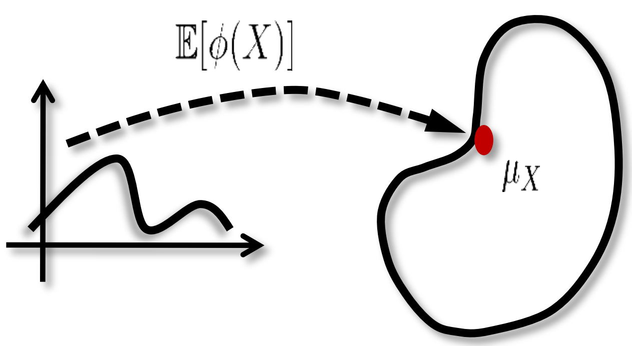
School of Computer Science
Carnegie Mellon

Probabilistic Graphical Models

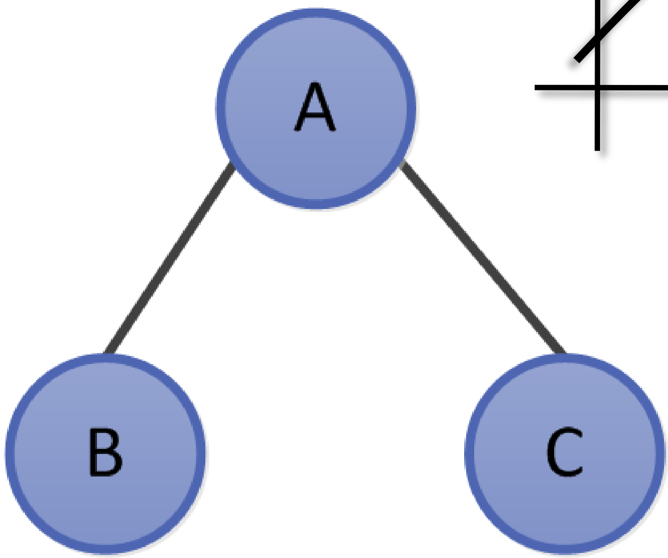
Kernel Graphical Models

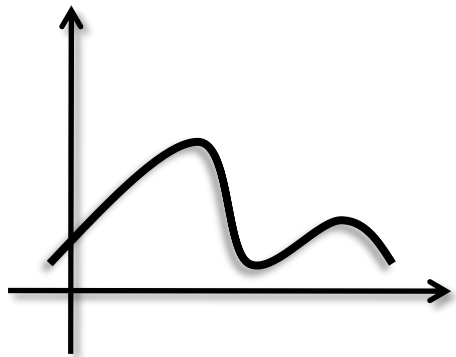
Ankur Parikh

Lecture 20, April 1, 2013



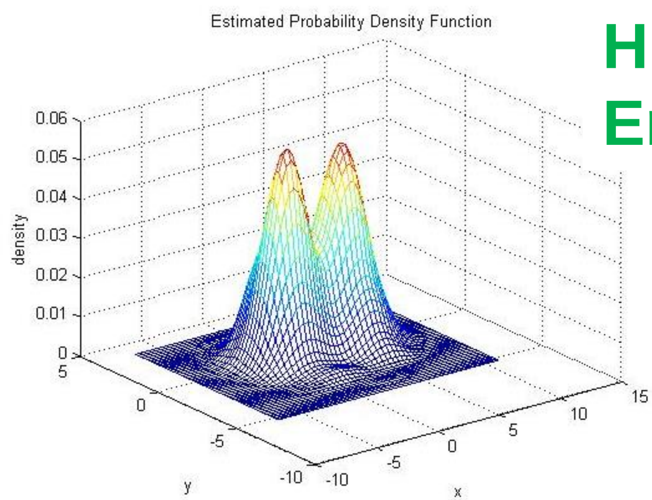
Nonparametric Graphical Models





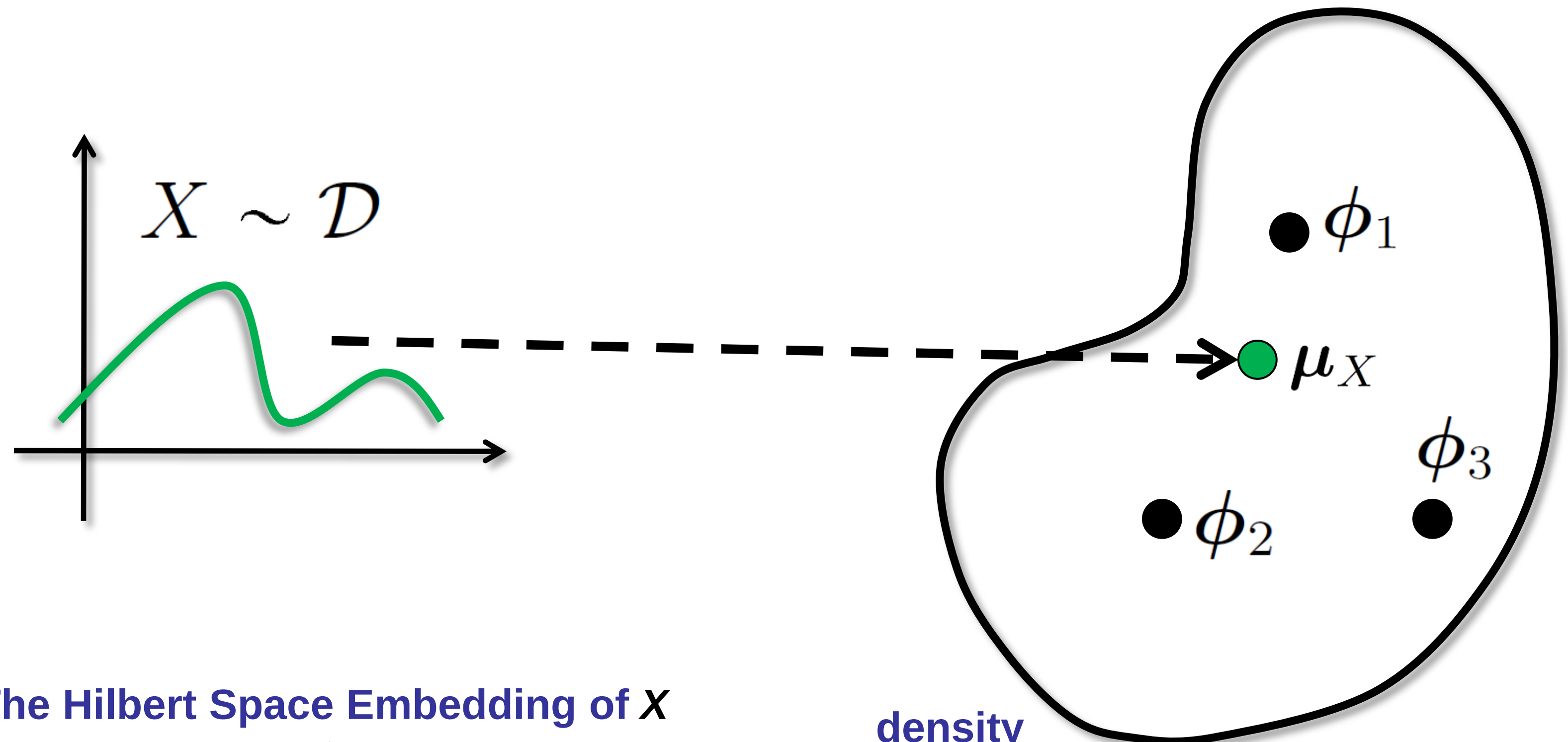
How do we make a conditional probability table out of this?

Hilbert Space Embeddings!!!!!



- How to learn parameters?
- How to perform inference?

Review: Embedding Distribution of One Variable[Smola et al. 2007]



The Hilbert Space Embedding of ***X***

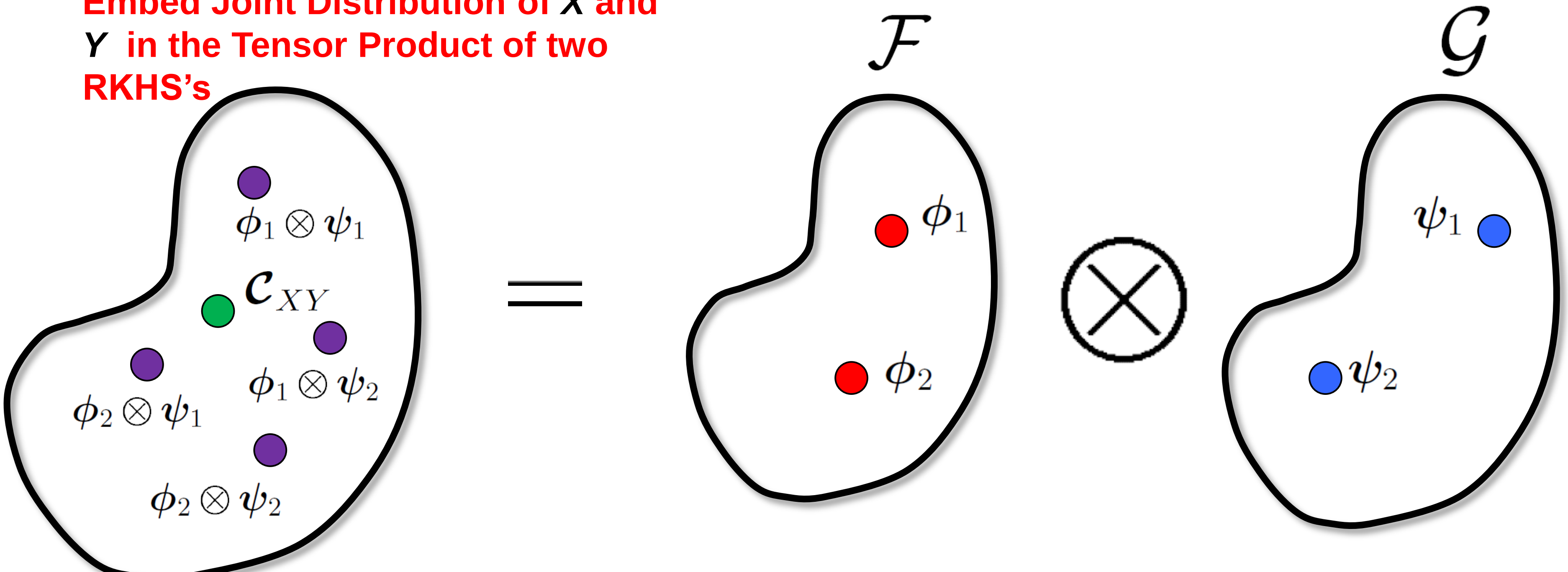
density

$$\boldsymbol{\mu}_X(\cdot) = \mathbb{E}_{X\sim\mathcal{D}}[\phi_X] = \int \boldsymbol{p}_{\mathcal{D}}(X)\phi_X(\cdot)\, dX$$

Review: Cross Covariance Operator [Smola et al. 2007]

$$\mathcal{C}_{XY} = \mathbb{E}_{XY}[\phi_X \otimes \psi_Y]$$

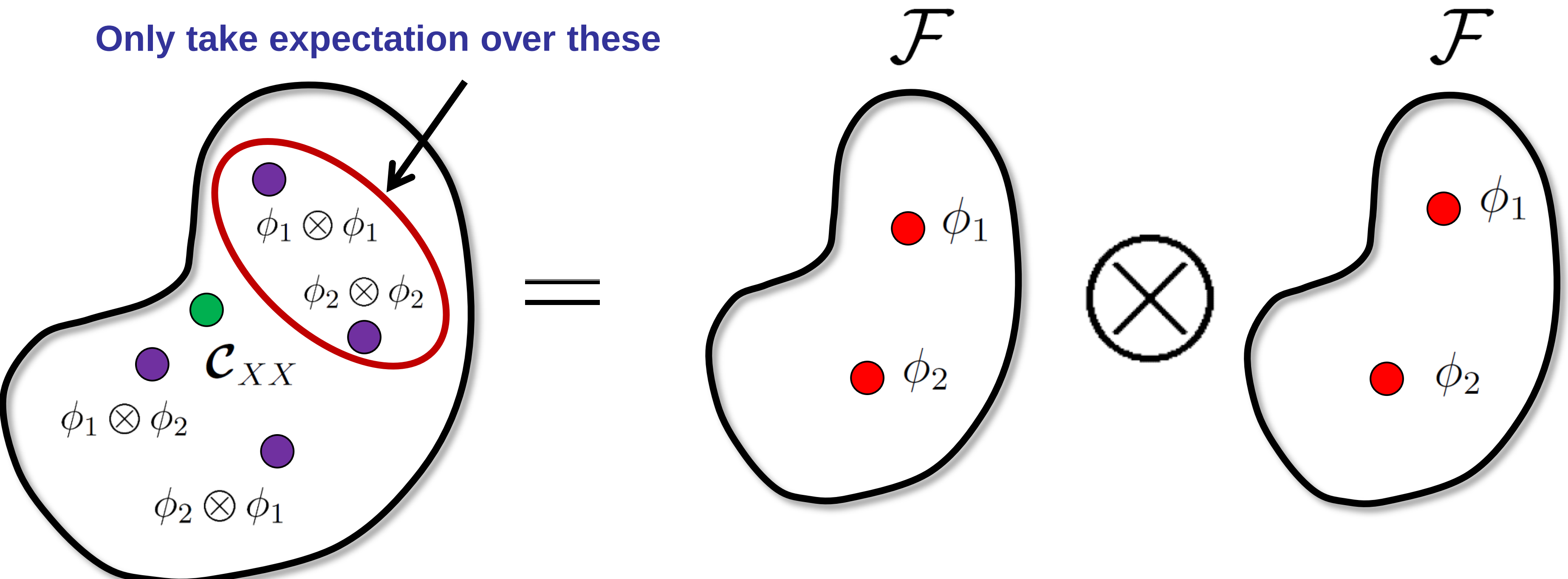
Embed Joint Distribution of *X* and *Y* in the Tensor Product of two RKHS's



