Title: How to use a Gaussian Boson Sampler to learn from graph-structured data

Speakers: Maria Schuld

Collection: Machine Learning for Quantum Design

Date: July 08, 2019 - 11:30 AM

URL: http://pirsa.org/19070005

Abstract: A device called a â€˜Gaussian Boson Samplerâ€™ has initially been proposed as a near-term demonstration of classically intractable quantum computation. But these devices can also be used to decide whether two graphs are similar to each other. In this talk, I will show how to construct a feature map and graph similarity measure (or â€˜graph kernelâ€™) using samples from an optical Gaussian Boson Sampler, and how to combine this with a support vector machine to do machine learning on graph-structured datasets. I will present promising benchmarking results and try to motivate why such a continuous-variable quantum computer can actually extract interesting properties from graphs.

How to use a Gaussian Boson Sampler to learn from graph-structured data

Maria Schuld

University of KwaZulu-Natal and Xanadu

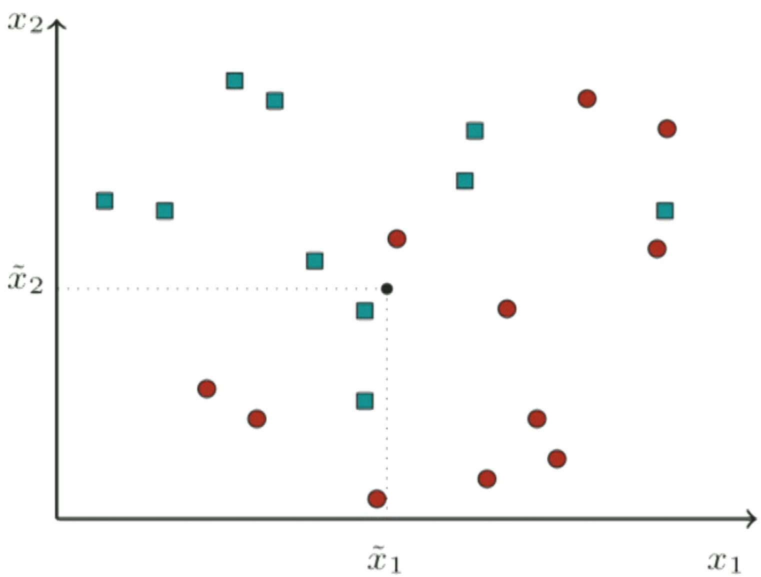
July 2019


PERIMETER
PI
INSTITUTE

IN A NUTSHELL

Machine learning uses distances between feature vectors.

