product Kernels

Kernel Machines

- Many algorithms use kernels... all inputs summarized via a matrix of kernel evaluations between pairs of points

$$or\ equivalently\ \ K_{ij} = k\left(x_i, x_j\right)$$

- Generally called Kernel Machines when classifiers
- Examples:

Support Vector Machine
Bayes Point Machine
Relevance Vector Machine
Gaussian Process Regression
Support Vector Machine Regression
Ellipsoidal Kernel Machines
Semidefinite Embedding
Minimum Volume Embedding
etc.

Kernels

- Inner product of mapped inputs: $k(x,y) = \phi(x)^T \phi(x)$
- Example Kernels:

  p=# of twists in SVM decision boundary

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

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

- Kernels replace inner products in SVMs, allow nonlinearity:

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

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

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

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) \\ k(x_3,x_1) & k(x_3,x_2) & k(x_3,x_3) \end{bmatrix} \quad or\ equivalently \quad K_{ij} = k(x_i, x_j)$$

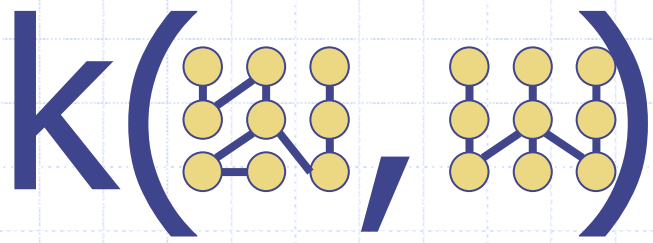
Standard Kernels over Vectors

- Accept two input vectors x, y and produce a scalar output:

- Homogeneous Polynomial $k(x,y) = (x^T y)^p$
- Non-Homogeneous Polynomial $k(x,y) = (x^T y + 1)^p$
- RBF kernel $k(x,y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$
- Tanh Kernel $k(x,y) = \tanh(\kappa x^T y - \delta)$
- Triangular Kernel $k(x,y) = 1 - \left|(x-y)/d\right|$

Kernels more Generally

- Can work with non-vectorial inputs!
- Consider the case where the (x,y) pair are
  - strings
  - graphs
  - probability distributions
  - diffusion process



- Kernel just need to satisfy certain basic properties:

1) It's symmetric $k(x,y) = k(y,x)$

2) Satisfies Mercer's Condition: $\int k(x,y)\,g(x)\,g(y)\,dxdy \geq 0$

$for\ any\ g(x)\ such\ that \int g(x)^2\,dx\ \ is\ finite$

...similar to saying Gram matrix is positive definite

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 … #y #z #aa #ab #ac …]

= [3 0 0 1 0 0 … 0 0 1 0 0 …]

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

- 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

$\phi_a(\text{aardvark}) = 3$ $\phi_{ar}(\text{aardvark}) = 2$

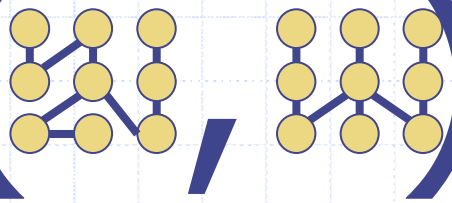
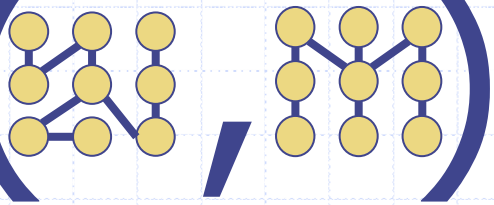
- Implicit Kernel is more efficient

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!

Invariance/Independence in Kernels

- Not only do we want kernels that satisfy Mercer…
- We want some kind of invariance: $k\left(x,y\right)=k\left(x,Gy\right)$

k() = k()

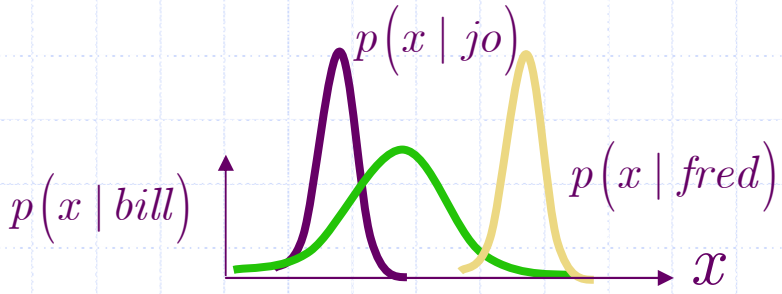
- Another consideration is independence in kernels…
- For example, input has 2 dimensions that are independent

$$x=\begin{bmatrix}x_1\\x_2\end{bmatrix} \qquad y=\begin{bmatrix}y_1\\y_2\end{bmatrix} \qquad k\left(x,y\right)=k_1\left(x_1,y_1\right)+k_2\left(x_2,y_2\right)$$

- How to get such desiderata in general? à Open Problem
- One way is to work with probabilistic kernels…

Probabilistic Kernels

- We will consider two cases of probabilistic kernels…
  Fisher Kernel and Probability Product Kernel
- For example, how similar are a set of people who have p(x) each person's weight distribution over 365 days



$$\rightarrow \quad K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) \\ k(x_3, x_1) & k(x_3, x_2) & k(x_3, x_3) \end{bmatrix}$$

- Fisher Kernel à gets an affinity between p and p' by approximating KL-divergence $KL(p \| p') = \int p \log \frac{p}{p'} dx$
- Probability Product Kernel à gets an affinity by using the Hellinger-divergence $H(p \| p') = \frac{1}{2} \int \left(\sqrt{p} - \sqrt{p'} \right)^2 dx$