Embedding of $\overrightarrow{\mathbb{P}}[X, Y]$

Review: Auto Covariance Operator [Smola et al. 2007]

$$\mathcal{C}_{XX} = \mathbb{E}_X[\phi_X \otimes \phi_X]$$



Embedding of $\mathrm{Diag}(\mathbb{P}[X])$

Review: Conditional Embedding Operator [Song et al. 2009]

- Conditional Embedding Operator:

$$\mathcal{C}_{X|Y} = \mathcal{C}_{XY}\mathcal{C}_{YY}^{-1}$$

- Has Following Property:

$$\mathbb{E}_{X|y}[\phi_X|y] = \mathcal{C}_{X|Y}\phi_y$$

- Analogous to "Slicing" a Conditional Probability Table in the Discrete Case:

$$\overrightarrow{\mathbb{P}}[X|Y=1] = \overrightarrow{\mathbb{P}}[X|Y]\boldsymbol{\delta}_1$$

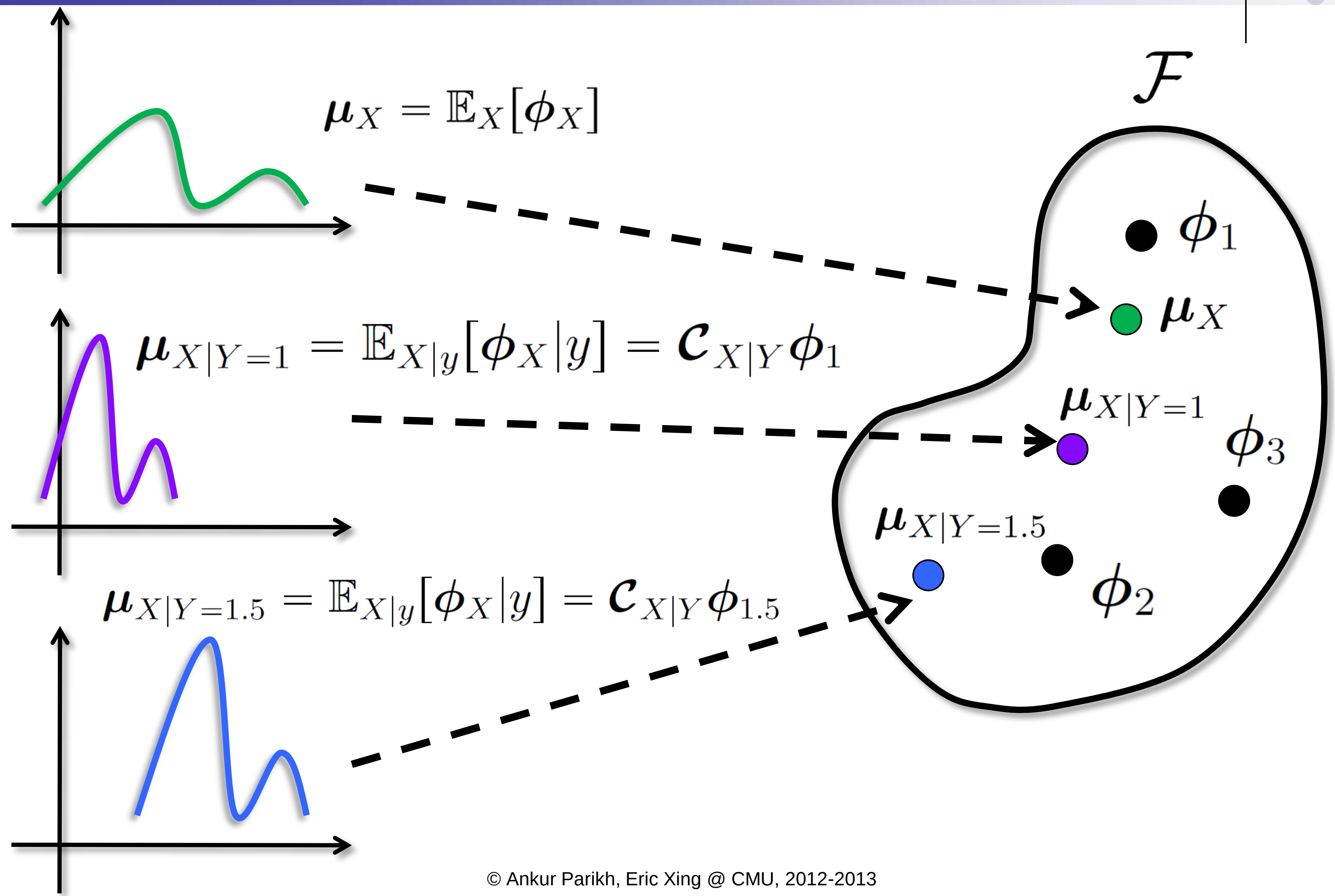
Slicing the Conditional Probability Matrix

$$\mathcal{P}[X]$$

$$\mathcal{P}[X|Y=1] = \mathcal{P}[X|Y]\boldsymbol{\delta}_1$$

$$\mathcal{P}[X|Y=2] = \mathcal{P}[X|Y]\boldsymbol{\delta}_2$$

"Slicing" the Conditional Embedding Operator



Why we Like Hilbert Space Embeddings

We can marginalize and use chain rule in Hilbert Space too!!!

Sum Rule:

$$\mathbb{P}[X] = \int_Y \mathbb{P}[X, Y] = \int_Y \mathbb{P}[X|Y]\mathbb{P}[Y]$$

Chain Rule:

$$\mathbb{P}[X, Y] = \mathbb{P}[X|Y]\mathbb{P}[Y] = \mathbb{P}[Y|X]\mathbb{P}[Y]$$

Sum Rule in RKHS:

$$\boldsymbol{\mu}_X = \mathcal{C}_{X|Y}\boldsymbol{\mu}_Y$$

Chain Rule in RKHS:

$$\mathcal{C}_{YX} = \mathcal{C}_{Y|X}\mathcal{C}_{XX} = \mathcal{C}_{X|Y}\mathcal{C}_{YY}$$

We will prove these now

Sum Rules

- The sum rule can be expressed in two ways:

- First way:

$$\mathbb{P}[X] = \sum_{Y} \mathbb{P}[X, Y]$$

Does not work in RKHS, since there is no "sum" operation for an operator

- Second way:

$$\mathbb{P}[X] = \sum_{Y} \mathbb{P}[X | Y] \mathbb{P}[Y]$$

Works in RKHS!!!

- What is special about the second way? Intuitively, it can be expressed elegantly as matrix multiplication ☺

Sum Rule (Matrix Form)

- Sum Rule

$$\mathbb{P}[X] = \sum_Y \mathbb{P}[X|Y]\mathbb{P}[Y]$$

- Equivalent view using Matrix Algebra

$$\mathcal{P}[X] = \mathcal{P}[X|Y] \times \mathcal{P}[Y]$$

$$\begin{pmatrix} \mathbb{P}[X=0] \\ \mathbb{P}[X=1] \end{pmatrix} = \begin{pmatrix} \mathbb{P}[X=0|Y=0] & \mathbb{P}[X=0|Y=1] \\ \mathbb{P}[X=1|Y=0] & \mathbb{P}[X=1|Y=1] \end{pmatrix} \times \begin{pmatrix} \mathbb{P}[Y=0] \\ \mathbb{P}[Y=1] \end{pmatrix}$$

Important Notation for this Lecture

- We will use the calligraphic P to denote that the probability is being treated as a matrix/vector/tensor

- Probabilities

$$\mathbb{P}[X, Y] = \mathbb{P}[X|Y]\mathbb{P}[Y]$$

- Probability Vectors/Matrices/Tensors

$$\mathcal{P}[X] = \mathcal{P}[X|Y]\mathcal{P}[Y]$$

Chain Rule (Matrix Form)

- Chain Rule

$$\mathbb{P}[X,Y] = \mathbb{P}[X|Y]\mathbb{P}[Y] = \mathbb{P}[Y|X]\mathbb{P}[Y]$$

- Equivalent view using Matrix Algebra

Means on diagonal

$$\mathcal{P}[X,Y] = \mathcal{P}[X|Y] \times \mathcal{P}[\oslash Y]$$

$$\begin{pmatrix} \mathbb{P}[X=0,Y=0] & \mathbb{P}[X=0,Y=1] \\ \mathbb{P}[X=1,Y=0] & \mathbb{P}[X=1,Y=1] \end{pmatrix} =$$

$$\begin{pmatrix} \mathbb{P}[X=0|Y=0] & \mathbb{P}[X=0|Y=1] \\ \mathbb{P}[X=1|Y=0] & \mathbb{P}[X=1|Y=1] \end{pmatrix} \times \begin{pmatrix} \mathbb{P}[Y=0] & 0 \\ 0 & \mathbb{P}[Y=1] \end{pmatrix}$$

- Note how diagonal is used to keep ***Y*** from being marginalized out.

Example

- What about?

$$\mathcal{P}[B|A]\mathcal{P}[\oslash A]\mathcal{P}[C|A]^\top$$

$$\mathcal{P}[B,C]$$

- **Only if B and C are conditionally independent given A!!!**

Different Proof of Matrix Sum Rule with Expectations

- Let's now derive the matrix sum rule differently.

- Let $\boldsymbol{\delta_i}$ denote an indicator vector, that is 1 in the i^{th} position.

$$\boldsymbol{\delta}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \boldsymbol{\delta}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\mathcal{P}[X] = \mathbb{E}_X[\boldsymbol{\delta}_X] = \mathbb{P}[X = 0]\boldsymbol{\delta}_0 + \mathbb{P}[X = 1]\boldsymbol{\delta}_1$$

$$\begin{aligned} \mathcal{P}[X|Y = y] &= \mathbb{E}_{X|Y=y}[\boldsymbol{\delta}_X] \\ &= \mathbb{P}[X = 0|Y = y]\boldsymbol{\delta}_0 + \mathbb{P}[X = 1|Y = y]\boldsymbol{\delta}_1 \end{aligned}$$

Random Variables?

$$\mathcal{P}[X] = \mathbb{E}_X[\delta_X]$$

Remember this is a probability vector.
It is not a random variable.

This is a random vector

$$\boldsymbol{\mu}_X = \mathbb{E}_X[\phi_X]$$

Similarly, this is a function in an RKHS.
It is not a random variable.

This is a random function

Expectation Proof of Matrix Sum Rule Cont.

$$
\begin{aligned}
\mathcal{P}[X|Y]\mathcal{P}[Y] &= \mathcal{P}[X|Y]\mathbb{E}_Y[\boldsymbol{\delta}_Y] \\
&= \mathbb{E}_Y[\mathcal{P}[X|Y]\boldsymbol{\delta}_Y] \\
&= \mathbb{E}_Y[\mathbb{E}_{X|Y}[\boldsymbol{\delta}_X]] \\
&= \mathbb{E}_{XY}[\boldsymbol{\delta}_X] \\
&= \mathcal{P}[X]
\end{aligned}
$$

This is a conditional probability matrix, so it is not a random variable (despite the misleading notation), and thus the Expectation can be pulled out

This is a random variable

Proof of RKHS Sum Rule

- Now apply the same technique to the RKHS Case.

$$\mathcal{C}_{X|Y}\boldsymbol{\mu}_Y$$

$$= \mathcal{C}_{X|Y}\mathbb{E}_Y[\psi_Y]$$

$$= \mathbb{E}_Y[\mathcal{C}_{X|Y}\psi_Y]$$ **Move expectation outside**

$$= \mathbb{E}_Y[\mathbb{E}_{X|Y}[\phi_X|Y]]$$ **Property of conditional embedding**

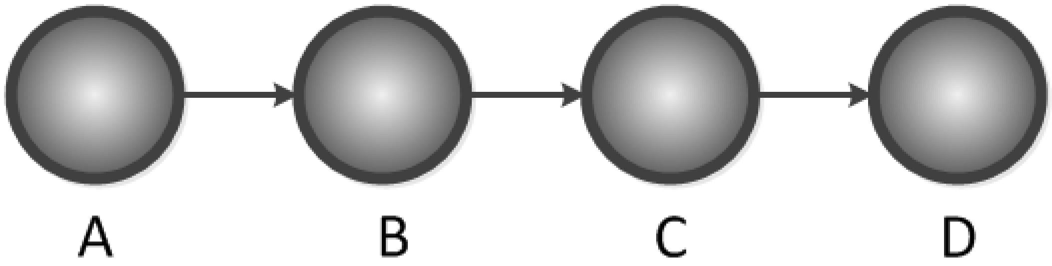
$$= \mathbb{E}_{XY}[\phi_X]$$ **Property of Expectation**

$$= \boldsymbol{\mu}_X$$ **Definition of Mean Map**

Kernel Graphical Models [Song et al. 2010, Song et al. 2011]

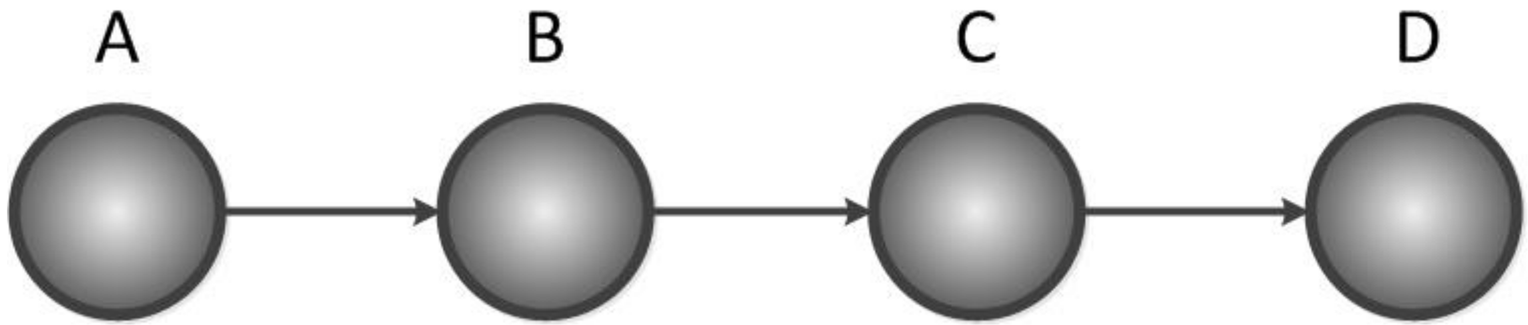
- The idea is to replace the CPTs with RKHS operators/functions.

