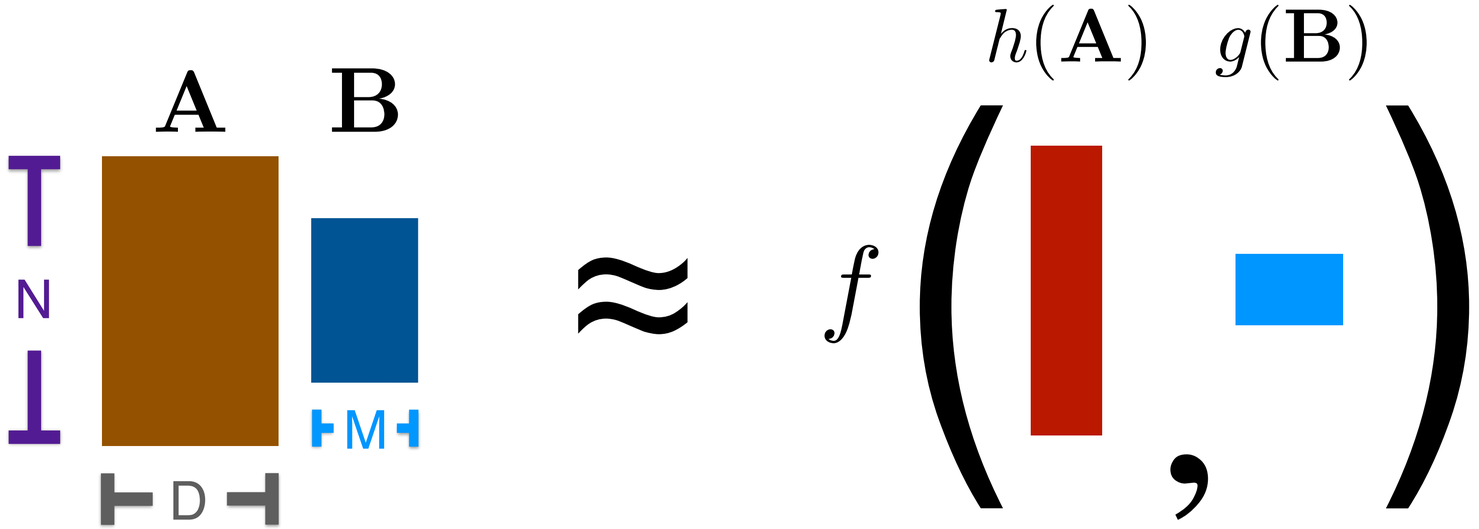
Multiplying Matrices Without Multiplying

Davis Blalock[1,2] & John Guttag[1]

[1]MIT CSAIL

[2]MosaicML

Problem Formulation



$$\underset{\theta}{\operatorname{argmin}} \, ||f(h(\mathbf{A};\theta), g(\mathbf{B};\theta)) - \mathbf{AB}||_F$$

- Assumptions:
  - Single machine
  - Dense matrices
  - Have training set $\tilde{\mathbf{A}}$
  - Works best on tall matrices

Encoding A with “One 1” sparsity

Row **a** $\langle$ `1.64  8.72  -4.4  91.0  16.1 -95.5 89.4 -26.7 -41.8 61.0 212.3 71.5 -70.8` $\rangle$

Encoding A with “One 1” sparsity

Row **a** $\langle$ `1.64  8.72  -4.4  91.0  16.1 -95.5 89.4 -26.7 -41.8 61.0 212.3 71.5 -70.8` $\rangle$

Trained “one 1” encoding

Row **a** as a sparse vector $\langle$ `0 0 1 0|0 1 0 0|0 0 0 1` $\rangle$

Encoding A with “One 1” sparsity

Row **a** ⟨1.64 8.72 -4.4 91.0 16.1 -95.5 89.4 -26.7 -41.8 61.0 212.3 71.5 -70.8⟩