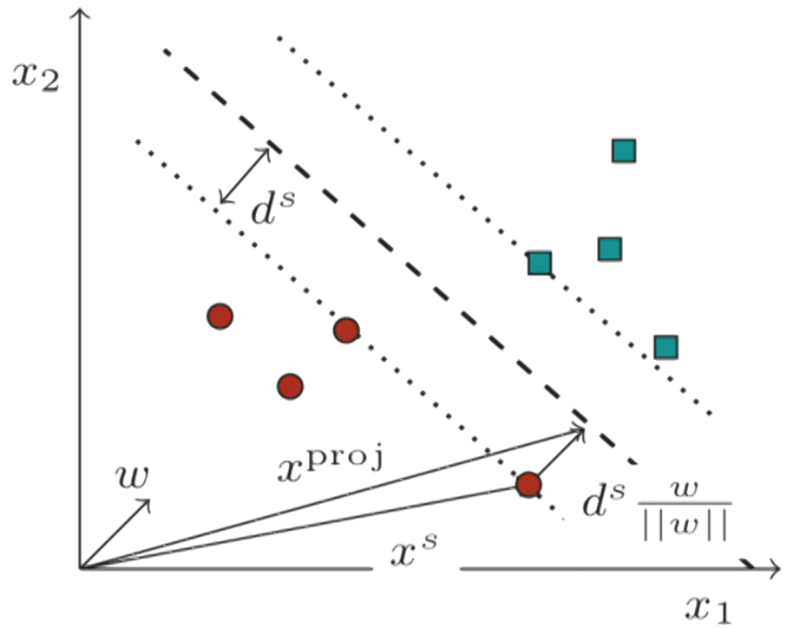
K-Nearest-Neighbour



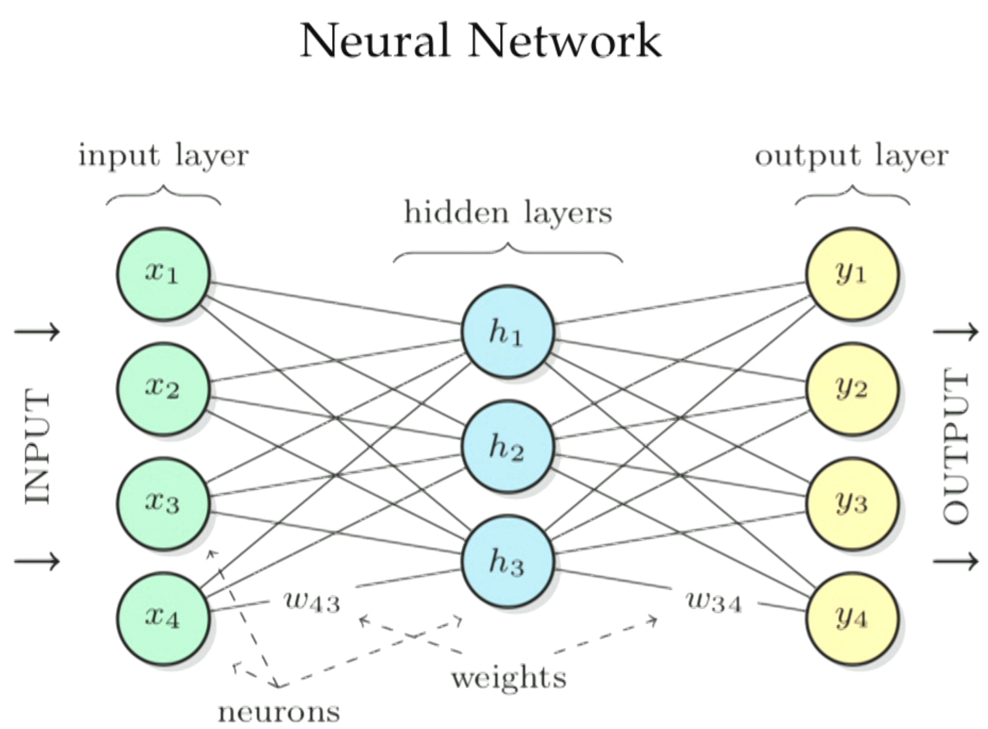
MOTIVATION

Machine learning uses distances between feature vectors.

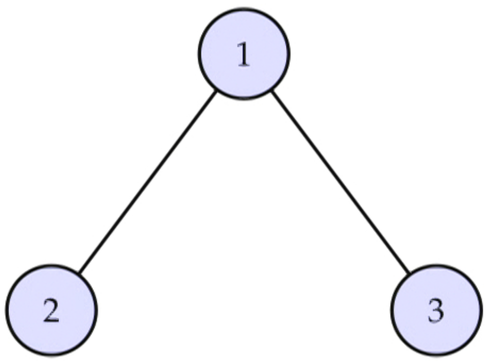
Support Vector Machine



Machine learning uses distances between feature vectors.



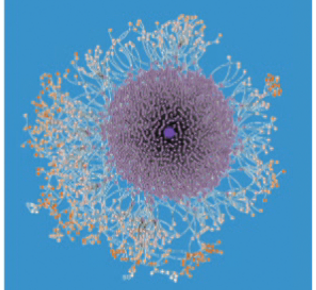
It is difficult to define distances between graphs.



$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

A gallery of *graph kernels* has been developed.

GraKeL



0.1a5

Search docs

User Manual

GraKeL

Kernels (between graphs)

Core Kernel Framework
Edge Histogram Kernel
Graph Hopper Kernel
Graphlet Sampling Kernel
Hadamard Code Kernel
Kernel (general class)
Lovasz Theta Kernel
Multiscale Laplacian Kernel
Neighborhood Hash Kernel
Neighborhood Subgraph Pairwise Distance Kernel

Docs » GraKeL » Kernels (between graphs) View page source

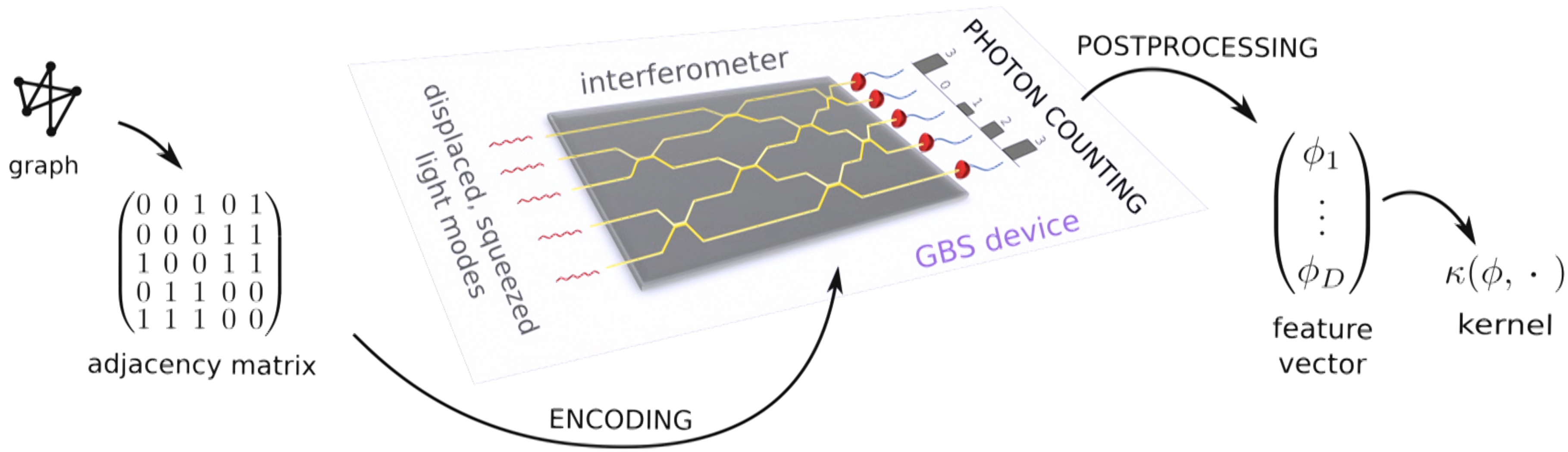
Kernels (between graphs)