- Let's do this for a simple example first.

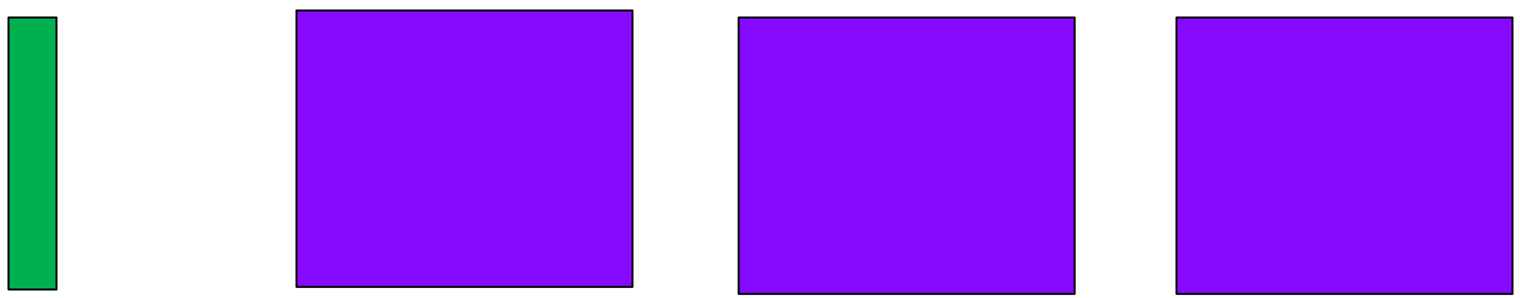


- **We would like to compute** $\mathbb{P}[A = a, D = d]$

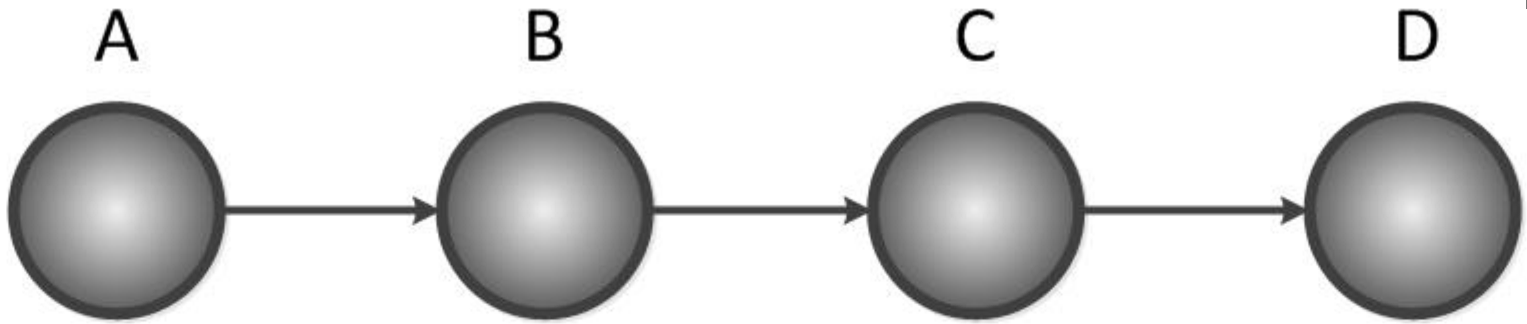
Consider the Discrete Case



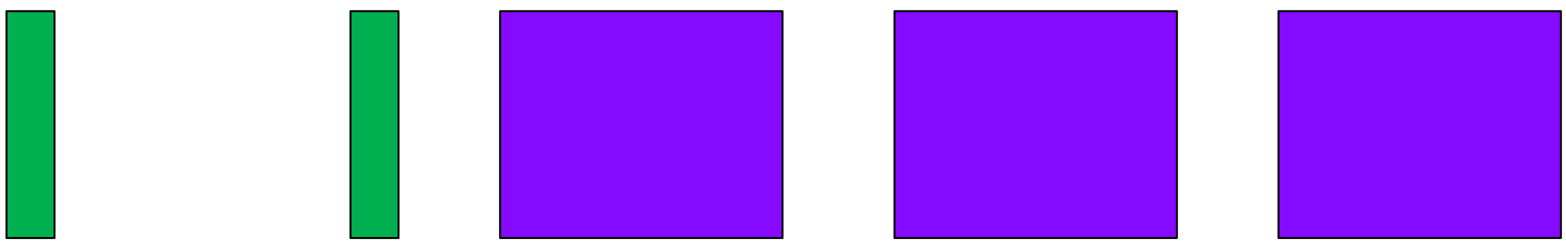
$\mathcal{P}(A)$ $\quad$ $\mathcal{P}(B|A)$ $\quad$ $\mathcal{P}(C|B)$ $\quad$ $\mathcal{P}(D|C)$



Inference as Matrix Multiplication

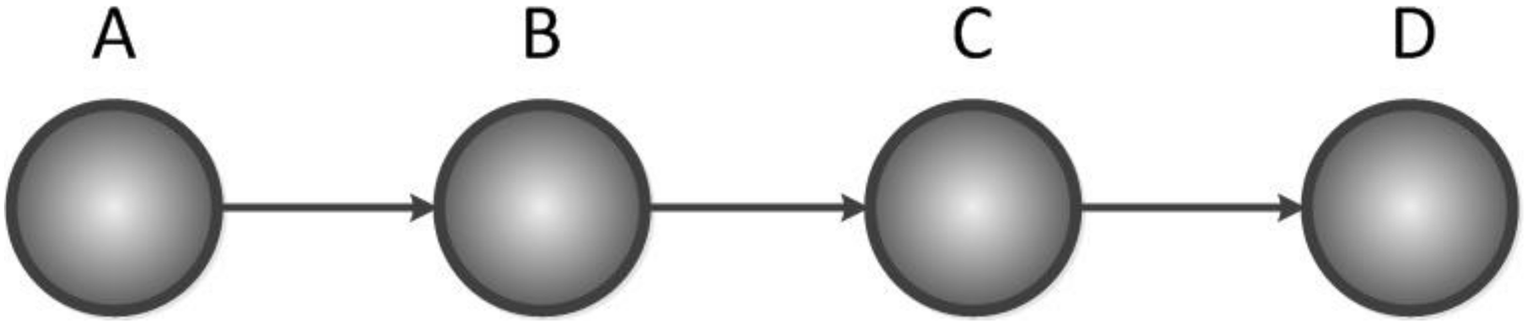


$$\mathcal{P}(D) = \mathcal{P}(A)\mathcal{P}(B|A)^\top\mathcal{P}(C|B)^\top\mathcal{P}(D|C)^\top$$

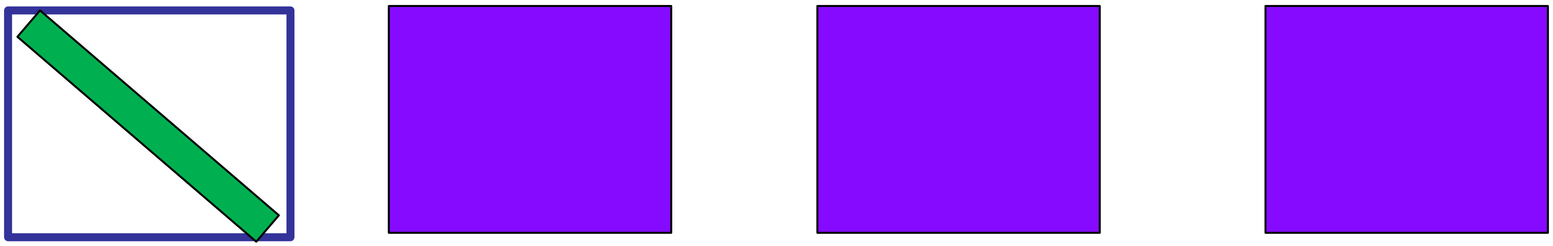


Oops....we accidentally integrated out A

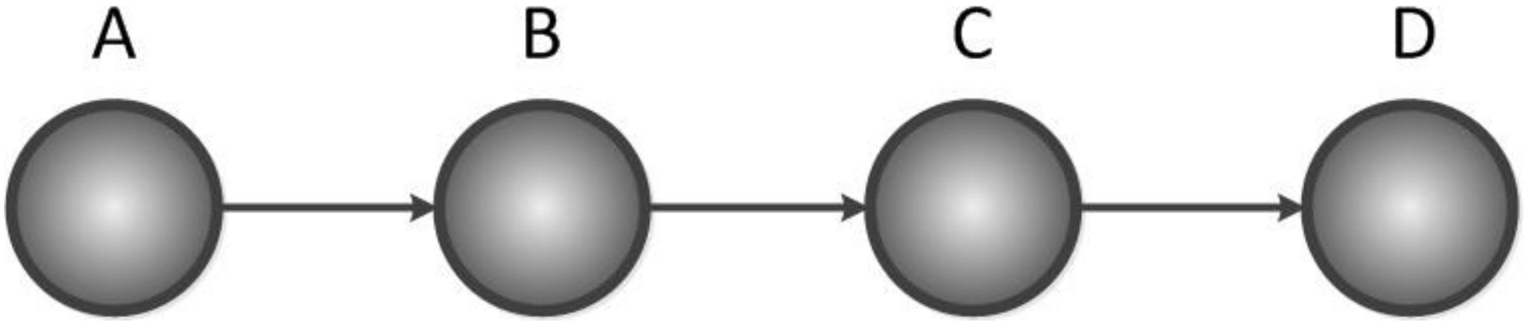
Put A on Diagonal Instead



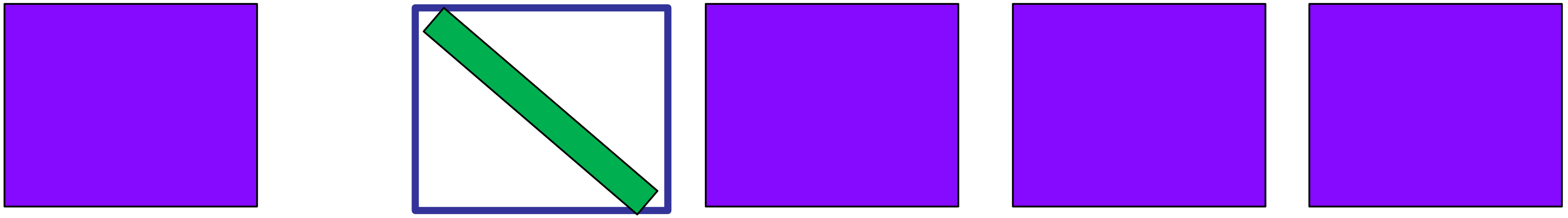
$\mathcal{P}(\emptyset A)$ $\mathcal{P}(B|A)$ $\mathcal{P}(C|B)$ $\mathcal{P}(D|C)$