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

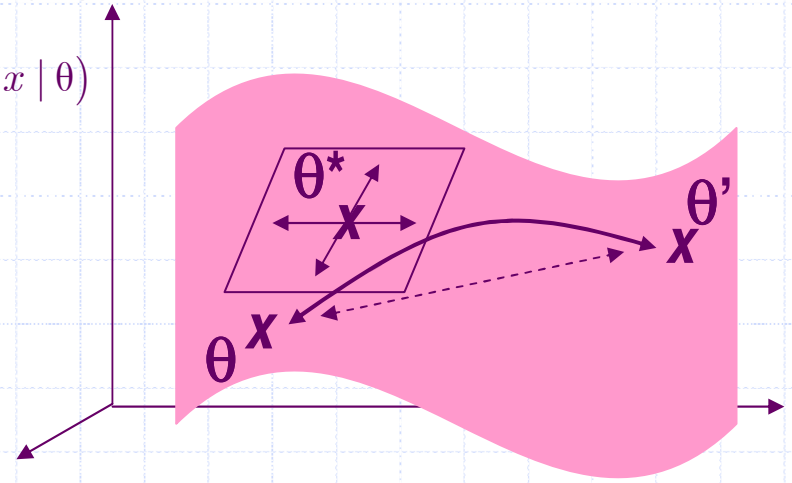
$$KL \approx \tfrac{1}{2}\left(\theta - \theta'\right)^T I_{\theta^*} \left(\theta - \theta'\right)$$

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

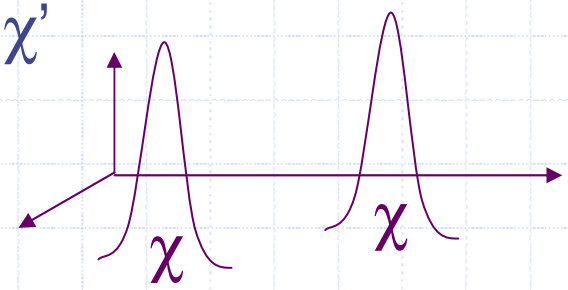
Probability Product Kernels

Computing the kernel: given inputs χ and χ'

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

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

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



2) The natural affinity for Hellinger is known: Bhattacharyya Affinity

$$k(\chi, \chi') = \int \sqrt{p}\,\sqrt{p'}\,dx$$

Probability Product Kernel:

$$k_\rho(\chi, \chi') = \int (p)^\rho (p')^\rho\,dx$$

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)

$$k(\chi,\chi') = \int N^{\rho}(x \mid \mu,\Sigma) N^{\rho}(x \mid \mu',\Sigma') dx$$

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

$$where\ \ \Sigma^{\dagger} = \left(\rho\Sigma^{-1} + \rho\Sigma'^{-1}\right)^{-1}\ \ and\ \ \mu^{\dagger} = \rho\Sigma^{-1}\mu + \rho\Sigma'^{-1}\mu'$$

Fisher here is just a quadratic kernel...

Multinomial Product Kernels

Bernoulli: (binary)

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

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

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

Multinomial: (discrete)

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

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

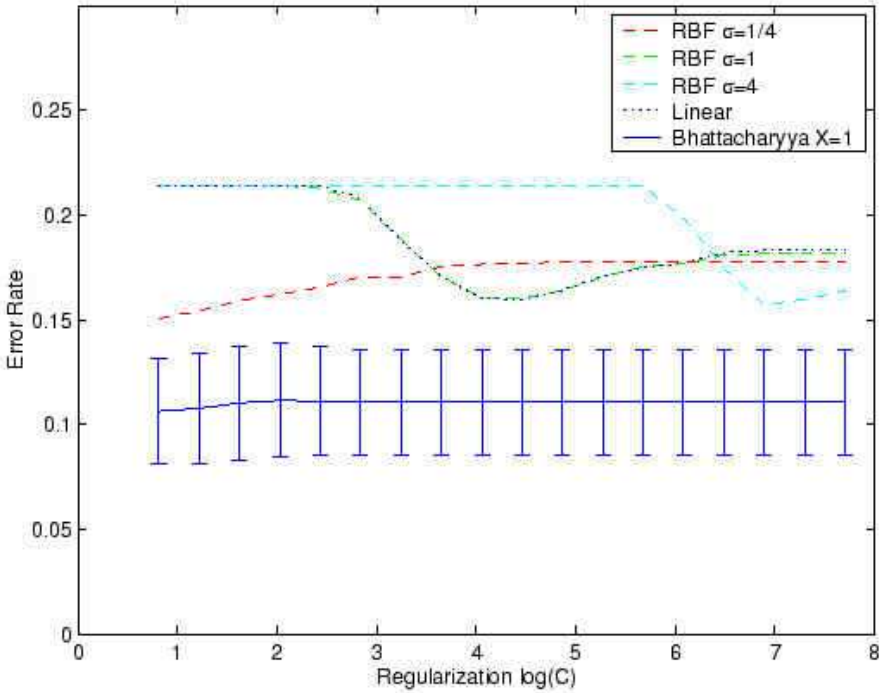
For multinomial counts (for N words):

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

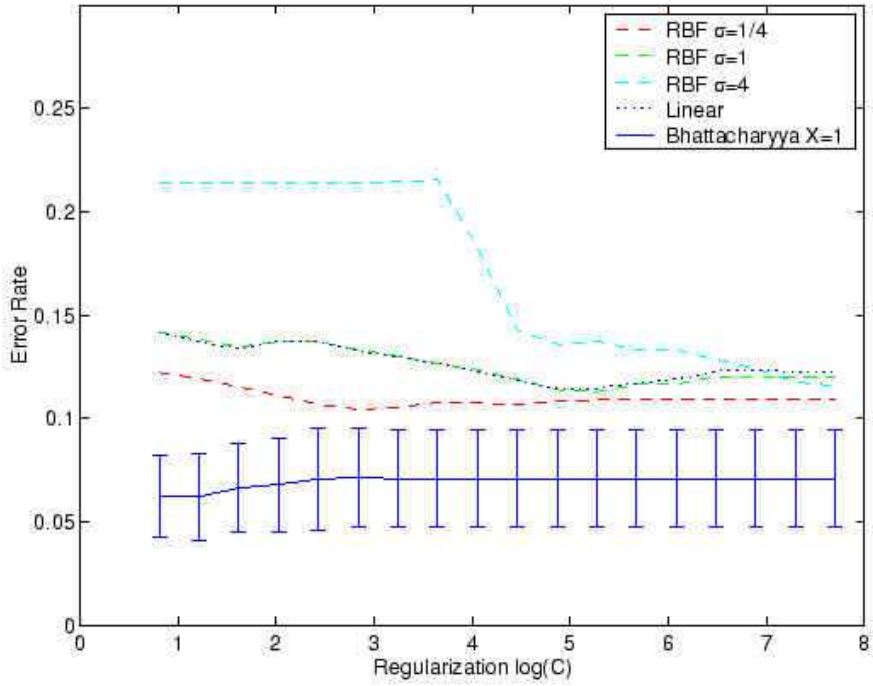
Fisher for Multinomial is linear

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

Exponential Family Kernels

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

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

Maximum likelihood is straightforward: $\frac{\partial}{\partial \theta} K(\theta) = \frac{1}{n} \sum_n T(x_n)$