The documentation of the *kernels* submodule.

- Core Kernel Framework
- Edge Histogram Kernel
- Graph Hopper Kernel
- Graphlet Sampling Kernel
- Hadamard Code Kernel
- Kernel (general class)
- Lovasz Theta Kernel
- Multiscale Laplacian Kernel
- Neighborhood Hash Kernel
- Neighborhood Subgraph Pairwise Distance Kernel
- ODD-STh Kernel
- Propagation Kernel
- Pyramid Match Kernel
- Random Walk Kernel
- Shortest Path Kernel
- Subgraph Matching Kernel
- SVM Theta Kernel
- Vertex Histogram Kernel
- Weisfeiler Lehman Framework

Previous Next

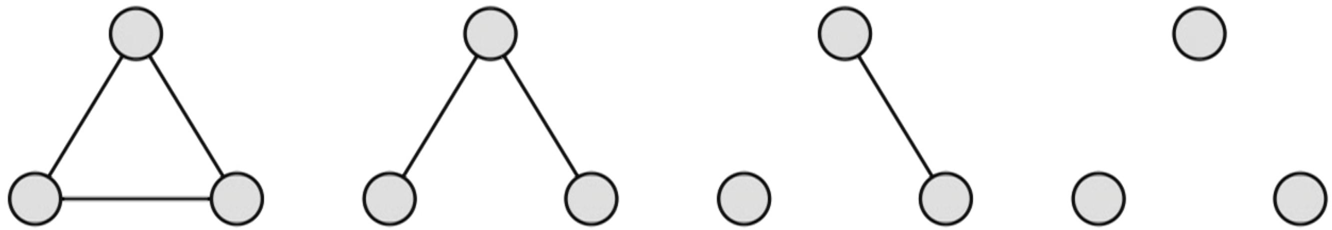
We can use quantum light to formulate a graph kernel.



[Schuld, Bradler, Israel, Su, Gupt - arXiv:1905.12646]

The GBS graph kernel is very similar to a *graphlet sampling kernel*.

Graphlet Sampling Kernel

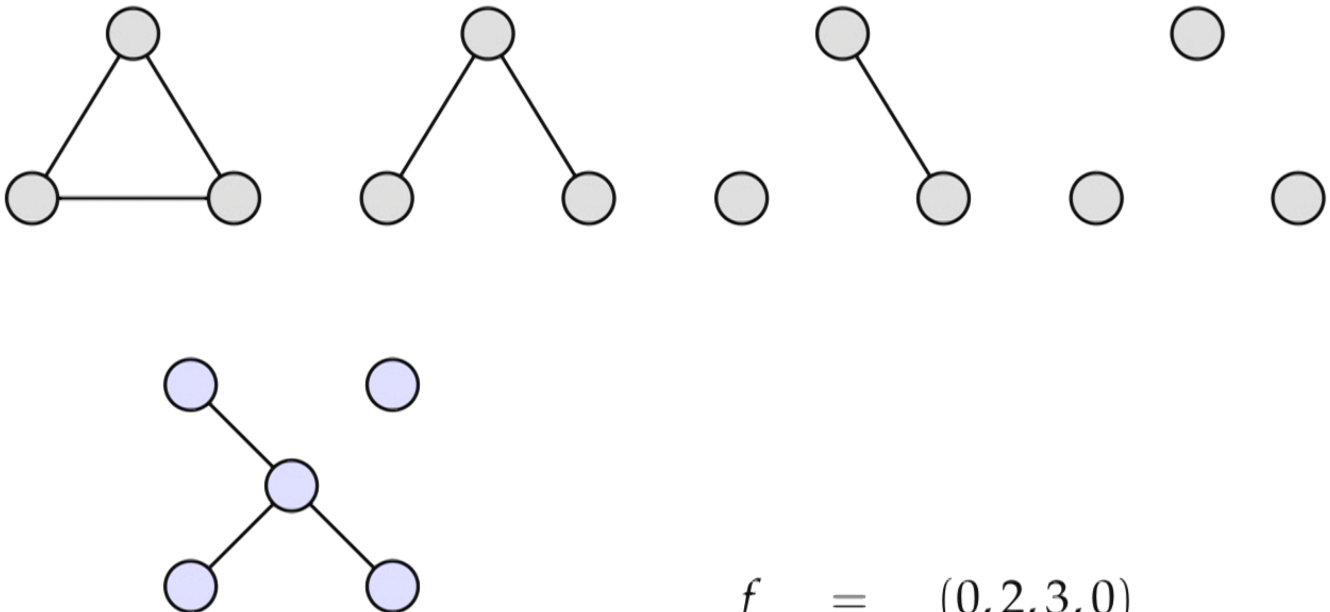


Shervashidze, Nino, et al. *Artificial Intelligence and Statistics* (2009).