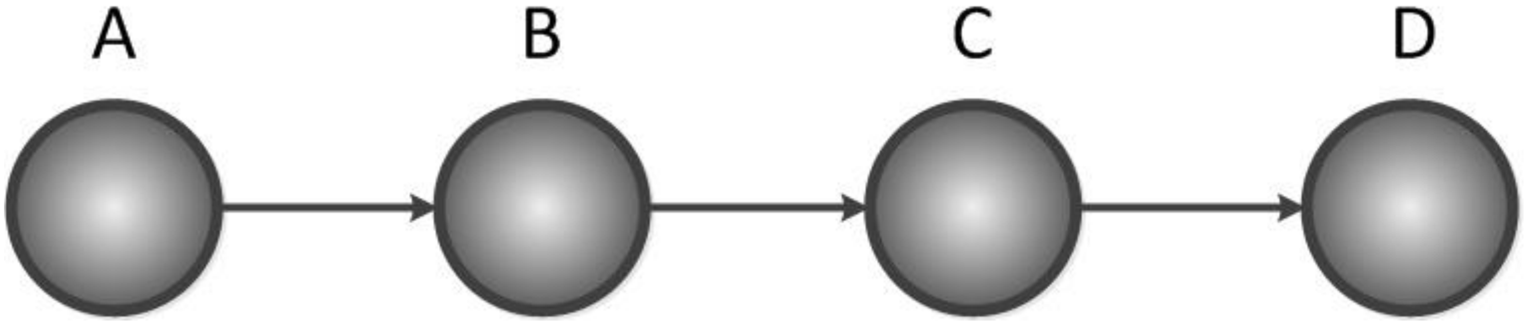
Now it works



$$\mathcal{P}(A, D) = \mathcal{P}(\oslash A)\mathcal{P}(B|A)^\top \mathcal{P}(C|B)^\top \mathcal{P}(D|C)^\top$$



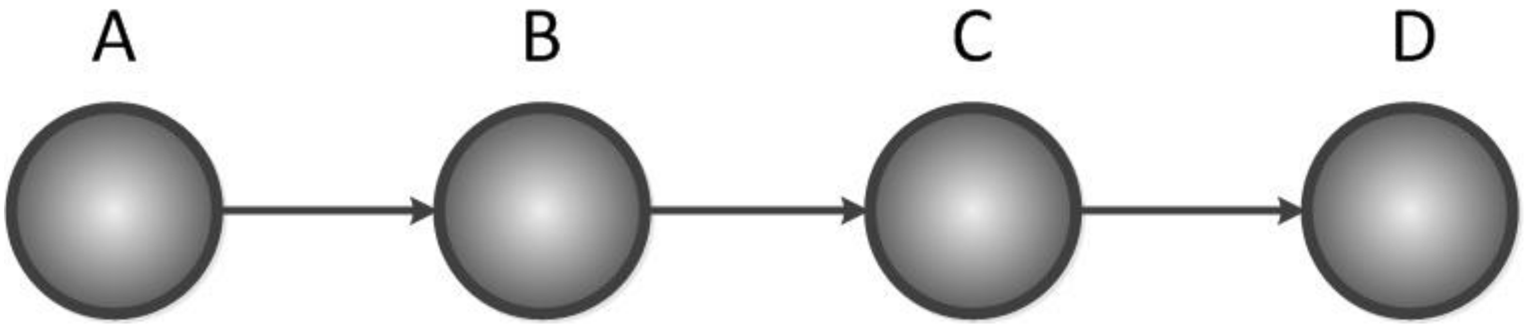
Introducing evidence



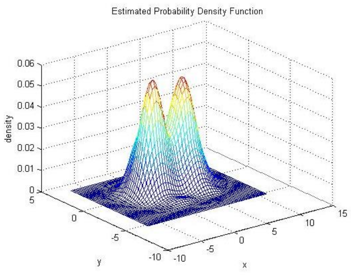
- Introduce evidence with delta vectors

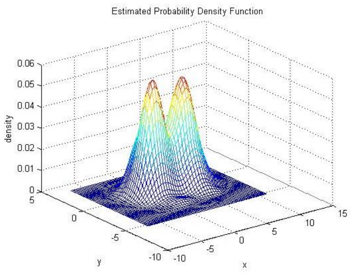
$$\mathcal{P}(A = a, D = d) = \delta_a^\top \mathcal{P}(A, D)\delta_d$$

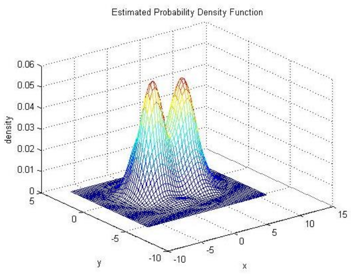
Now with Kernels

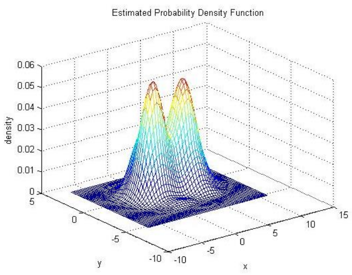


$\mathcal{C}_{AA}$ $\mathcal{C}_{B|A}$ $\mathcal{C}_{C|B}$ $\mathcal{C}_{D|C}$

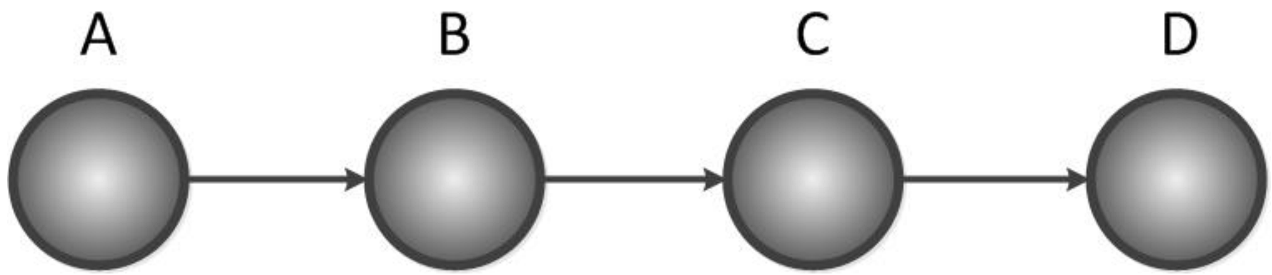








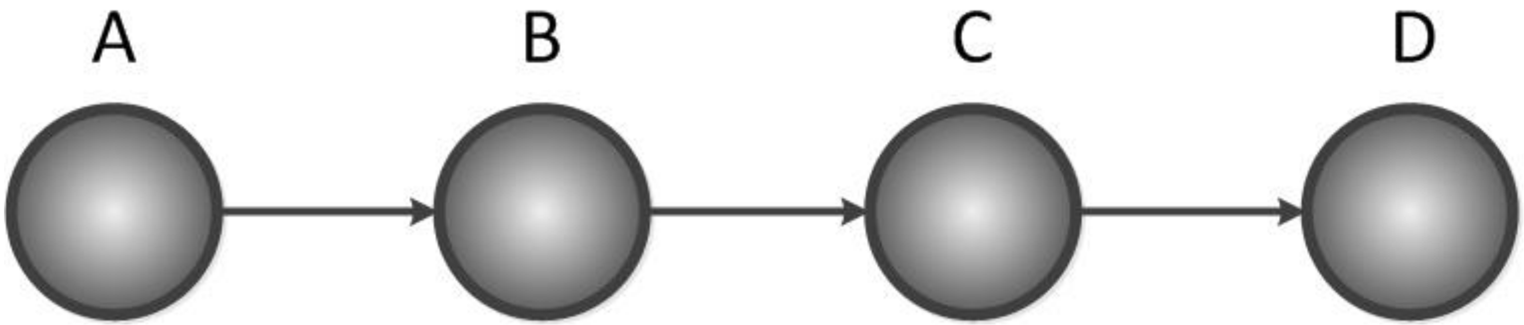
Sum-Product with Kernels



$$\mathcal{C}_{AB} = \mathcal{C}_{AA}\mathcal{C}_{B|A}^{\top}$$

$$\mathcal{C}_{AD} = \mathcal{C}_{AA}\mathcal{C}_{B|A}^{\top}\mathcal{C}_{B|C}^{\top}\mathcal{C}_{C|D}^{\top}$$

Sum-Product with Kernels



$$\text{some number} = \phi_a^\top \mathcal{C}_{A,D} \phi_d$$

What does it mean to evaluate the mean map at a point?

- Consider just evaluating one random variable *X* at a particular evidence value using the Gaussian RBF Kernel:

$$
\begin{aligned}
\langle \boldsymbol{\mu}_X, \boldsymbol{\phi}_{\bar{x}} \rangle &= \mathbb{E}_X \left[\langle \boldsymbol{\phi}_X, \boldsymbol{\phi}_{\bar{x}} \rangle \right] \\
&= \mathbb{E}_X \left[\boldsymbol{K}(X, \bar{x}) \right] \\
&= \mathbb{E}_X \left[\exp\left(\frac{-\|X - \bar{x}\|_2^2}{\sigma^2} \right) \right]
\end{aligned}
$$

- What does this looks like?

Kernel Density Estimation!

- Consider Kernel Density Estimate at point $\bar{x}$:

$$\mathbb{P}_{kde}[X = \bar{x}] \propto \mathbb{E}\left[\exp\left(\frac{-\|X - \bar{x}\|_2^2}{\sigma^2}\right)\right]$$

- And its empirical estimate:

$$\widehat{\mathbb{P}}_{kde}[X = \bar{x}] \propto \frac{1}{N}\sum_{n=1}^{N}\exp\left(-\frac{\|X - \bar{x}\|_2^2}{\sigma^2}\right)$$

- So evaluating the mean map at a point is like an unnormalized kernel density estimate. To find the "MAP" assignment, we can evaluate on a grid of points, and then pick the one with the highest value.

Multiple Variables

- Kernel Density Estimation with Gaussian RBF Kernel in Multiple Variables is:

$$\mathbb{P}_{kde}[\boldsymbol{X}_{1:\mathcal{O}} = \bar{x}_{1:\mathcal{O}}] \propto \mathbb{E}\left[\prod_{o=1}^{\mathcal{O}} \exp\left(-\frac{\|X_O - \bar{x}_o\|_2^2}{\sigma^2}\right)\right]$$

- Like evaluating a "Huge" Covariance Operator using Gaussian RBF Kernel (without normalization):

$$\left\langle \mathcal{C}_{X_1,\ldots,X_{\mathcal{O}}}, \phi_{\bar{x}_1} \otimes \phi_{\bar{x}_2} \cdots \otimes \phi_{\bar{x}_{\mathcal{O}}} \right\rangle$$

What is the problem with this?

- The empirical estimate is very inaccurate because of curse of dimensionality

$$\widehat{\mathbb{P}}_{kde}[\boldsymbol{X}_{1:\mathcal{O}} = \bar{x}_{1:\mathcal{O}}] \propto \frac{1}{N} \sum_{n=1}^{N} \prod_{o=1}^{\mathcal{O}} \exp\left(- \frac{\|X_O^{(n)} - \bar{x}_o\|_2^2}{\sigma^2} \right)$$

- Empirically computing the "huge" covariance operator will have the same problem.

- But then what is the point of Hilbert Space Embeddings?

We can factorize the "Huge" Covariance Operator

- Hilbert Space Embeddings allow us to factorize the huge covariance operator using the graphical model structure that kernel density estimation does not do.

$$\langle \mathcal{C}_{X_1,\ldots,X_{\mathcal{O}}}, \phi_{\bar{x}_1} \otimes \phi_{\bar{x}_2} \cdots \otimes \phi_{\bar{x}_{\mathcal{O}}} \rangle$$

Factorizes into smaller covariance/conditional embedding operators using the graphical model that are more efficient to estimate.

$$\mathcal{C}_{AA} \qquad \mathcal{C}_{B|A} \qquad \mathcal{C}_{C|B} \qquad \mathcal{C}_{D|C}$$

Kernel Graphical Models: The Overall Picture

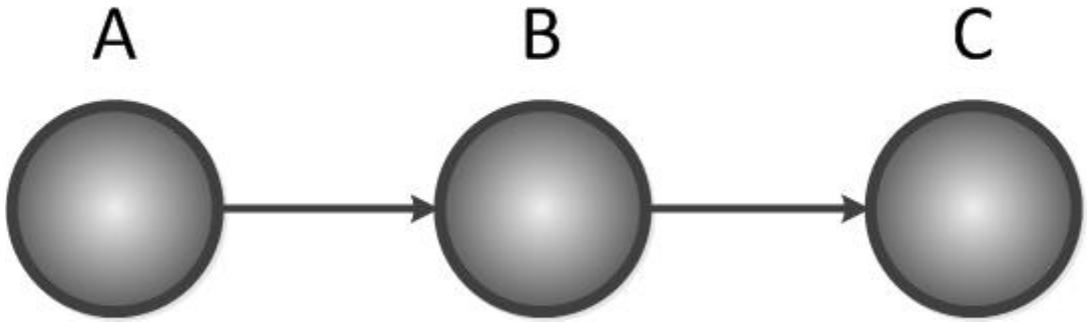
Naïve way to represent joint distribution of discrete variables is to store and manipulate a "huge" probability table.

Discrete Graphical Models allow us to factorize the "huge" joint distribution table into smaller factors.

Naïve way to represent joint distribution for many continuous variables is to use multivariate kernel density estimation.

Kernel Graphical Models allow us to factorize joint distributions of continuous variables into smaller factors.

Consider an Even Simpler Graphical Model



$\mathcal{C}_{AA}$ $\mathcal{C}_{B|A}$ $\mathcal{C}_{C|B}$

We are going to show how to estimate these operators from data.

