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: \( k(X, X') = f(X)^T f(X') \)
• Example Kernels:
  
  P-th Order Polynomial Kernel: \( k(x, y) = (x^T y + 1)^p \)

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

• Kernels replace inner products in SVMs, allow nonlinearity:

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

\[
f(x) = \text{sign}\left(\sum_i \alpha_i y_i k(x, x_i) + b\right)
\]

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

\[
K = \begin{bmatrix}
  k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) \\
  k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) \\
  k(x_3, x_1) & k(x_3, x_2) & k(x_3, x_3)
\end{bmatrix}
\]

\[
K_{ij} = k(x_i, x_j)
\]

\( p = \# \text{ of twists in SVM decision boundary} \)
Standard Kernels over Vectors

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

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

• Inputs are two strings: $X = \text{“aardvark”}$
  $X' = \text{“accent”}$

• 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 $X$

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

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

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

• 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

*\( \phi_s(X) = \text{number of times substring } s \text{ appears in } X \)*

*\( \phi_a(\text{aardvark}) = 3 \quad \phi_{ar}(\text{aardvark}) = 1 \)*

*Implicit Kernel* is more efficient

\( k(X,X'):\text{ for each substring } s \text{ of } X, \text{ count how often } s \text{ appears in } X' \)

...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.

Approximate KL by quadratic form:

\[ KL(p || p') \approx \frac{1}{2} (\theta - \theta')^T I_{\theta}^* (\theta - \theta') \]

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

\[ K(\chi, \chi') \approx U_x I_{\theta}^{-1} U_{x'} \]

\[ U_\chi = \nabla_{\theta}^* \log p(\chi | \theta) \]

Fisher Info

\[ I_{\theta}^*(i, j) = \int p(\chi | \theta^*) \frac{\partial \log p(\chi | \theta)}{\partial \theta_i} \frac{\partial \log p(\chi | \theta)}{\partial \theta_j} d\chi \]
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:
\[
JS(p, q) = \frac{1}{2} KL(p, \frac{p+q}{2}) + \frac{1}{2} KL(q, \frac{p+q}{2})
\]

Symmetrized KL:
\[
SKL(p, q) = \frac{1}{2} KL(p, q) + \frac{1}{2} KL(q, p)
\]

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

… 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 $\chi$ and $\chi'$

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

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

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

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 $\rho$)
(continuous)
If $\mu = \chi$ $\mu' = \chi'$ get RBF Kernel
Fisher here is linear

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

Gaussian Covariance: (any $\rho$)

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

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

Bernoulli: (binary)

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

\[ K(\chi, \chi') = \int \left( p(x) \right)^{\rho} \left( p'(x) \right)^{\rho} \, dx \]

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

Multinomial: (discrete)

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

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

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
# Exp. Family Product Kernels

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

\[
p(x | \theta) = \exp \left( A(x) + T(x)^T \theta - K(\theta) \right)
\]

Maximum likelihood is straightforward:

\[
\frac{\partial}{\partial \theta} K(\theta) = \frac{1}{n} \sum_n T(x_n)
\]

All have above form but different \(A(x), T(x), \text{convex } K(\theta)\)

<table>
<thead>
<tr>
<th>Family</th>
<th>(A(X))</th>
<th>(K(\theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian (mean)</td>
<td>(-\frac{1}{2} X^T X - \frac{D}{2} \log(2\pi))</td>
<td>(\frac{1}{2} \theta^T \theta)</td>
</tr>
<tr>
<td>Gaussian (variance)</td>
<td>(-\frac{1}{2} \log(2\pi))</td>
<td>(-\frac{1}{2} \log(\theta))</td>
</tr>
<tr>
<td>Multinomial</td>
<td>(\log(\Gamma(\eta + 1)) - \log(\nu)) (\eta \log(1 + \sum_{d=1}^D \exp(\theta_d)))</td>
<td></td>
</tr>
<tr>
<td>Exponential</td>
<td>0</td>
<td>(- \log(-\theta))</td>
</tr>
<tr>
<td>Gamma</td>
<td>(- \exp(X) - X)</td>
<td>(\log \Gamma(\theta))</td>
</tr>
<tr>
<td>Poisson</td>
<td>(\log(X!))</td>
<td>(\exp(\theta))</td>
</tr>
</tbody>
</table>
Exp. Family Product Kernels

Compute the Bhattacharyya Kernel for the e-family:

\[ p(x | \theta) = \exp \left( A(x) + T(x)^T \theta - K(\theta) \right) \]

Analytic solution for e-family:

\[ K(\chi, \chi') = \int p(x | \theta)^{1/2} p(x | \theta')^{1/2} \, dx \]
\[ = \exp \left( K\left( \frac{1}{2} \theta + \frac{1}{2} \theta' \right) - \frac{1}{2} K\left( \frac{1}{2} \theta \right) - \frac{1}{2} K\left( \theta' \right) \right) \]

Only depends on convex cumulant-generating function \( K(\theta) \)