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

Family	$\mathcal{A}(X)$	$\mathcal{K}(\theta)$
Gaussian (mean)	$-\frac{1}{2} X^T X - \frac{D}{2} \log(2\pi)$	$\frac{1}{2} \theta^T \theta$
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	$\log(X!)$	$\exp(\theta)$

Exponential Family 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:

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

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

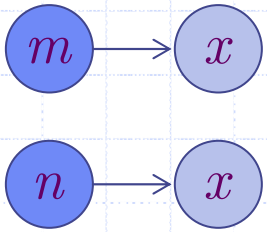
Meanwhile, Fisher Kernel is always linear in sufficient stats…

$$U_x = \nabla_{\theta^*} \log p(\chi \mid \theta) = T(\chi) - \nabla_{\theta^*} K(\theta)$$
$$U_\chi I_{\theta^*}^{-1} U_{\chi'} = \left(T(\chi) - g\right)^T I_{\theta^*}^{-1} \left(T(\chi') - g\right)$$

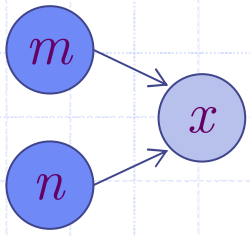
Mixture Product Kernels

Beyond Exponential Family: Mixtures and Hidden Variables
Easier for ρ=1 Expected Likelihood kernel…

$$\chi \rightarrow p(x) = \sum_{m=1}^{M} p(m) p\left(x \mid m\right)$$
$$\chi' \rightarrow p'(x) = \sum_{n=1}^{N} p'(n) p'\left(x \mid n\right)$$



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



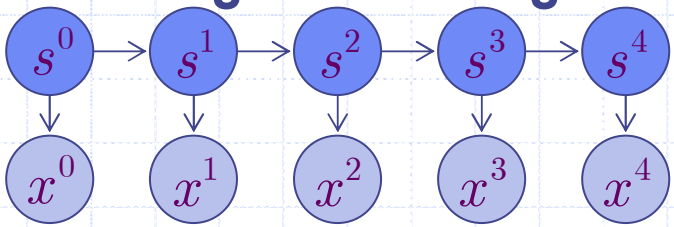
Use M*N subkernel evaluations from our previous repertoire

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:

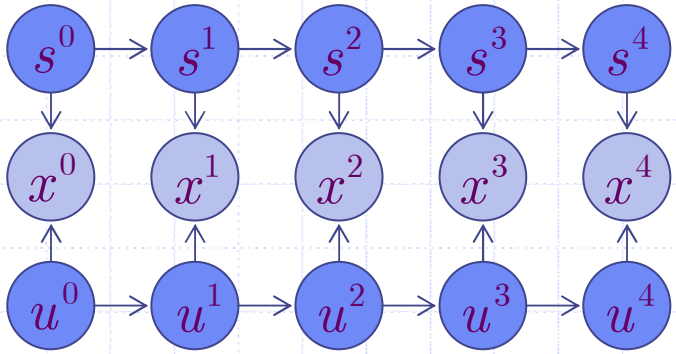
$$\begin{aligned} k(\chi, \chi') &= \int p(x)\, p'(x)\, dx \\ &= \sum_S \sum_U p(S)\, p'(U) \int p(x \mid S)\, p'(x \mid U)\, dx \\ &= \sum_S \sum_U p(S)\, p'(U)\, k_{S,U}(x, x') \end{aligned}$$

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

HMM Product Kernels

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

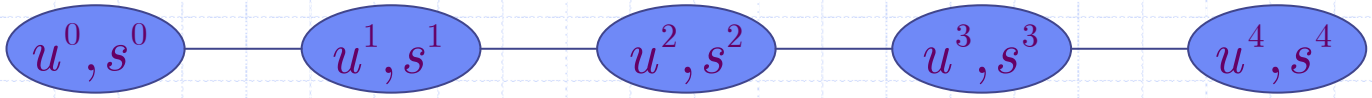
Take advantage of structure in HMMs via Bayesian network



Only compute subkernels for common parents

Evaluate total of $O(T\,|s||u|)$ subkernels

Form +ve clique potential functions, sum via junction tree algorithm



Probability Product Kernel

- PPK for 2 Gaussian HMMs with states S & U
- Get |S|x|U| interaction table between all pairs of emissions

$$\Psi(i,j) = \frac{|\Sigma^\dagger|^{1/2}}{|\Sigma_i|^{\beta/2}|\Sigma_j|^{\beta/2}\exp(-\frac{\beta}{2}(\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j{}^T\Sigma_j{}^{-1}\mu_j - \mu^{\dagger T}\Sigma^\dagger\mu^\dagger)}$$

$$\Sigma^\dagger = (\Sigma_i^{-1} + \Sigma_j{}^{-1})^{-1} \text{ and } \mu^\dagger = \Sigma_i^{-1}\mu_i + \Sigma_j{}^{-1}\mu_j.$$