The Kernel Matrix

$$\boldsymbol{K}_{XX} = \begin{array}{cc} N & \\ \begin{bmatrix} \langle \phi_{x_1}, \phi_{x_1} \rangle & \cdots & \langle \phi_{x_1}, \phi_{x_N} \rangle \\ \vdots & \ddots & \vdots \\ \langle \phi_{x_N}, \phi_{x_1} \rangle & \cdots & \langle \phi_{x_N}, \phi_{x_N} \rangle \end{bmatrix} & N \end{array}$$

Empirical Estimate Auto Covariance

$$\mathcal{C}_{XX} = \mathbb{E}_X[\phi_X \otimes \phi_X]$$

$$\widehat{\mathcal{C}}_{XX} = \frac{1}{N}\sum_{n=1}^{N} \phi_{x_n} \otimes \phi_{x_n}$$

$$\widehat{\mathcal{C}}_{XX} = \frac{1}{N}\Phi_X \Phi_X^\top$$

Defined on next slide

Conceptually,

$$\Phi_X := \overbrace{\begin{bmatrix} \phi_{x_1} & \phi_{x_2} & \cdots & \phi_{x_n} \end{bmatrix}}^{N} \; \infty$$

Conceptually,

$$\sum_{n=1}^{N} v_i \phi_{x_n} = \overset{\Phi_X}{\begin{bmatrix} \Uparrow & \Uparrow & & \Uparrow \\ \phi_{x_1} & \phi_{x_2} & \cdots & \phi_{x_n} \\ \Downarrow & \Downarrow & & \Downarrow \end{bmatrix}} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_N \end{pmatrix}$$

Conceptually,

$$\begin{pmatrix} \phi_{x_1}^\top \boldsymbol{f} \\ \phi_{x_2}^\top \boldsymbol{f} \\ \cdots \\ \phi_{x_n}^\top \boldsymbol{f} \end{pmatrix} = \overset{\Phi_X^\top}{\begin{bmatrix} \longleftarrow & \phi_{x_1} & \longrightarrow \\ \longleftarrow & \phi_{x_2} & \longrightarrow \\ & \cdots & \\ \longleftarrow & \phi_{x_n} & \longrightarrow \end{bmatrix}} \boldsymbol{f}$$

Rigorously,

$\mathbf{\Phi}_X$ is an operator that maps vectors in $\mathbb{R}^N$ to functions in $\mathcal{F}$

such that: $$\sum_{n=1}^{N} \boldsymbol{v}_i \phi_{x_n} = \mathbf{\Phi}_X \boldsymbol{v}$$

Its adjoint (transpose) $\mathbf{\Phi}_X^\top$ can then be derived to be:

$$\begin{pmatrix} \langle \phi_{x_1}, \boldsymbol{f} \rangle \\ \langle \phi_{x_2}^\top, \boldsymbol{f} \rangle \\ \cdots \\ \langle \phi_{x_n}^\top, \boldsymbol{f} \rangle \end{pmatrix} = \mathbf{\Phi}_X^\top \boldsymbol{f}$$

Empirical Estimate Cross Covariance

$$\mathcal{C}_{YX} = \mathbb{E}[\phi_Y \otimes \phi_X]$$

$$\widehat{\mathcal{C}}_{YX} = \frac{1}{N} \sum_{n=1}^{N} \phi_{y_n} \otimes \phi_{x_n}$$

$$\widehat{\mathcal{C}}_{YX} = \frac{1}{N} \mathbf{\Phi}_Y \mathbf{\Phi}_X^\top$$

Getting the Kernel Matrix

- It can then be shown that,

$$\mathbf{\Phi}_X^\top \mathbf{\Phi}_X = \boldsymbol{K}_{XX} \qquad\qquad \boldsymbol{K}_{XX}(i,j) := \langle \phi_{x_i}, \phi_{x_j} \rangle$$

- This is finite and easy to compute!! ☺

- However, note that the estimates of the covariance operators are **not** finite since:

$$\widehat{\mathcal{C}}_{XX} = \frac{1}{N} \mathbf{\Phi}_X \mathbf{\Phi}_X^\top$$

Intuition 1: Why the Kernel Trick works

$$\hat{\mathcal{C}}_{XX} = \frac{1}{N}\sum_{n=1}^{N} \phi_{x_n} \otimes \phi_{x_n}$$

$$\hat{\mathcal{C}}_{XX} = \frac{1}{N}\Phi_X \Phi_X^\top$$

This operator is infinite dimensional but it has at most rank N

$$\mathbf{\Phi}_X^\top \mathbf{\Phi}_X = \boldsymbol{K}_{XX}$$

The kernel matrix is N by N, and thus the kernel trick is exploiting the low rank structure

Empirical Estimate of Conditional Embedding Operator

$$\widehat{\mathcal{C}}_{Y|X} = \widehat{\mathcal{C}}_{YX}\widehat{\mathcal{C}}_{XX}^{-1} \quad \textbf{?}$$

Sort of……

We need to regularize so that this is invertible

$\widehat{\mathcal{C}}_{YX}$ $\widehat{\mathcal{C}}_{XX}$ **regularizer**

$$\widehat{\mathcal{C}}_{Y|X} = \Phi_Y \Phi_X^\top \left(\frac{1}{N}\Phi_X \Phi_X^\top + \lambda I\right)^{-1}$$

Return of Matrix Inversion Lemma

- Believe it or not, the matrix inversion lemma works for linear, bounded operators too.

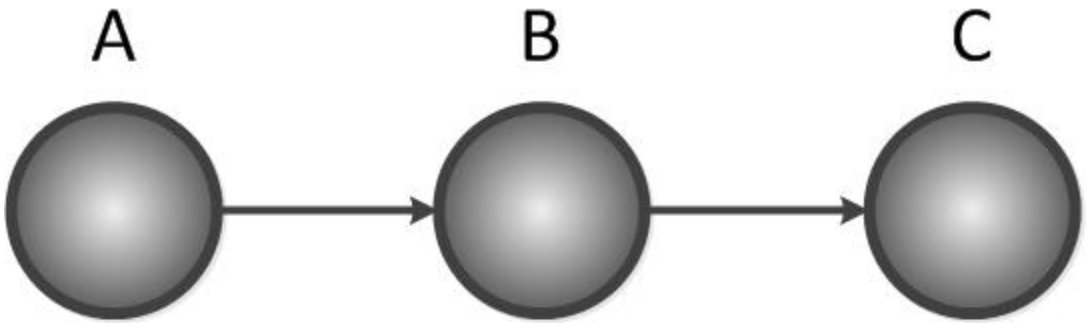
$$(\boldsymbol{E} - \boldsymbol{F}\boldsymbol{H}^{-1}\boldsymbol{G})^{-1} = \boldsymbol{E}^{-1} + \boldsymbol{E}^{-1}\boldsymbol{F}(\boldsymbol{H} - \boldsymbol{G}\boldsymbol{E}^{-1}\boldsymbol{F})^{-1}\boldsymbol{G}\boldsymbol{E}^{-1}$$

- Using it we get,

$$\widehat{\mathcal{C}}_{Y|X} = \Phi_Y(\Phi_X^\top \Phi_X + \lambda N \boldsymbol{I})^{-1}\Phi_X^\top$$

$$\widehat{\mathcal{C}}_{Y|X} = \Phi_Y(\boldsymbol{K}_{XX} + \lambda N \boldsymbol{I})^{-1}\Phi_X^\top$$

But Our estimates are still Infinite….



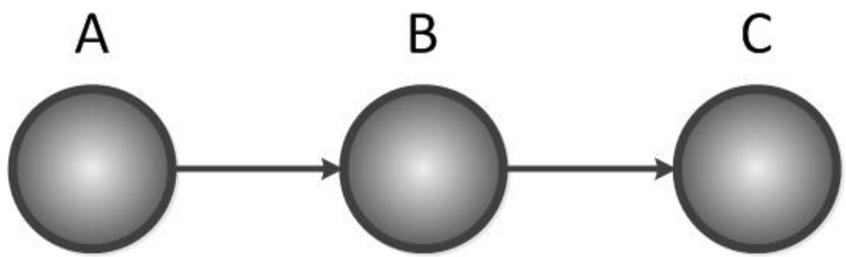
$$\widehat{\mathcal{C}}_{AA} = \frac{1}{N}\Phi_A\Phi_A^\top$$

$$\widehat{\mathcal{C}}_{B|A} = \Phi_B(\boldsymbol{K}_{AA} + \lambda N\boldsymbol{I})^{-1}\Phi_A^\top$$

$$\widehat{\mathcal{C}}_{C|B} = \Phi_C(\boldsymbol{K}_{BB} + \lambda N\boldsymbol{I})^{-1}\Phi_B^\top$$

Lets do inference and see what happens.

Running Inference



$$\widehat{\mathcal{C}}_{AC} = \widehat{\mathcal{C}}_{AA} \widehat{\mathcal{C}}_{B|A}^{\top} \widehat{\mathcal{C}}_{C|B}^{\top}$$

$$\widehat{\mathcal{C}}_{AC} = \frac{1}{N} \Phi_A \underbrace{\boxed{\Phi_A^{\top} \Phi_A}}_{\boldsymbol{K}_{AA}} (\boldsymbol{K}_{AA} + \lambda N \boldsymbol{I})^{-1} \underbrace{\boxed{\Phi_B^{\top} \Phi_B}}_{\boldsymbol{K}_{BB}} (\boldsymbol{K}_{BB} + \lambda N \boldsymbol{I})^{-1} \Phi_C^{\top}$$

Incorporating the Evidence

$$\phi_a^\top \widehat{\mathcal{C}}_{AC}\phi_c =$$

$$\frac{1}{N}\boxed{\phi_a^\top \boldsymbol{\Phi}_A} \boldsymbol{K}_{AA}(\boldsymbol{K}_{AA} + \lambda N \boldsymbol{I})^{-1} \times$$

$$\boldsymbol{K}_{BB}(\boldsymbol{K}_{BB} + \lambda N \boldsymbol{I})^{-1} \boxed{\boldsymbol{\Phi}_C^\top \phi_c}$$

$$\boldsymbol{K}_{AA}(1:N, a) \qquad \boldsymbol{K}_{CC}(1:N, c)$$

Reparameterize the Model

Finite!!!!!

A

$$\widehat{\mathcal{C}}_{AA} \Rightarrow \widehat{\mathcal{D}}_A = \frac{1}{N}\boldsymbol{K}_{AA}$$

B

$$\widehat{\mathcal{C}}_{B|A} \Rightarrow \widehat{\mathcal{D}}_B = (\boldsymbol{K}_{AA} + \lambda N \boldsymbol{I})^{-1}\boldsymbol{K}_{BB}$$

C

$$\widehat{\mathcal{C}}_{C|B} \Rightarrow \widehat{\mathcal{D}}_C = (\boldsymbol{K}_{BB} + \lambda N \boldsymbol{I})^{-1}$$

Evidence:

$$\phi_a \quad \phi_c \Rightarrow \boldsymbol{K}_{AA}(1:N, a) \quad \boldsymbol{K}_{CC}(1:N, c)$$

Intuition 2: Why the Kernel Trick Works

$$\begin{array}{c}
N \\
\boldsymbol{K}_{XX} = \begin{bmatrix} \langle \phi_{x_1}, \phi_{x_1} \rangle & \cdots & \langle \phi_{x_n}, \phi_{x_n} \rangle \\ \vdots & \ddots & \vdots \\ \langle \phi_{x_n}, \phi_{x_1} \rangle & \cdots & \langle \phi_{x_n}, \phi_{x_n} \rangle \end{bmatrix} \quad N
\end{array}$$

Intuition 2: Why the Kernel Trick Works

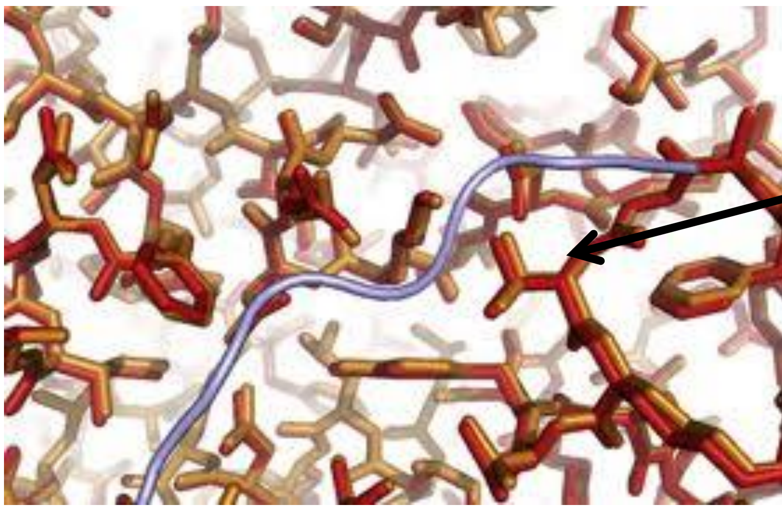
Evaluating a feature function at the N data points!!!

$$\boldsymbol{K}_{XX} = \begin{bmatrix} \phi_{x_1}(x_1) & \cdots & \phi_{x_1}(x_N) \\ \vdots & \ddots & \vdots \\ \phi_{x_N}(x_1) & \cdots & \phi_{x_N}(x_N) \end{bmatrix}$$

Intuition 2: Why the Kernel Trick Works

- Generally people interpret the kernel matrix to be a similarity matrix.
- However, we can also view each row of the kernel matrix as evaluating a function at the N data points.
- Although the function may be continuous and not easily represented analytically, we only really care about what its value is on the N data points.
- Thus, when we only have a finite amount of data, the computation should be inherently finite.