MOTIVATION

The GBS graph kernel is very similar to a *graphlet sampling kernel*.

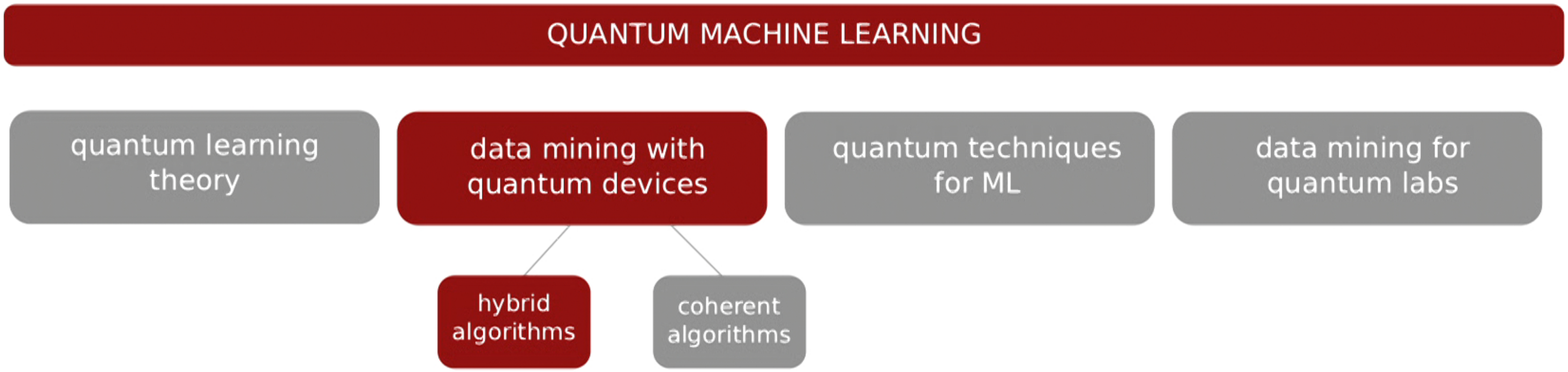
Graphlet Sampling Kernel



$$f = (0, 2, 3, 0)$$

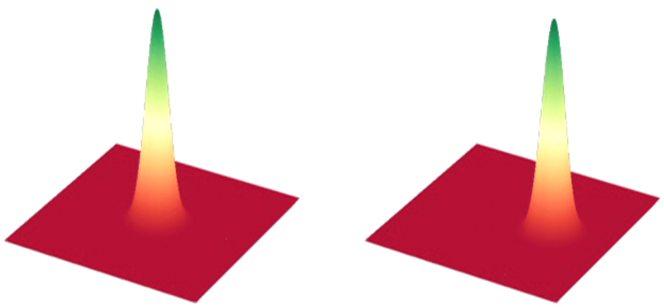
Shervashidze, Nino, et al. *Artificial Intelligence and Statistics* (2009).

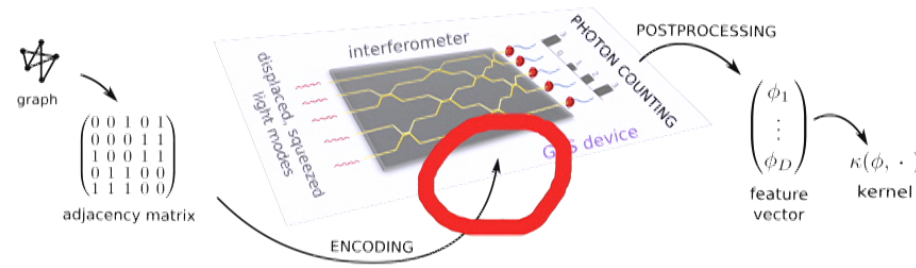
The work is part of hybrid/near-term QML.



MOTIVATION

A GBS device can encode a graph.



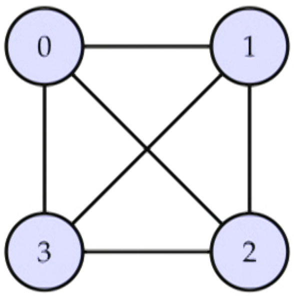


A Gaussian state of M optical modes is fully described by a *covariance matrix* $\sigma \in \mathbb{R}^{2M\times 2M}$ as well as a *displacement vector* $d \in \mathbb{R}^{2M}$.