- Then simple pseudo-code:

state prior

transition

$$\Phi(q_0, q_0') = p(q_0)^\beta p'(q_0')^\beta$$

for $t = 1 \ldots T$

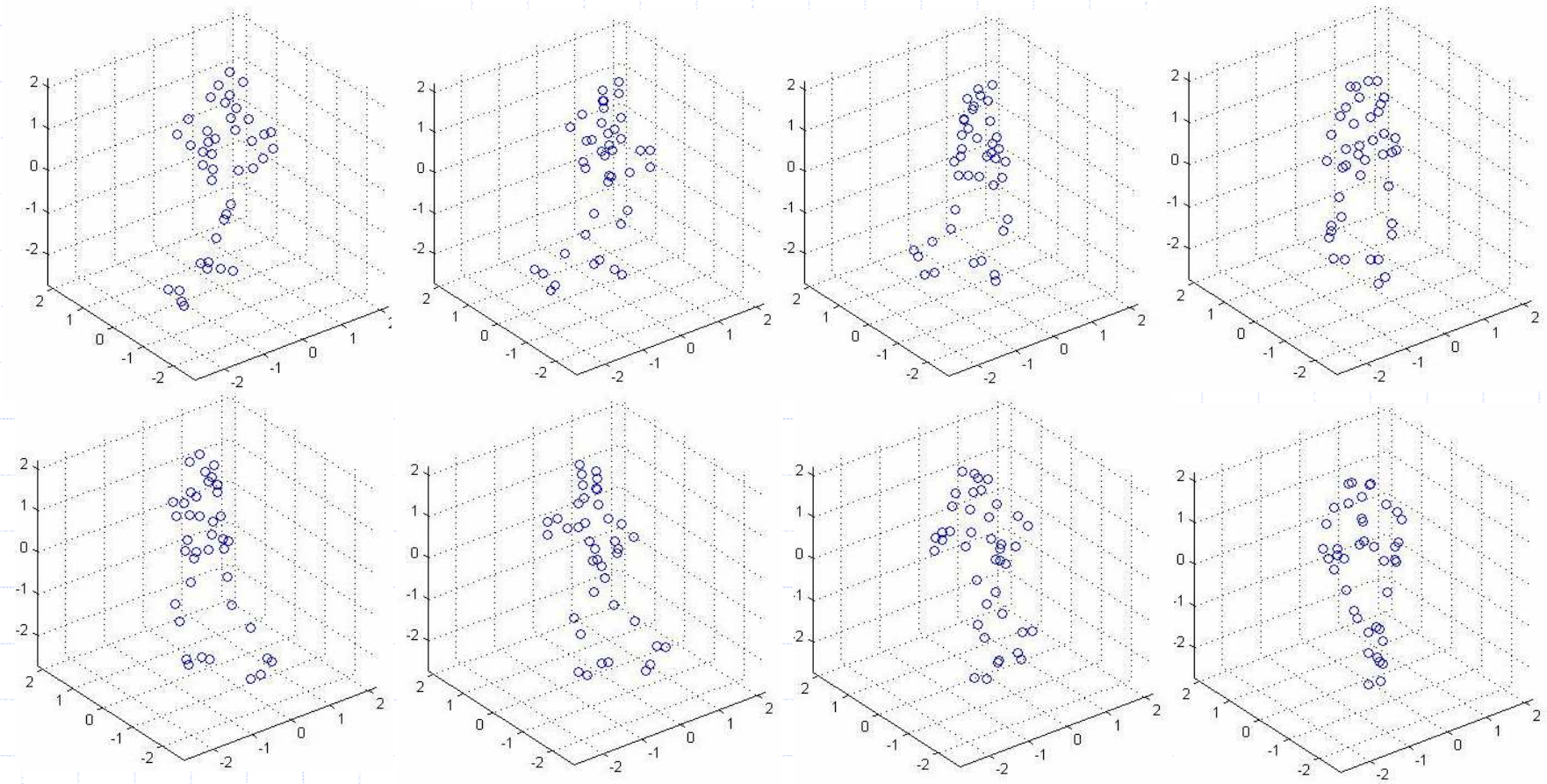
$$\Phi(q_t, q_t') = \sum_{q_{t-1}} \sum_{q_{t-1}'} p(q_t|q_{t-1})^\beta p(q_t'|q_{t-1}')^\beta \Psi(q_{t-1}, q_{t-1}') \Phi(q_{t-1}, q_{t-1}')$$

end

$$\tilde{\mathcal{K}}(\theta, \theta') = \sum_{q_T} \sum_{q_T'} \Phi(q_T, q_T')\Psi(q_T, q_T')$$

Visualizing HMMs

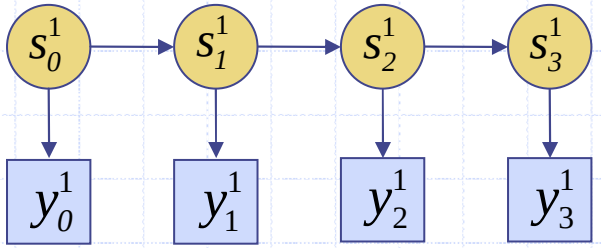
- Rotating walk/run motion data
- Each sequence train a 2-state HMM



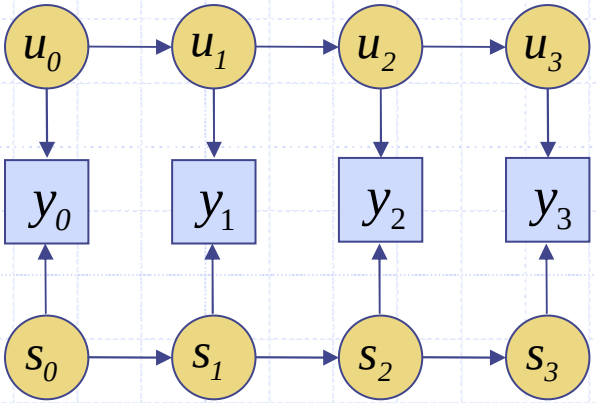
Visualizing HMMs

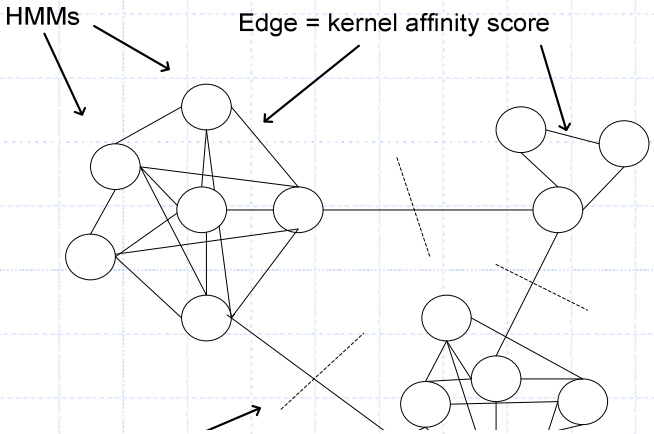
1) For each time series, parametrically learn an HMM