Trained “one 1” encoding

Row **a** as a sparse vector ⟨0 0 1 0|0 1 0 0|0 0 0 1⟩

Store which element is 1

Row **a** as a vector of indices ⟨ 01 | 10 | 00 ⟩

Encoding A with “One 1” sparsity

Row **a** ⟨1.64 8.72 -4.4 91.0 16.1 -95.5 89.4 -26.7 -41.8 61.0 212.3 71.5 -70.8⟩

Trained “one 1” encoding

Row **a** as a sparse vector ⟨0 0 1 0|0 1 0 0|0 0 0 1⟩

Store which element is 1

Row **a** as a vector of indices ⟨ 01 | 10 | 00 ⟩

- Sparse vector is output of trained, nonlinear encoding function
  - Totally new vector; **not a heuristically zeroed/quantized a**

With One 1, Dot Products = Table Lookups

Dot Product: $\sum_{i=1}^{D} a[i]b[i]$

Row **a** as a sparse vector

0	0	1	0	0	1	0	0	0	0	0	1

Column **b** as a dense vector

1.6	91	-8	22	104	9.1	10	42	95	2.4	0.2	-1

Encoding g(**b**) so that lengths match

=

Table Lookups: $\sum_{c=1}^{C} b[c, a[c]]$

Row **a** as a list of indices

01	10	00

Column **b** as a list of tables

1.6	91	-8	22	104	9.3	10.9	42	95	2.4	0.2	-1

One 1 Sparsity Yields “Dense” Matrices

$h(\mathbf{A})$

01	10	00
11	01	01
10	00	11
00	11	01

$g(\mathbf{B})$

$\text{Tbl}_{0,0}$	$\text{Tbl}_{0,1}$
$\text{Tbl}_{1,0}$	$\text{Tbl}_{1,1}$
$\text{Tbl}_{2,0}$	$\text{Tbl}_{2,1}$