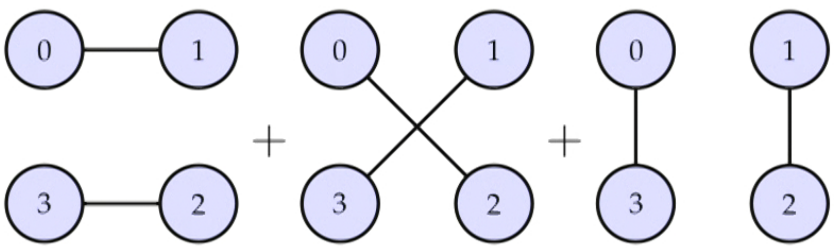
We can associate such a state with an adjacency matrix A via

$$\sigma = (\mathbb{1} - X\tilde{A})^{-1} - \frac{\mathbb{1}}{2}, \text{ with } X = \begin{pmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{pmatrix}, \tilde{A} = c\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}.$$

The Hafnian is related to the number of perfect matchings.

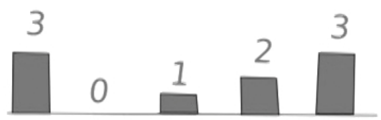


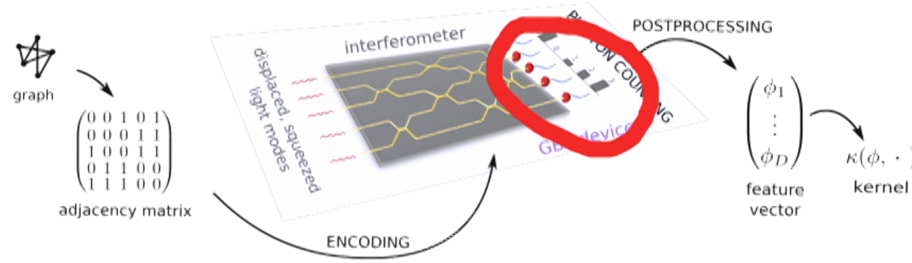
$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$



$$\mathrm{Haf}(A) = a_{01}a_{23} + a_{02}a_{13} + a_{03}a_{12}$$

The photon counting distribution depends on the Hafnian.

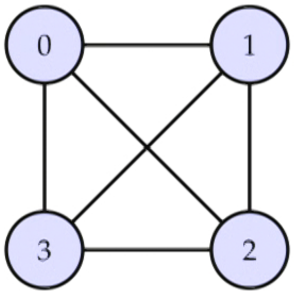




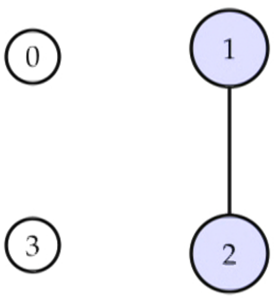
Let $\mathbf{n} = (n_1, ..., n_M)$, $n_m \in \mathbb{N}$ be a "photon click pattern" measured by M photon detectors. The probability of this measurement is given by

$$p(\mathbf{n}) \propto \mathrm{Haf}^2(A_{\mathbf{n}})$$

$A_{\mathbf{n}}$ is an (extended) subgraph of A.

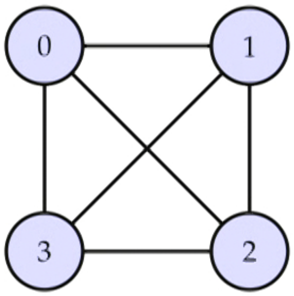


$$A_{[1,1,1,1]} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

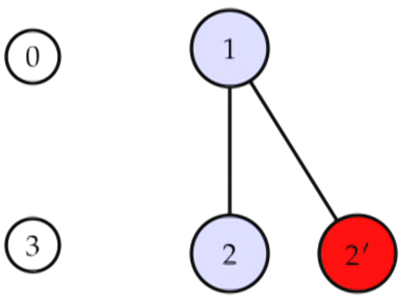


$$A_{[0,1,1,0]} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$A_{\mathbf{n}}$ is an (extended) subgraph of A.



$$A_{[1,1,1,1]} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$



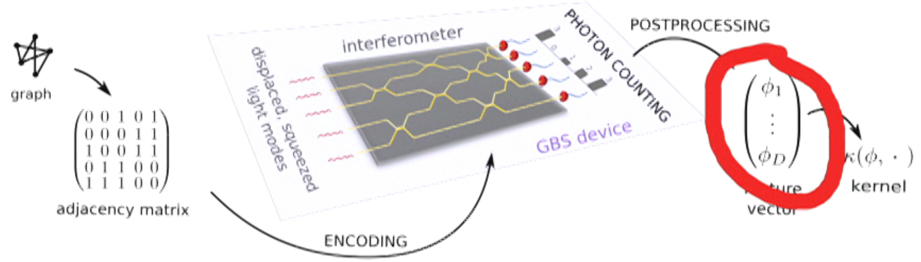
$$A_{[0,1,2,0]} = \begin{pmatrix} a_{11} & a_{12} & a_{12'} \\ a_{21} & a_{22} & a_{22'} \\ a_{2'1} & a_{2'2} & a_{2'2'} \end{pmatrix}$$