$$\max p\left(y_0^1, y_1^1, y_2^1, y_3^1\right)$$



2) Compute kernel between all pairs of HMMs



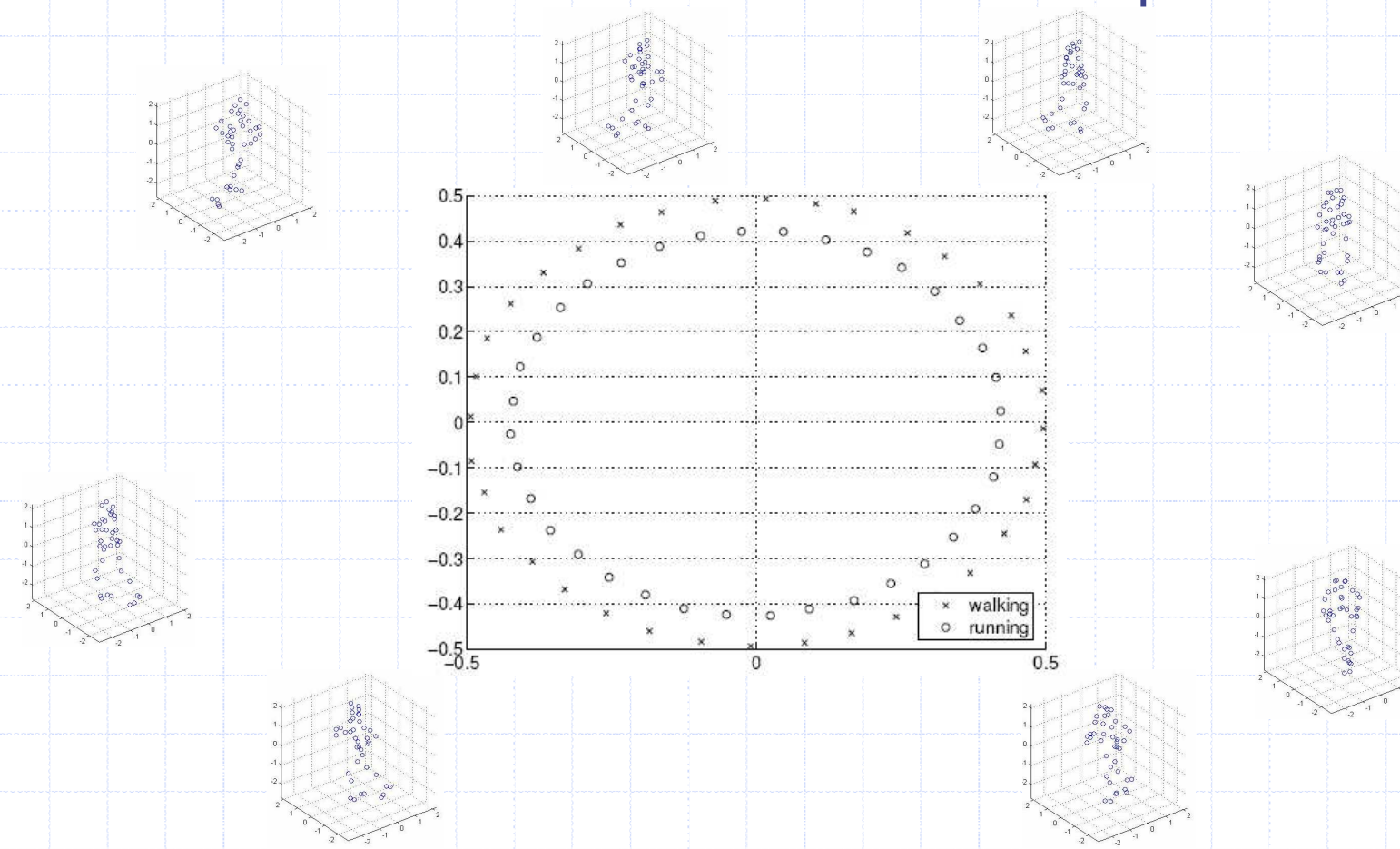


$$k\left(HMM_1, HMM_2\right) = \int p\left(Y \mid HMM_1\right) p\left(Y \mid HMM_2\right) dY$$

3) Run embedding tool on kernel Matrix (MDS, SDE, MVE)

Visualizing HMMs

- The rotated walks and runs appear as two concentric rings when we visualize the kernel matrix from all pairs of HMMs