Protein Sidechains

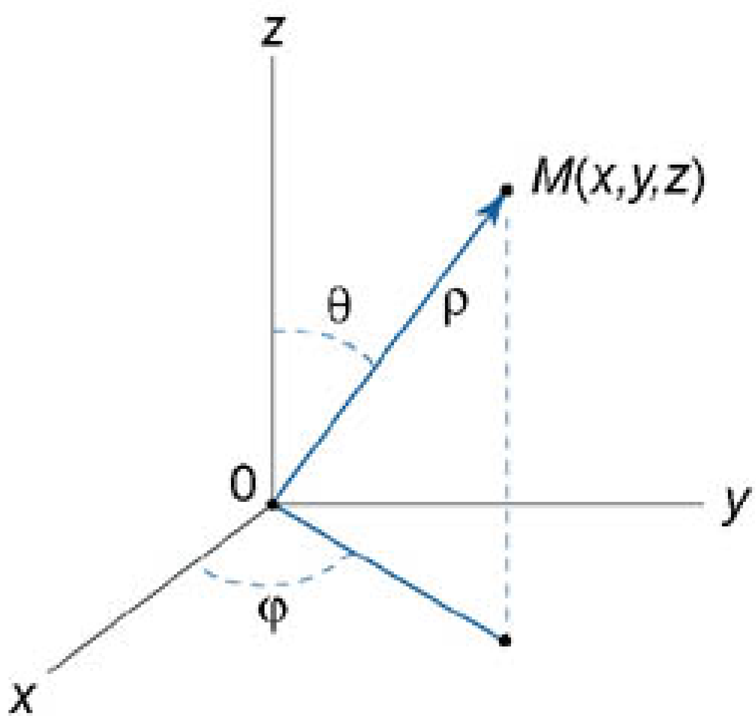


Goal is to predict the 3D configuration of each sidechain

http://t3.gstatic.com/images?q=tbn:ANd9GcS_nfJy1o9yrDt37YlpK7i5s0f7QFqhPrG7-1CLm2AfWNt5wCE50pIKNZd0

Protein Sidechains

- 3D configuration of the sidechain is determined by two angles (spherical coordinates).

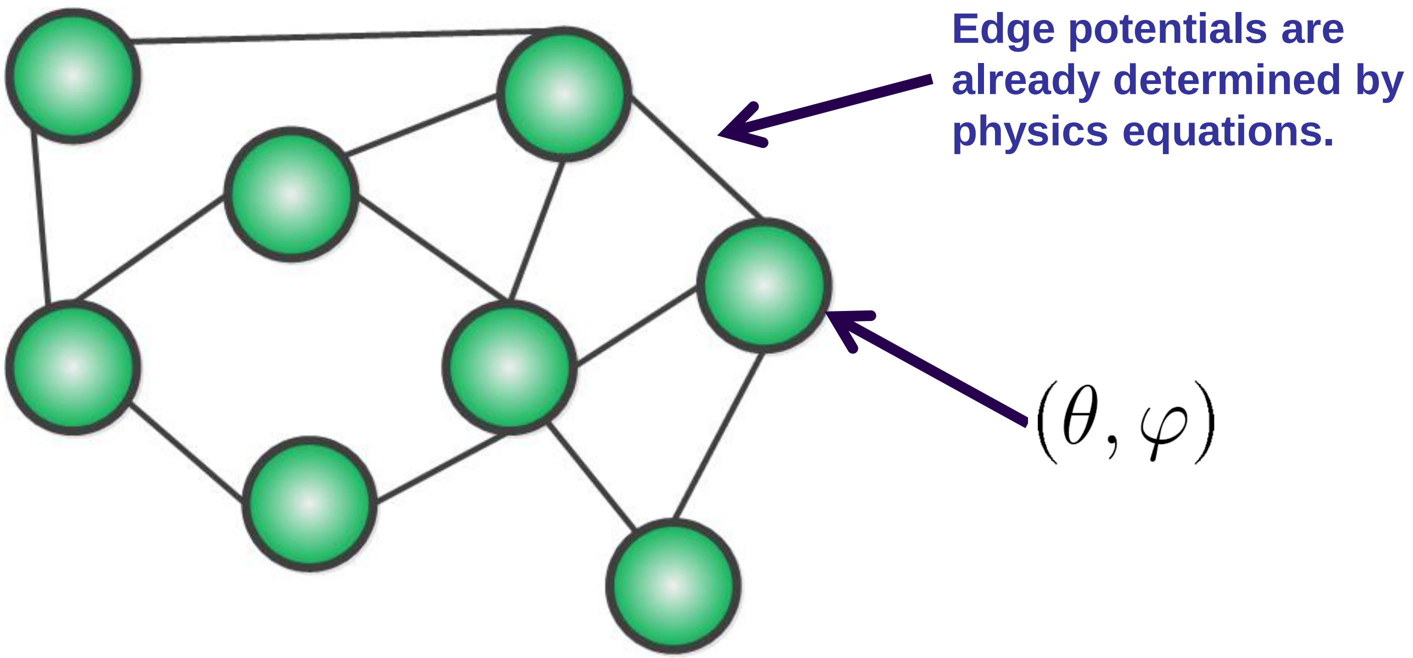


http://www.math24.net/images/triple-int23.jpg

The Graphical Model

- Construct a Markov Random Field.

- Each side-chain angle pair is a node. There is an edge between side-chains that are nearby in the protein.



The Graphical Model

- Goal is to find the MAP assignment of all the sidechain angle pairs.

- Note that this is not Gaussian. But it is easy to define a kernel between angle pairs:

$$\boldsymbol{K}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \exp\left(\boldsymbol{p}_i^\top \boldsymbol{p}_j\right)$$

- Can then run Kernel Belief Propagation ☺

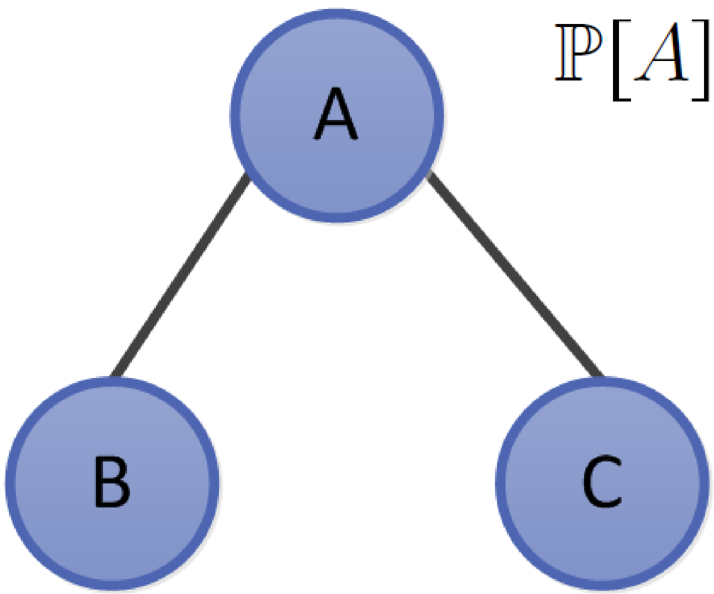
References

- Smola, A. J., Gretton, A., Song, L., and Schölkopf, B., A Hilbert Space Embedding for Distributions, Algorithmic Learning Theory, E. Takimoto (Eds.), Lecture Notes on Computer Science, Springer, 2007.
- L. Song. Learning via Hilbert space embedding of distributions. PhD Thesis 2008.
- Song, L., Huang, J., Smola, A., and Fukumizu, K., Hilbert space embeddings of conditional distributions, International Conference on Machine Learning, 2009.
- Song, L., Gretton, A., and Guestrin, C., Nonparametric Tree Graphical Models via Kernel Embeddings, Artificial Intelligence and Statistics (AISTATS), 2010.
- Song, L., Gretton, A., Bickson, D., Low, Y., and Guestrin, C., Kernel Belief Propagation, International Conference on Artifical Intelligence and Statistics (AISTATS), 2011.

Supplemental: Kernel Belief Propagation on Trees

Kernel Tree Graphical Models [Song et al. 2010]

- The goal is to somehow replace the CPTs with RKHS operators/functions.



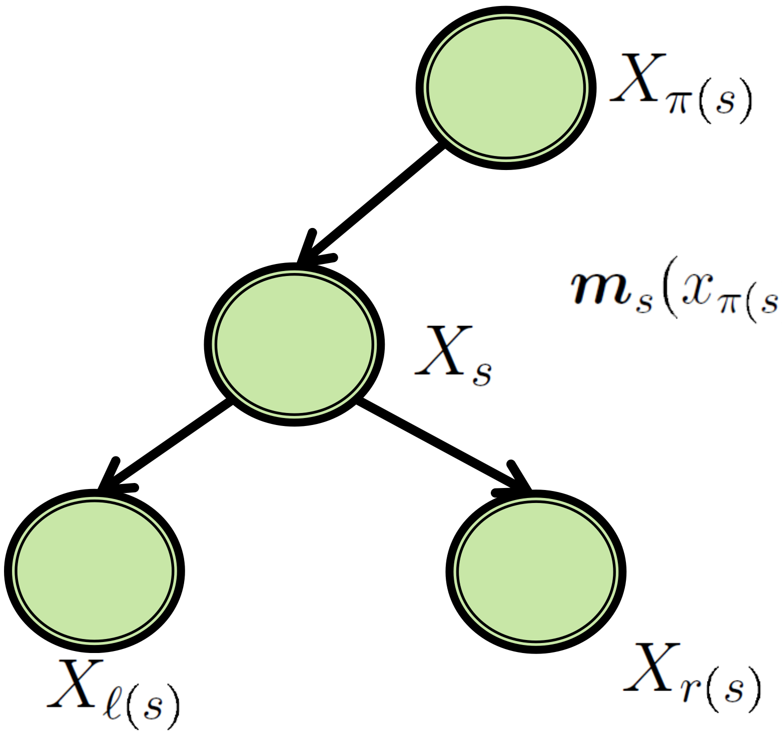
$\mathbb{P}[A] \Rightarrow$ **?**

$\mathbb{P}[C|A] \Rightarrow$ **?**

- **But we need to do this in a certain way so that we can still do inference.**

Message Passing/Belief Propagation

- We need to "matricize" message passing to apply the RKHS trick (but matrices are not enough, we need tensors ☺)



$$\boldsymbol{m}_s(x_{\pi(s)}) \quad = \quad \sum_{X_s} \mathbb{P}[X_s | x_{\pi(s)}] \boldsymbol{m}_{\ell(s)}(X_s) \boldsymbol{m}_{r(s)}(X_s)$$

Outline

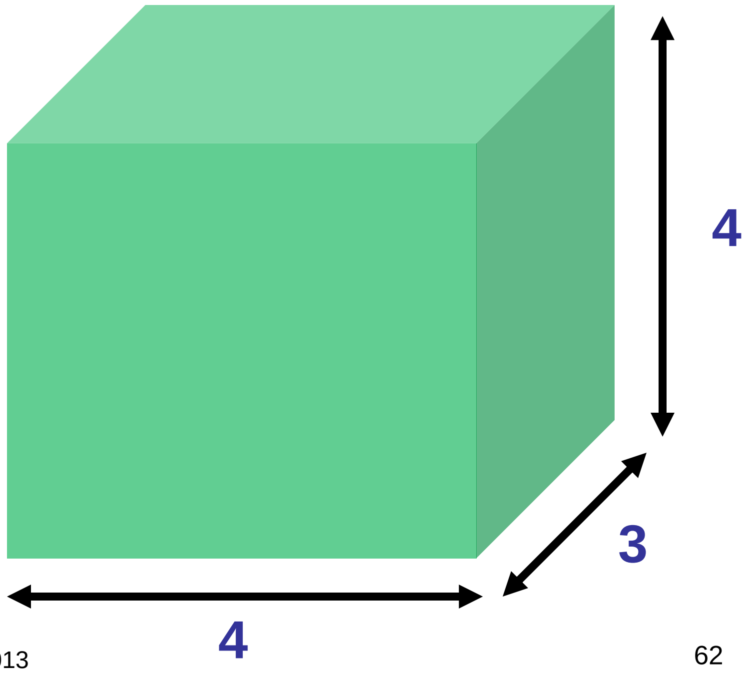
- Show how to represent discrete graphical models using higher order tensors
- Derive *Tensor Message Passing*
- Show how *Tensor Message Passing* can also be derived using Expectations
- Derive ***Kernel Message Passing*** ***[Song et al. 2010]*** using the intuition from *Tensor Message Passing* / Expectations
- (For simplicity, we will assume a binary tree – all internal nodes have 2 children).