Original graph
$A =$

$\mathbf{n}$	$\lvert\mathbf{n}\rvert$	$A_{\mathbf{n}}$	$\mathrm{Haf}(A_{\mathbf{n}})$
[0, 0, 0]	0		0
[1, 0, 0]			0
[0, 1, 0]	1		0
[0, 0, 1]			0
[1, 1, 0]			1
[1, 0, 1]	2		1
[0, 1, 1]			1
[2, 0, 0]			0
[0, 2, 0]	2		0
[0, 0, 2]			0

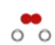
$\mathbf{n}$	$\lvert\mathbf{n}\rvert$	$A_{\mathbf{n}}$	$\mathrm{Haf}(A_{\mathbf{n}})$
[1, 1, 1]	3		0
[2, 1, 0]			0
[2, 0, 1]			0
[1, 2, 0]	3		0
[1, 0, 2]			0
[0, 2, 1]			0
[0, 1, 2]			0
[3, 0, 0]			0
[0, 3, 0]	3		0
[0, 0, 3]			0

We summarise photon events to orbits.



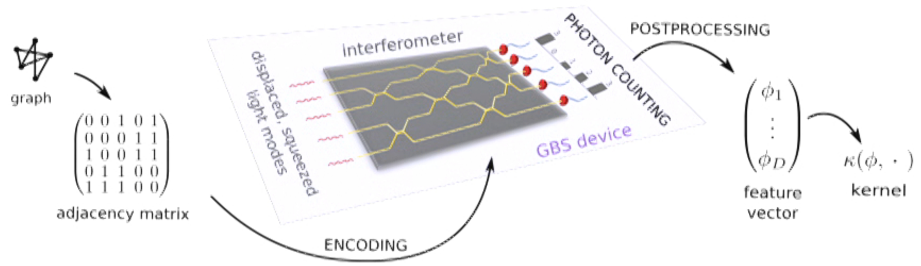
An *orbit* is a set of photon click events which are permutations of each other.

We summarise photon events to orbits.

$O_{\mathbf{n}^*}$	$\mathbf{n}$	$\lvert\mathbf{n}\rvert$	$A_{\mathbf{n}}$	$\mathrm{Haf}(A_{\mathbf{n}})$
$O_{[0,0,0]}$	$[0,0,0]$	0		0
$O_{[1,0,0]}$	$[1,0,0]$	1		0
	$[0,1,0]$			0
	$[0,0,1]$			0
$O_{[1,1,0]}$	$[1,1,0]$	2		1
	$[1,0,1]$			1
	$[0,1,1]$			1
$O_{[2,0,0]}$	$[2,0,0]$	2		0
	$[0,2,0]$			0
	$[0,0,2]$			0

$O_{\mathbf{n}^*}$	$\mathbf{n}$	$\lvert\mathbf{n}\rvert$	$A_{\mathbf{n}}$	$\mathrm{Haf}(A_{\mathbf{n}})$
$O_{[1,1,1]}$	$[1,1,1]$	3		0
$O_{[2,1,0]}$	$[2,1,0]$	3		0
	$[2,0,1]$			0
	$[1,2,0]$			0
	$[1,0,2]$			0
	$[0,2,1]$			0
	$[0,1,2]$			0
$O_{[3,0,0]}$	$[3,0,0]$	3		0
	$[0,3,0]$			0
	$[0,0,3]$			0

The probabilities of the orbits are the features of a graph.



$$f_G = \begin{pmatrix} p(O_{[0,0,0]}) \\ \vdots \\ p(O_{[0,0,3]}) \\ \vdots \end{pmatrix}, \ \kappa(G, G') = \langle f_G, f_{G'} \rangle$$

GBS KERNEL

RUNTIME & INTERPRETATION

18 / 26

We can extract the features in the fraction of a second.

$$S = \left\lceil \frac{2(\log(2)D + \log(\frac{1}{\delta}))}{\epsilon^2} \right\rceil$$

For $k = 8$, $D = 67$, $\epsilon = 0.05$ and $\delta = 0.05$, we need $39,550$ samples.

Photon number resolving detectors can accumulate about 10^5 samples per second.