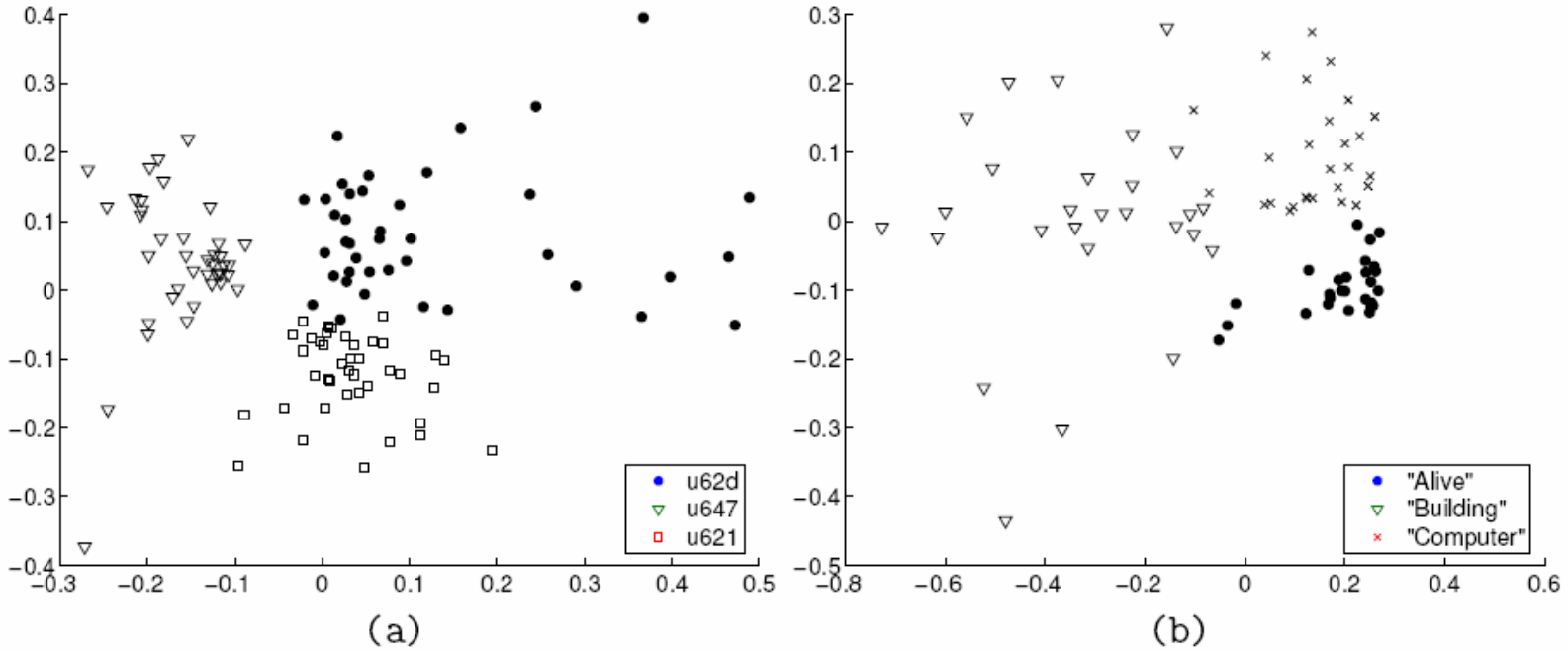


Visualizing HMMs

Embed several HMMs using probability product kernel:

a) Arabic Dataset: each character is a time series datum of spatial coordinates.

b) Sign Language Dataset: each datum is a time series of hand movement coordinates



Sampling Product Kernels

Sampling: approximate kernel

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

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…

Gaussian Product Kernels

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

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

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

k( , )

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

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

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

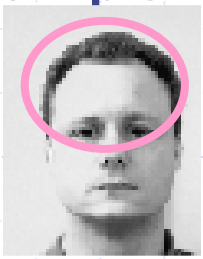
$$k(\chi,\chi') = |\Sigma|^{-\rho/2}\,|\Sigma'|^{-\rho/2}\,|\Sigma^\dagger|^{1/2}\exp\left(-\tfrac{\rho}{2}\mu^T\Sigma^{-1}\mu - \tfrac{\rho}{2}\mu'^T\Sigma'^{-1}\mu' + \tfrac{1}{2}\mu^{\dagger T}\Sigma^\dagger\mu^\dagger\right)$$

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

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

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

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

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

Gives kernelized or augmented Gaussian μ and Σ via Gram

Kernelized Gaussian Product

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

Now have:

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

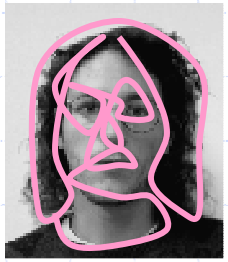
Still invariant to order of pixels

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

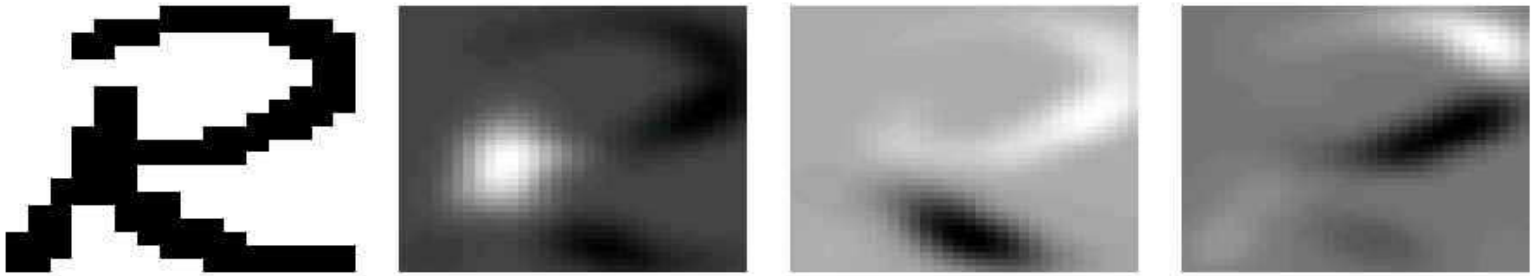
$$\begin{bmatrix} \kappa(x_1,x_1) & \dots & \kappa(x_1,x_N) \\ \vdots & & \vdots \\ \kappa(x_N,x_1) & \dots & \kappa(x_N,x_N) \end{bmatrix} \qquad \begin{bmatrix} \kappa(x_1,x_1) & \dots & \kappa(x_1,x_M) \\ \vdots & & \vdots \\ \kappa(x_M,x_1) & \dots & \kappa(x_M,x_M) \end{bmatrix}$$