Tensors

- Multidimensional arrays
- A Tensor of order ***N*** has ***N*** modes (***N*** indices):

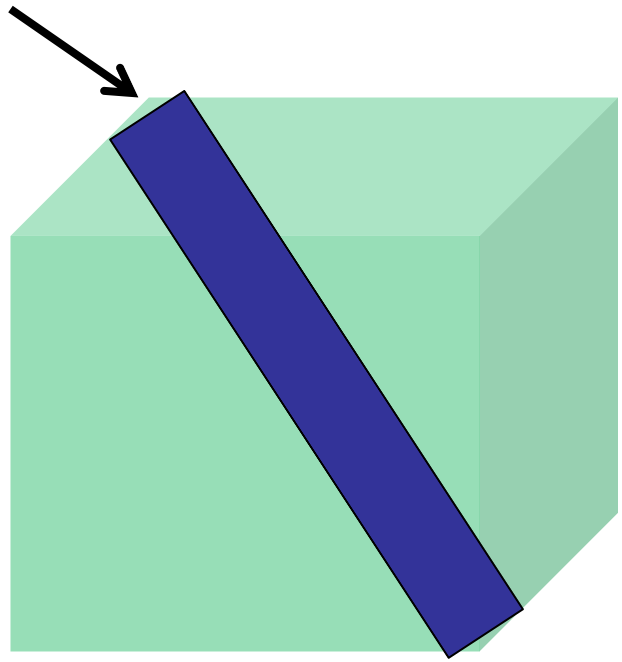
$$\mathcal{T}(i_1, ..., i_N)$$

- Each mode is associated with a dimension. In the example,
  - Dimension of mode 1 is 4
  - Dimension of mode 2 is 3
  - Dimension of mode 3 is 4



Diagonal Tensors

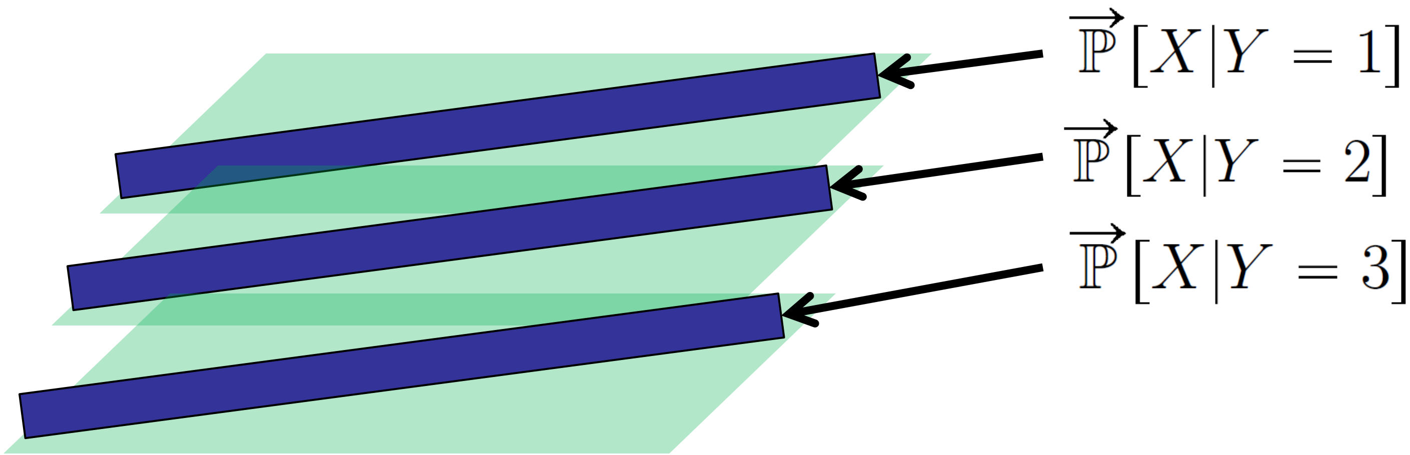
$$\overrightarrow{\mathbb{P}}[X]$$



$$\boldsymbol{\mathcal{T}}(i,j,k) = \begin{cases} \mathbb{P}[X=i] & \text{if } i=j=k \\ 0 & \text{otherwise} \end{cases}$$

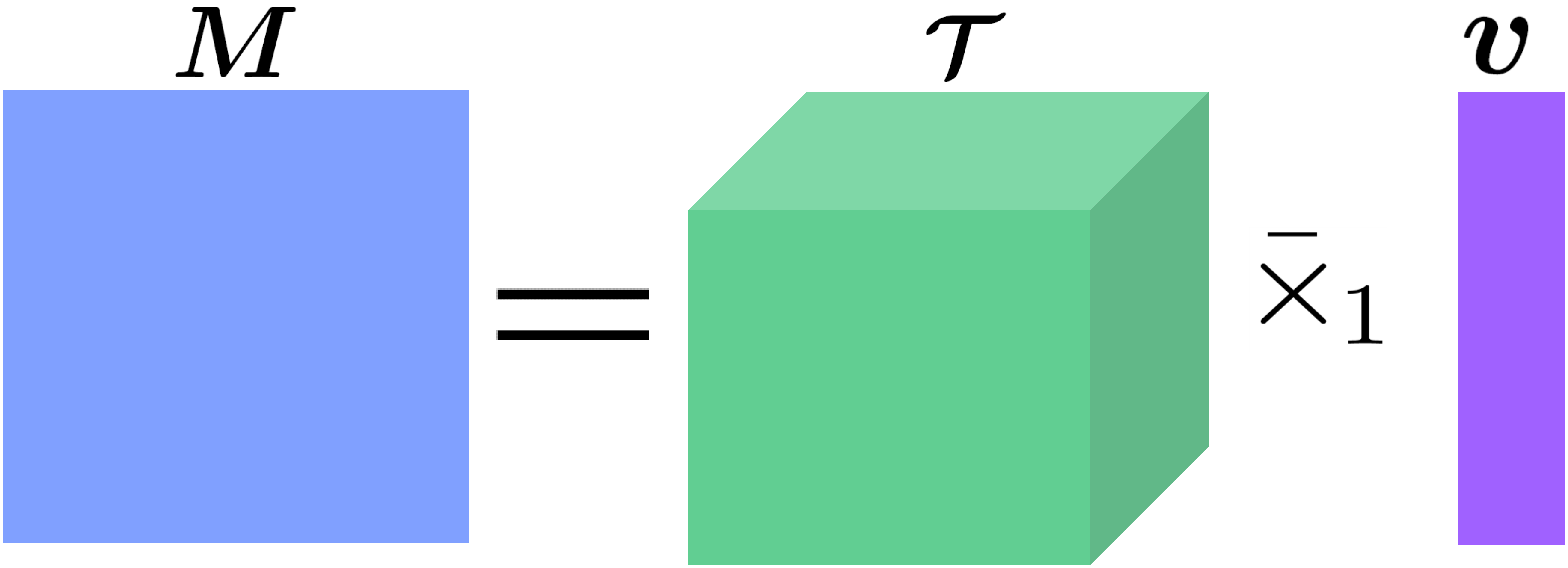
Partially Diagonal Tensors

$$\mathcal{T}(i,j,k) = \begin{cases} \mathbb{P}[X=i|Y=k] & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$



Tensor Vector Multiplication

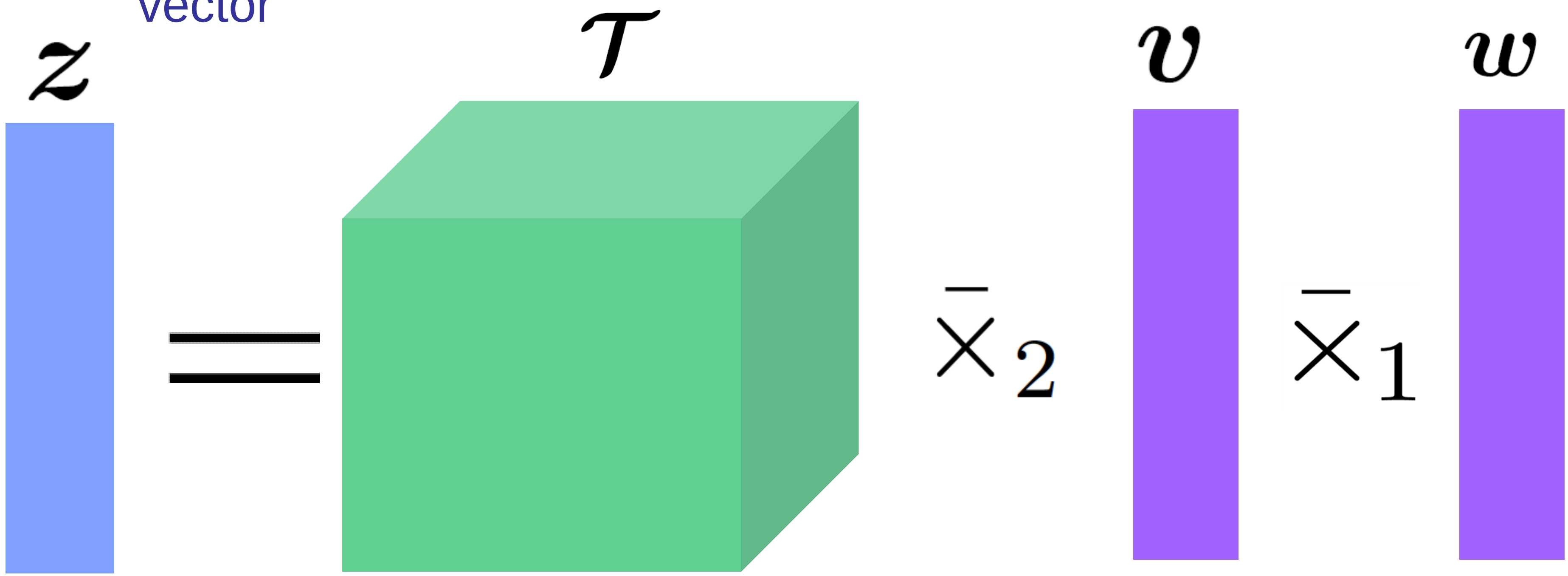
- Multiplying a 3rd order tensor by a vector produces a matrix



$$\boldsymbol{M}(j,k) = \sum_i \boldsymbol{\mathcal{T}}(i,j,k)\boldsymbol{v}(i)$$

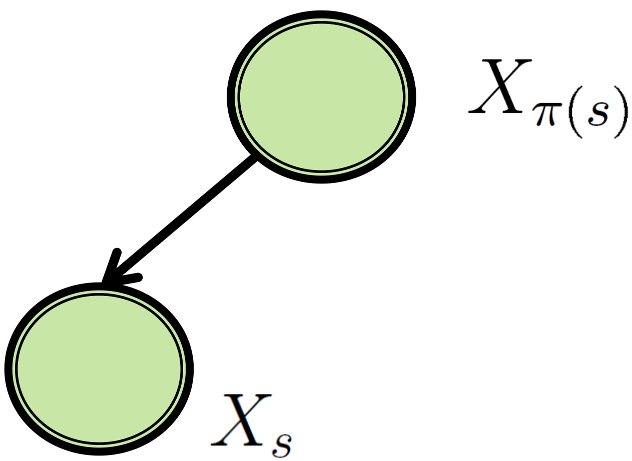
Tensor Vector Multiplication Cont.

- Multiplying a 3rd order tensor by two vectors produces a vector



$$\boldsymbol{z}(k) = \sum_{i} \left(\sum_{j} \mathcal{T}(i,j,k)\boldsymbol{v}(i) \right) \boldsymbol{w}(j) = \sum_{i,j} \mathcal{T}(i,j,k)\boldsymbol{v}(i)\boldsymbol{w}(j)$$

Conditional Probability Table At Leaf is a Matrix

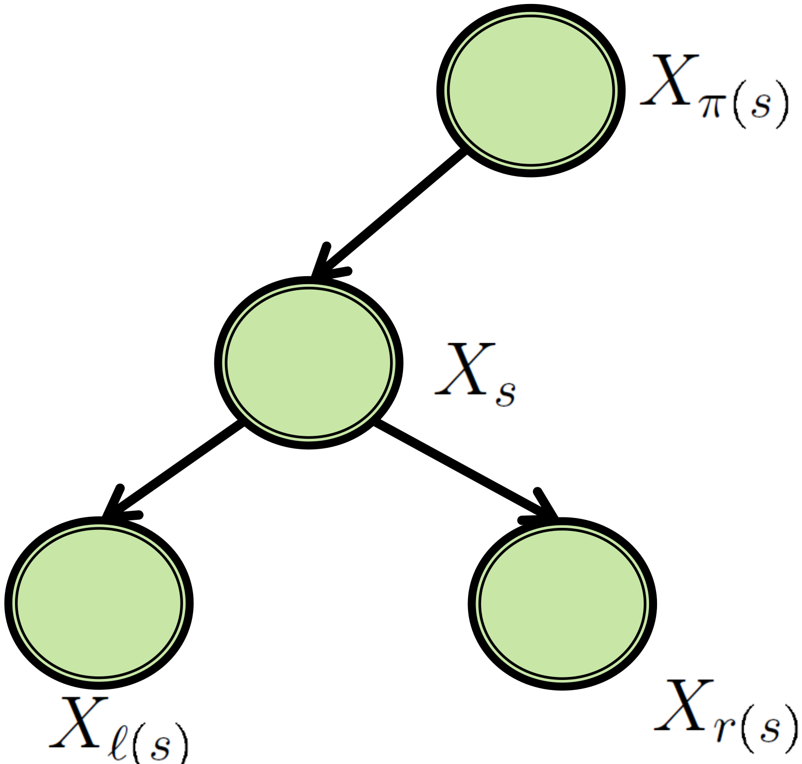


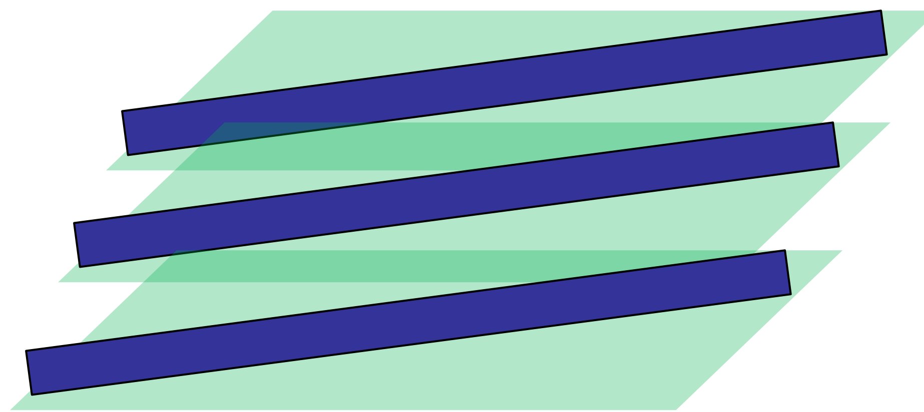
$$\overrightarrow{\mathbb{P}}[X_s | X_{\pi(s)}]$$