Meanwhile, Fisher Kernel is always linear in sufficient stats…

\[ U_x = \nabla_{\theta^*} \log p(\chi | \theta) = T(\chi) - \nabla_{\theta^*} K(\theta) \]
\[ U_\chi U_{\chi'}^{-1} = \left( T(\chi) - g \right)^T I_{\theta^*}^{-1} \left( T(\chi') - g \right) \]
Gaussian Product Kernels

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

\[
\left\{ \chi_1, \ldots, \chi_N \right\} \rightarrow p(x) \\
\left\{ \chi_1', \ldots, \chi_N' \right\} \rightarrow p'(x)
\]

\[
\mu = E \left\{ \chi_i \right\} \\
\Sigma = E \left\{ (\chi_i - \mu)(\chi_i - \mu)^T \right\}
\]

\[
\sim N \left( x \mid \mu, \Sigma \right) \\
\sim N \left( x \mid \mu', \Sigma' \right)
\]

\[
K(\chi, \chi') = \left| \Sigma \right|^{-\rho/2} \left| \Sigma' \right|^{-\rho/2} \left| \Sigma^\ast \right|^{1/2} \exp \left( -\frac{\rho}{2} \mu^T \Sigma^{-1} \mu - \frac{\rho}{2} \mu'^T \Sigma'^{-1} \mu' + \frac{1}{2} \mu^T \Sigma^\ast \mu^\ast \right)
\]
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 again when computing Gaussian mean & covariance

\[
\kappa(\chi, \chi') = \phi(\chi)^T \phi(\chi')
\]

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

\[
\kappa(\chi, \chi') = \exp\left(-\frac{1}{2\sigma^2} \|\chi - \chi'\|^2\right)
\]

Compute mini-kernel between each pixel in a given image...
Gives kernelized or augmented Gaussian \( \mu \) and \( \Sigma \) via Gram
Kernelized Gaussian Product

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

Now have:
\[ \mu = E \{ \phi(\chi) \} \quad \Sigma = E \left\{ (\phi(\chi) - \mu)(\phi(\chi) - \mu)^T \right\} \]

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

\[
\begin{bmatrix}
\kappa(x_1,x_1) & \ldots & \kappa(x_1,x_N) \\
\vdots & \ddots & \vdots \\
\kappa(x_N,x_1) & \ldots & \kappa(x_N,x_N)
\end{bmatrix}
\]

\[
\begin{bmatrix}
\kappa(x_1,x_1) & \ldots & \kappa(x_1,x_M) \\
\vdots & \ddots & \vdots \\
\kappa(x_M,x_1) & \ldots & \kappa(x_M,x_M)
\end{bmatrix}
\]

\[
K \left( \phi(\chi) \| \phi(\chi') \right) \propto \left| \Sigma \right|^{-\rho/2} \left| \Sigma' \right|^{-\rho/2} \left| \Sigma^\bullet \right|^{1/2} \exp \left( -\frac{\rho}{2} \mu^T \Sigma^{-1} \mu - \frac{\rho}{2} \mu'^T \Sigma'^{-1} \mu' + \frac{1}{2} \mu^T \Sigma^\bullet \mu^\bullet \right)
\]

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 50

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 $\rho=1$ Expected Likelihood kernel...

\[
\chi \rightarrow p(x) = \sum_{m=1}^{M} p(m)p(x \mid m)
\]

\[
\chi' \rightarrow p'(x) = \sum_{n=1}^{N} p'(n)p'(x \mid n)
\]

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

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

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

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

\[
\begin{array}{c c c c c}
 s^0 & \rightarrow & s^1 & \rightarrow & s^2 & \rightarrow & s^3 & \rightarrow & s^4 \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
 x^0 & \rightarrow & x^1 & \rightarrow & x^2 & \rightarrow & x^3 & \rightarrow & x^4 \\
\end{array}
\]

\[ \# \text{ configs} = |s|^T \]

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

\[ K(\chi, \chi') = \sum_S \sum_U \prod_t p(s_t | s_{t-1}) p'(u_t | u_{t-1}) \int p(x_t | s_t) p'(x_t | u_t) dx_t \]

\[ = \sum_{S,U} \prod_t p(s_t | s_{t-1}) p'(u_t | u_{t-1}) K_{s_t, u_t} \]

\[ = \sum_{S,U} p(s_0) p'(u_0) \psi(s_0, u_0) \prod_t p(s_t | s_{t-1}) p'(u_t | u_{t-1}) \psi(s_t, u_t) \]

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
Sampling Product Kernels

By definition, generative models can:
1) Generate a Sample
2) Compute Likelihood of a Sample

Thus, approximate probability product via sampling:

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

$$K(\chi, \chi') = K(p, p')$$

$$= \frac{\beta}{N} \sum_{x_i \sim p(x)} p'(x_i) + \frac{1-\beta}{N'} \sum_{x_i' \sim p'(x)} p(x_i')$$

Beta controls how much sampling from each distribution...