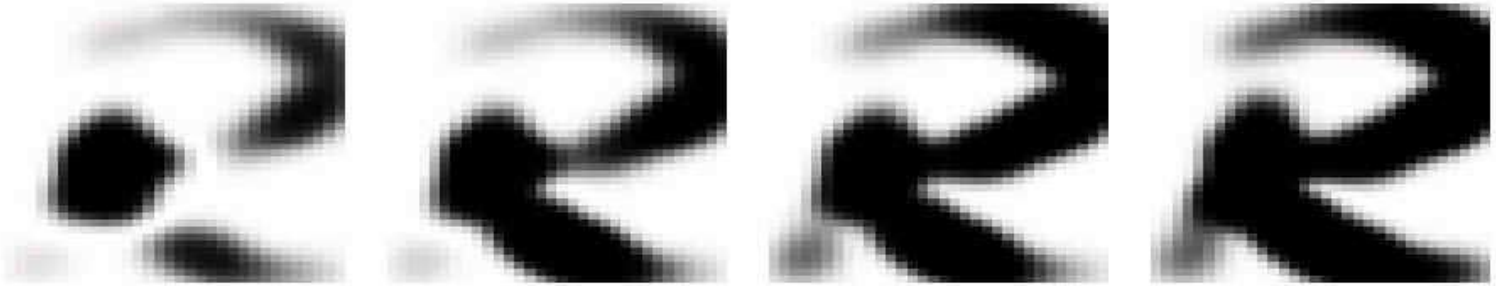
$$k\left(\phi(\chi)\middle\|\phi(\chi')\right) \propto \left|\Sigma\right|^{-\rho/2}\left|\Sigma'\right|^{-\rho/2}\left|\Sigma^{\dagger}\right|^{1/2}\exp\left(-\tfrac{\rho}{2}\mu^T\Sigma^{-1}\mu-\tfrac{\rho}{2}\mu'^T\Sigma'^{-1}\mu'+\tfrac{1}{2}\mu^{\dagger T}\Sigma^{\dagger}\mu^{\dagger}\right)$$

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

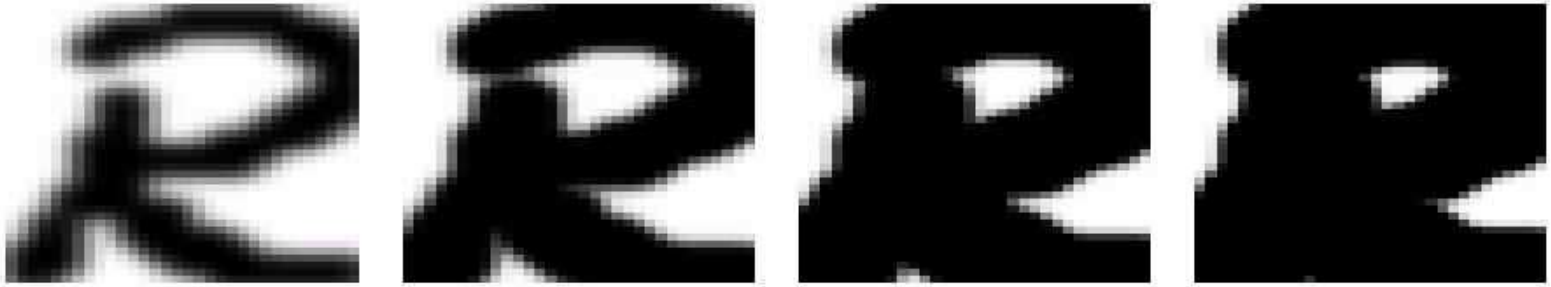
Kernelized Gaussian Product



Letter ‘R’ with 3 KPCA Components of RBF Kernel



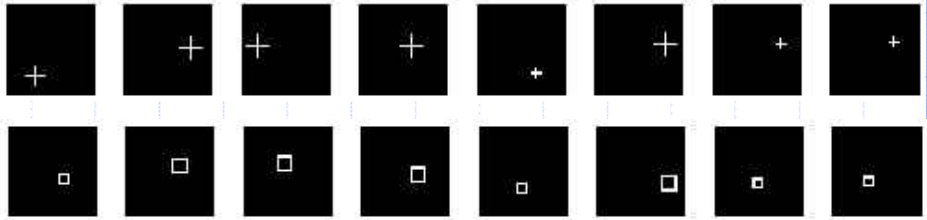
Reconstruction of Letter ‘R’ with 1-4 KPCA with RBF Kernel