CPT At Internal Node (Non-Root) is 3rd Order Tensor

- Note that we have

$$\overrightarrow{\mathbb{P}}[\oslash X_s | X_{\pi(s)}] = \begin{cases} \mathbb{P}[X_s = i | X_{\pi(s)} = k] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

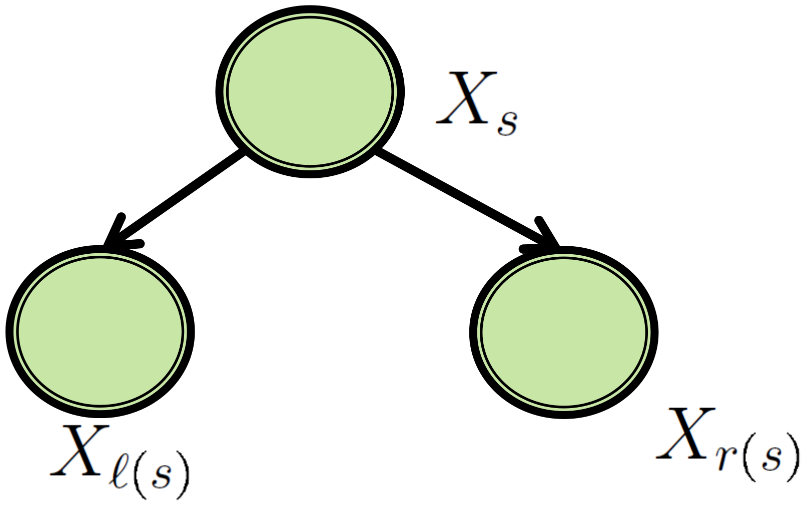


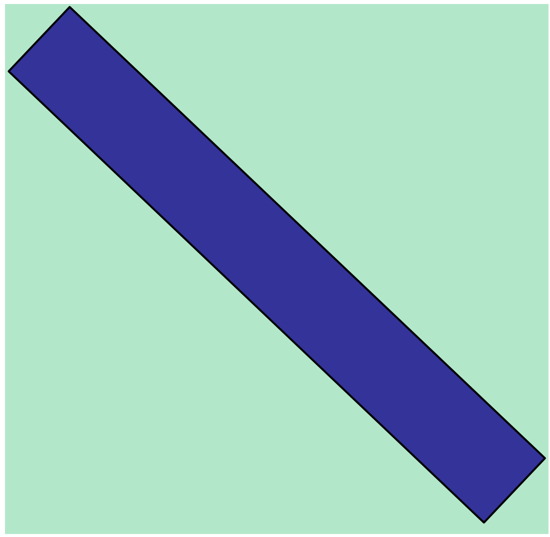


CPT At Root

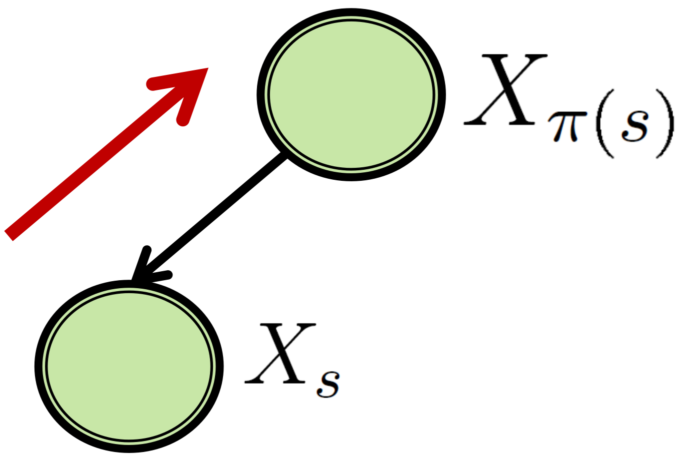
- CPT at root is a matrix.

$$\overrightarrow{\mathbb{P}}[\oslash X_s] = \begin{cases} \mathbb{P}[X_s = i] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$



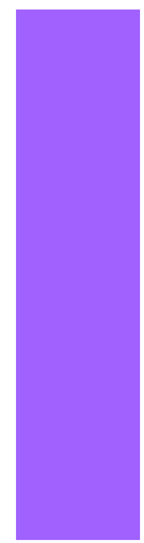


The Outgoing Message as a Vector (at Leaf)

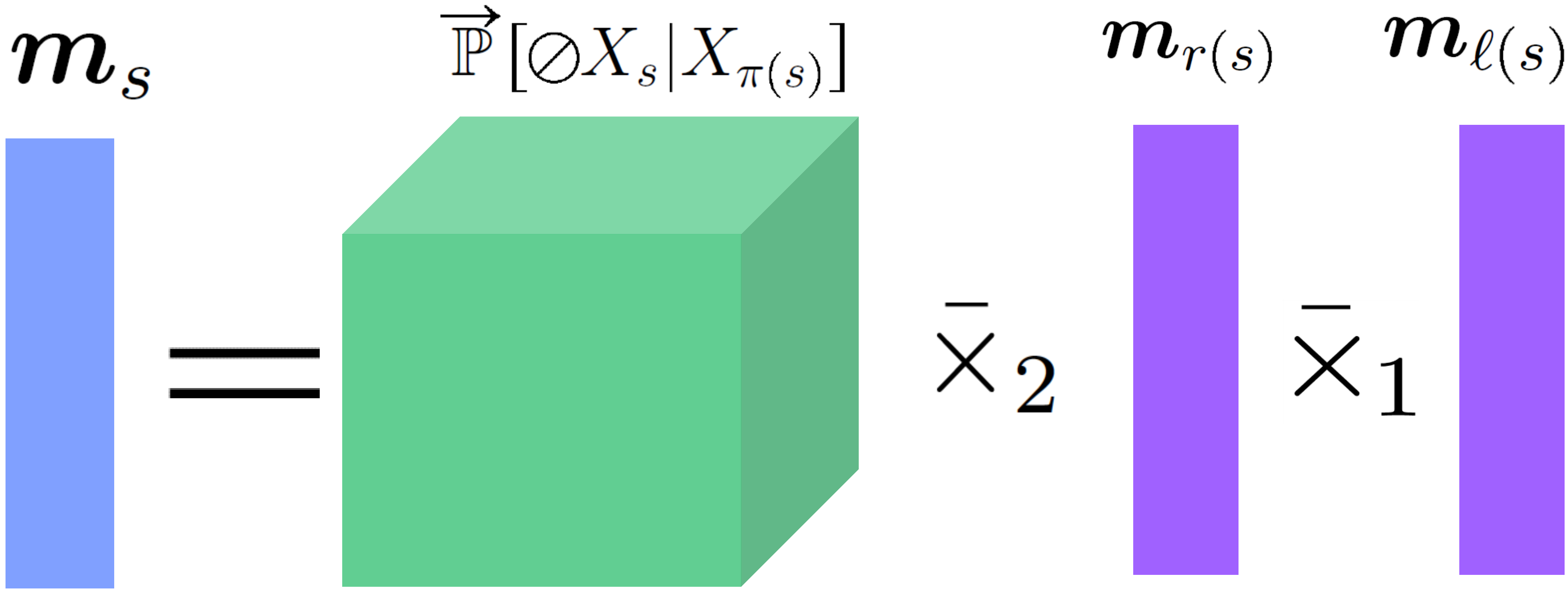


$$\boldsymbol{m}_s = \boldsymbol{\delta}_{\bar{x}_s}^{\top} \overrightarrow{\mathbb{P}}[X_s | X_{\pi(s)}]$$

"bar" denotes evidence



The Outgoing Message At Internal Node



$$\boldsymbol{m}_s(X_{\pi(s)} = k)$$
$$= \sum_{i,j} \mathbb{I}(i = j)\mathbb{P}[X_s = i | X_{\pi(s)} = k]\boldsymbol{m}_{x_{\ell(s)}}(X_s = i)\boldsymbol{m}_{x_{r(s)}}(X_s = j)$$

At the Root

$$\mathbb{P}[\text{evidence}] = \overrightarrow{\mathbb{P}}[\oslash X_s] \; \bar{\times}_2 \; \boldsymbol{m}_{r(s)} \; \bar{\times}_1 \; \boldsymbol{m}_{\ell(s)}$$

$$\mathbb{P}[\text{evidence}] = \sum_{i,j} \mathbb{I}(i=j)\mathbb{P}[X_s = i]\boldsymbol{m}_{x_{\ell(s)}}(X_s = i)\boldsymbol{m}_{x_{r(s)}}(X_s = j)$$

Kernel Graphical Models [Song et al. 2010, Song et al. 2011]

- The Tensor CPTs at each node are replaced with RKHS functions/operators

Leaf: $\overrightarrow{\mathbb{P}}[X_s | X_{\pi(s)}] \Rightarrow \mathcal{C}_{s|\pi(s)}$

Internal (non-root): $\overrightarrow{\mathbb{P}}[\oslash X_s | X_{\pi(s)}] \Rightarrow \mathcal{C}_{ss|\pi(s)}$

Root: $\overrightarrow{\mathbb{P}}[\oslash X_s] \Rightarrow \mathcal{C}_{ss}$

Conditional Embedding Operator for Internal Nodes

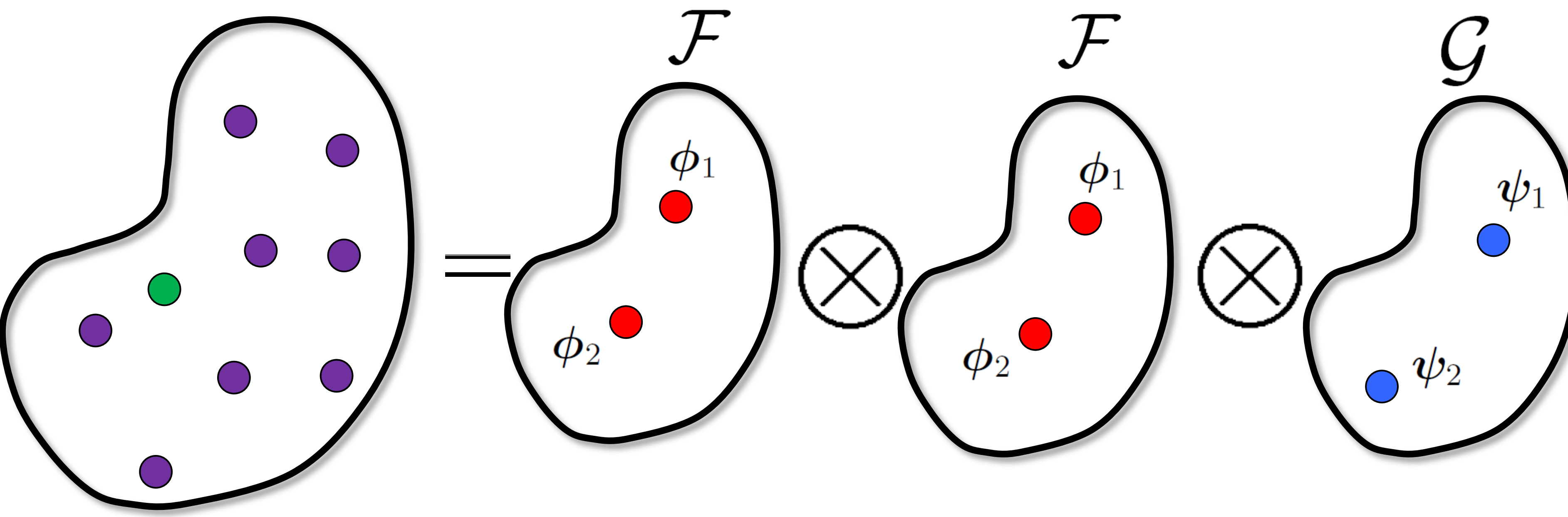
What is $\mathcal{C}_{ss|\pi(s)}$ **?**

$$\mathcal{C}_{XX|Y} = \mathcal{C}_{XXY}\mathcal{C}_{YY}{}^{-1}$$

Embedding of $\mathbb{P}[\oslash X_s | X_{\pi(s)}]$

Embedding of Cross Covariance Operator in Different RKHS

$$\mathcal{C}_{XXY} = \mathbb{E}_{XY}[\phi_X \otimes \phi_X \otimes \psi_Y]$$



Embedding of $\overrightarrow{\mathbb{P}}[\oslash X, Y]$