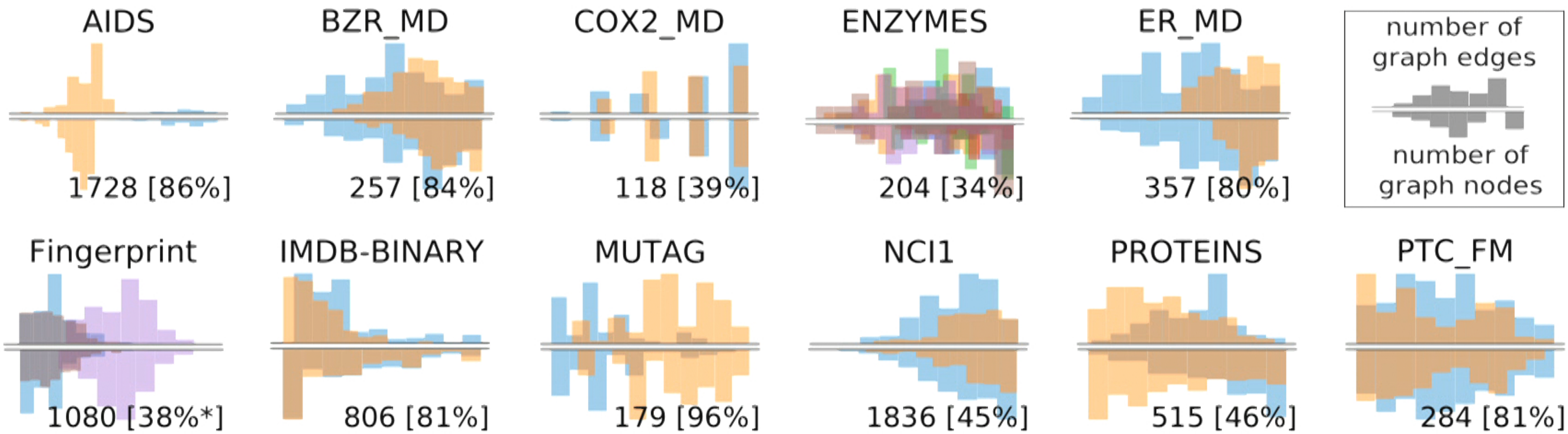
The features are related to matching polynomials.

- The coefficients $m(G,r) = \sum_{\mathbf{n}\in O_{[1,\ldots,1,0,\ldots]}} \mathrm{Haf}(A_{\mathbf{n}})$ of a *matching polynomial* count the number of r-matchings or "independent edge sets" in G – sets of r edges that have no vertex in common.

The features are related to matching polynomials.

- The coefficients $m(G,r) = \sum_{\mathbf{n}\in O_{[1,\dots,1,0,\dots]}} \mathrm{Haf}(A_{\mathbf{n}})$ of a *matching polynomial* count the number of r-matchings or "independent edge sets" in G – sets of r edges that have no vertex in common.
- The higher-order moments $E[X_1^{(1)} \dots X_1^{(n_1)} \dots X_M^{(1)} \dots X_M^{(n_M)}]$ of a *multivariate Gaussian distribution* $N(f_1(d), f_2(\sigma))$ are proportional to $\mathrm{Haf}(A_{\mathbf{n}})$

We can test the kernel on standard graph data sets.



The results are promising.

Dataset	GBS ($d = 0.0$)	GBS ($d = 0.25$)	GS	RW	SM
AIDS	99.60 ± 0.05	**99.62** ± 0.03	98.44 ± 0.09	56.95 ± 7.99	79.20 ± 0.68
BZR_MD	**62.73** ± 0.71	62.13 ± 1.44	60.60 ± 1.77	49.88 ± 3.74	61.90 ± 1.21
COX2_MD	44.98 ± 1.80	50.11 ± 0.97	55.04 ± 3.33	57.72 ± 3.26	**66.94** ± 1.22
ENZYMES	22.29 ± 1.60	28.01 ± 1.83	35.87 ± 2.19	21.13 ± 1.91	**36.70** ± 2.83
ER_MD	70.36 ± 0.78	**70.41** ± 0.47	65.65 ± 1.06	68.75 ± 0.53	68.21 ± 0.99
FINGERPRINT	65.42 ± 0.49	**65.85** ± 0.36	64.10 ± 1.52	47.69 ± 0.21	47.14 ± 0.62
IMDB-BIN	64.09 ± 0.34	68.71 ± 0.59	68.37 ± 0.62	66.38 ± 0.21	out of time*
MUTAG	**86.41** ± 0.33	85.58 ± 0.59	81.08 ± 0.93	83.02 ± 1.08	83.14 ± 0.24
NCI1	**63.61** ± 0.00	62.79 ± 0.00	49.96 ± 3.27	52.36 ± 2.63	51.36 ± 1.88
PROTEINS	**66.88** ± 0.22	66.14 ± 0.48	65.91 ± 1.29	56.27 ± 1.23	63.03 ± 0.84
PTC_FM	53.84 ± 0.96	52.45 ± 1.78	**59.48** ± 1.95	51.97 ± 2.68	54.92 ± 2.94