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

Kernelized Gaussian Product

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

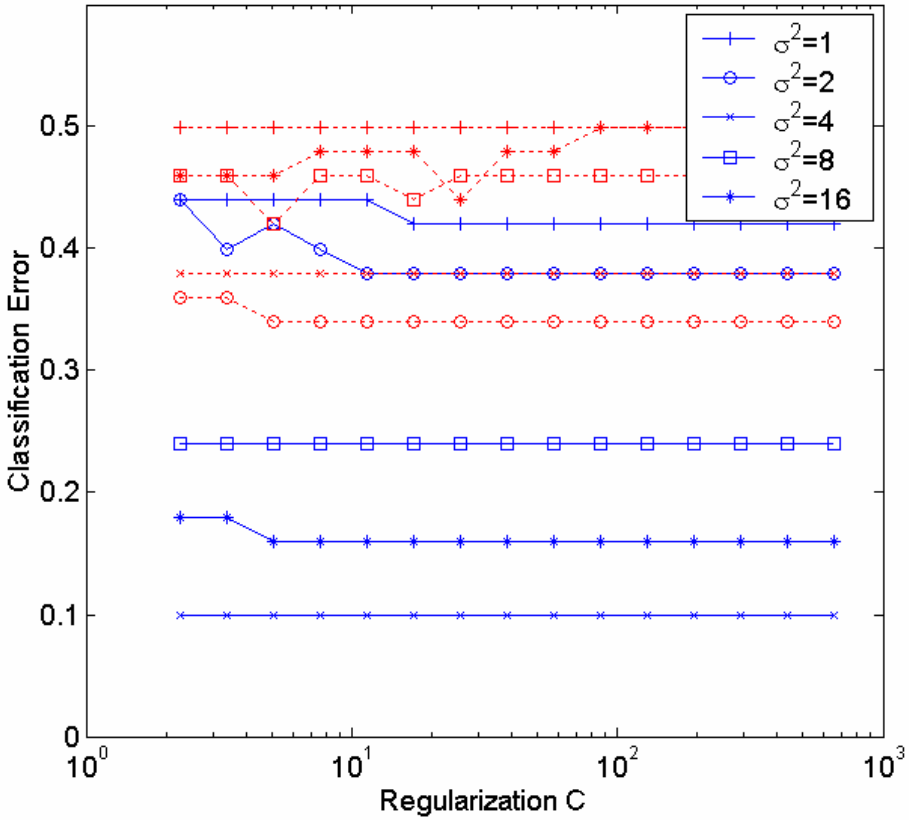


SVM: Train on 50, Test on 5

Fisher for Gaussian is Quadratic Kernel

RBF Kernel (red) 67% accuracy

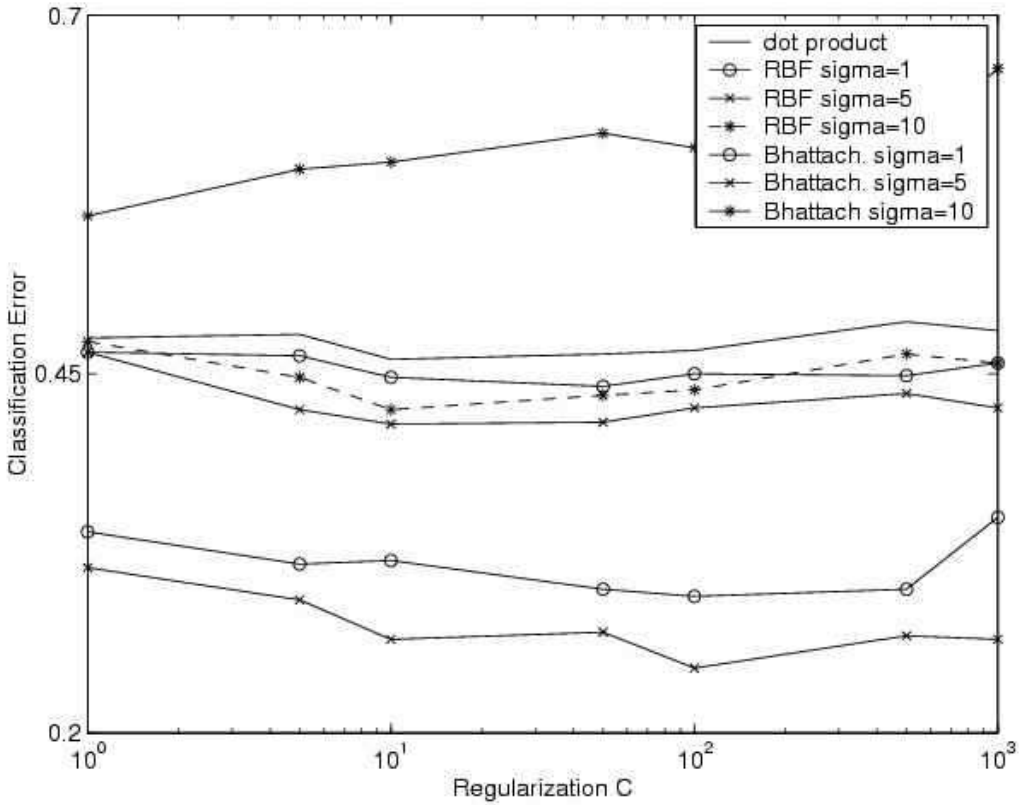
Bhattacharyya (blue) 90% accuracy



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



Regularization & SRM Bounds

- Kernels also motivated from Regularization Theory
- Bound methods (like SRM) minimize some modified risk
- Empirical risk plus some capacity or Regularizer Ω:

$$R_{reg}(f) = R_{emp}(f) + \Omega[f] \qquad where\, R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y)$$

- In structural risk minimization (SRM), Ω is the sqrt VC term

$$R(f) \leq R_{emp}(f) + \sqrt{\frac{h\left(\log\left(\frac{2N}{h}\right)+1\right) - \log\left(\frac{\eta}{4}\right)}{N}}$$

where N is # of points, h is VC dimension and bound holds with probability 1-η

Regularization & Stability Bound

- In stability bounds, Ω is as below

$$R\left(f^*\right) \leq R_{emp}\left(f^*\right) + \beta + \sqrt{\frac{-2\left(N\beta + M\right)\log\left(\eta/2\right)}{N}}$$

The bound holds with probability 1-η
with N training datapoints
with M as an upper bound on the max loss

$$M \geq \max_{x,y,f} \left|L\left(f\left(x\right), y\right)\right|$$

where the f* is returned by a β-stable algorithm.
In other words, the loss for any example (x,y) changes by at most β when we replace one of the training examples with another training example.

Regularization & Rademacher

- In stability bounds, Ω is as below

$$R(f) \leq \inf_{\alpha>0}\left[(1+\alpha)R_{emp}(f)+\left(1+\frac{1}{4\alpha}\right)\left(31r\log^2\frac{b}{r}+50\frac{b\varepsilon}{N}\right)\right]$$

The bound holds with probability 1-η
with N training datapoints
with paramteres r and b that are given
by the so-called Rademacher averages

$$b = E_{S,\sigma}\left[\sup_{f:E\left[L\left(f(x),y\right)\leq r\right]}\sum_{i=1}^{N}\sigma_i L\left(f\left(x_i\right),y\right)\right]$$

Where σ_i are +1 or -1 with 50%-50% probability

Regularization & Function Norm

- In Regularization theory, Ω is most often as below

$$R_{reg}(f) = \frac{1}{N}\sum_{i=1}^{N} L\left(f(x_i), y\right) + \|f\|_H^2$$
$$= \frac{1}{N}\sum_{i=1}^{N} L\left(f(x_i), y\right) + \langle f, f \rangle_H$$
$$= \frac{1}{N}\sum_{i=1}^{N} L\left(\langle f, k_{x_i} \rangle, y\right) + \langle f, f \rangle_H$$

Where line 1à 2 is because of Hilbert space
Where line 2à 3 is because of RKHS property
In other words, the k_x are prototypical functions such that

$$f(x_i) = \langle f, k_{x_i} \rangle$$

Representer Theorem: f will be in the span of k_{x1} … k_{xN}