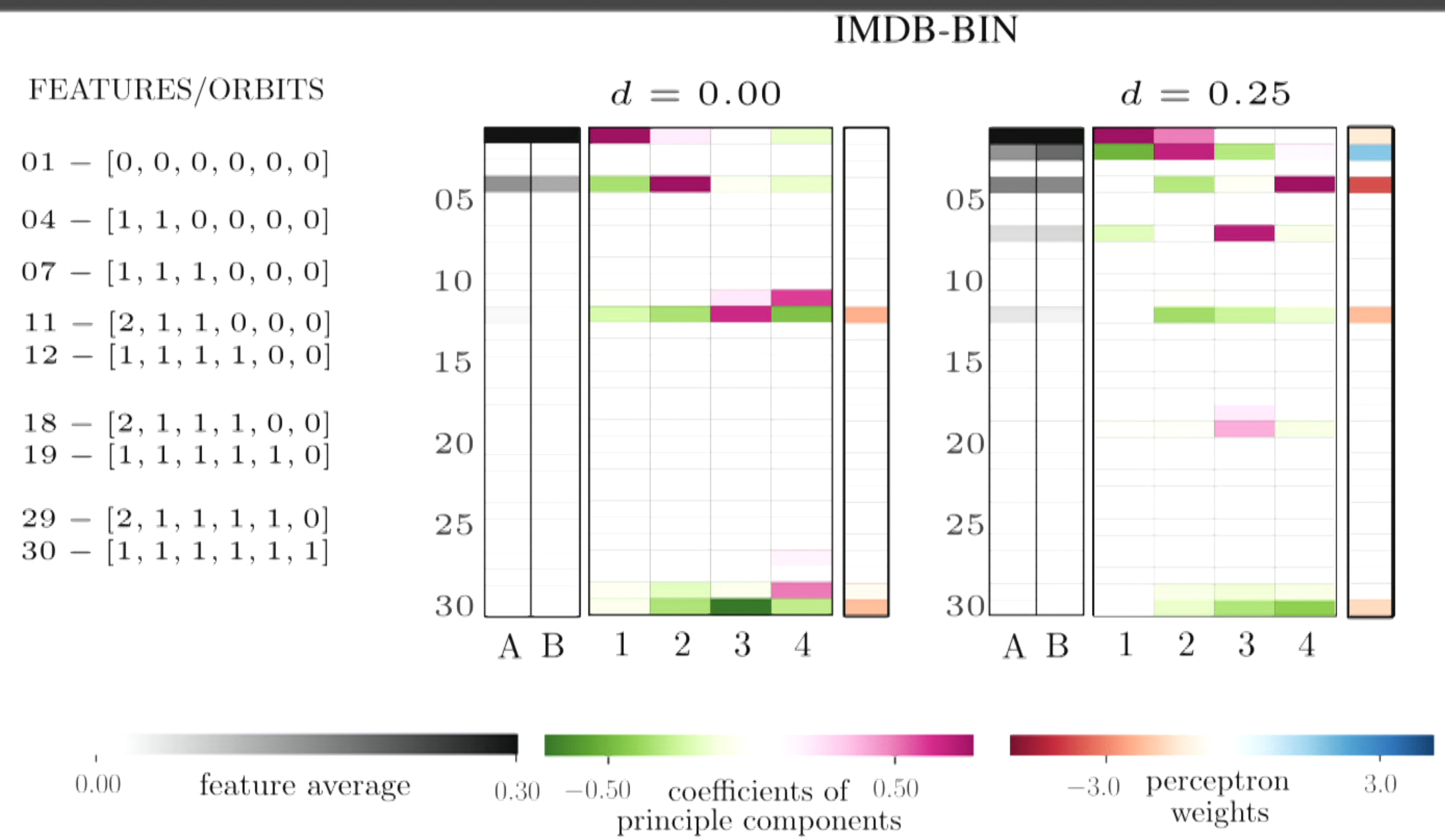
Displacement is an important hyperparameter.

$$p(\mathbf{n}) \propto \mathrm{Haf}(A_{\mathbf{n}})^2 \to p(\mathbf{n},\mathbf{d}) \propto \left[\sum_{n=0}^{M} \sum_{\{i_1..i_n\}\subseteq \mathcal{I}_M} b_{i_1} \cdot \cdots \cdot b_{i_n} \mathrm{Haf}(A_{\mathbf{n}-\{i_1,\ldots,i_n\}})\right]^2$$

Displacement is an important hyperparameter.

$$\text{LOOPHaf}(C) = a_{01}a_{23} + a_{02}a_{13} + a_{03}a_{12}$$
$$+ a_{00}a_{11}a_{23} + a_{01}a_{22}a_{33} + a_{02}a_{11}a_{33} + a_{00}a_{22}a_{13} + a_{00}a_{33}a_{12} + a_{03}a_{11}a_{22}$$
$$+ a_{00}a_{11}a_{22}a_{33}$$

Low-photon features seem most important.



Take-away.

- (For ML crowd:) It seems that duplicating nodes is a useful strategy when comparing graphs through subgraph structures.
- (For QML crowd:) Near-tearm quantum hardware can improve machine learning of graph-structured data