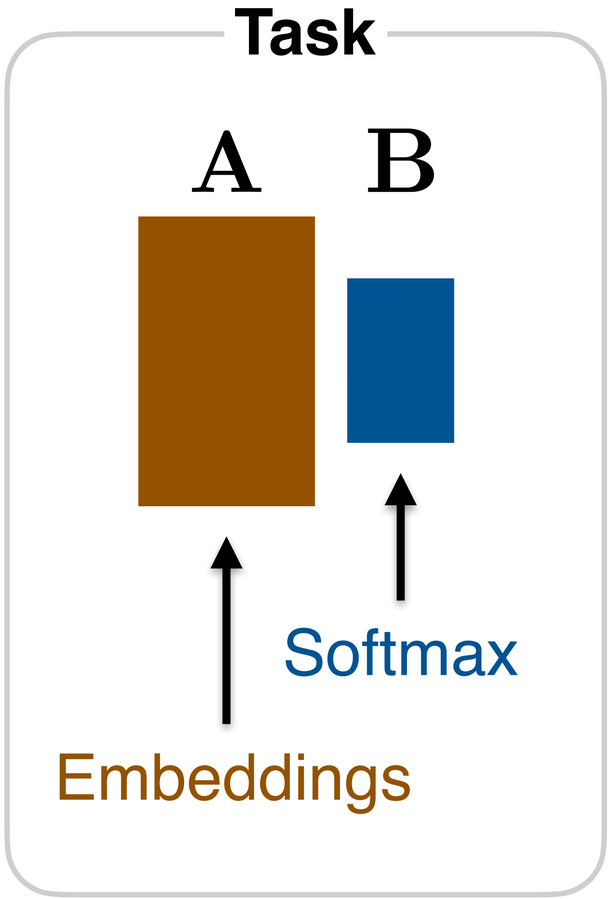
=

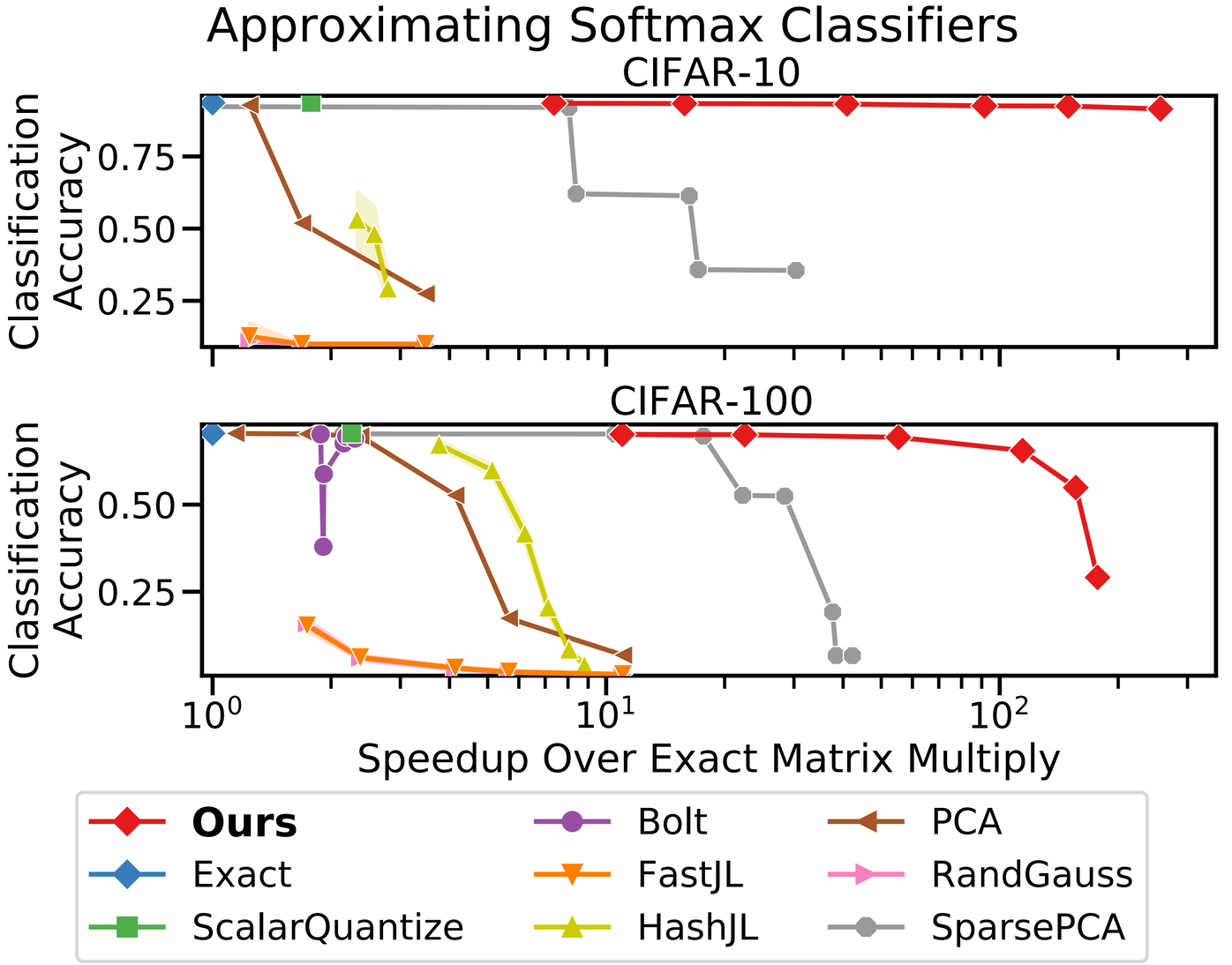
$f(h(\mathbf{A};\theta), g(\mathbf{B};\theta))$

1.4	9.3
-6	24
18	-11
91	-4.5

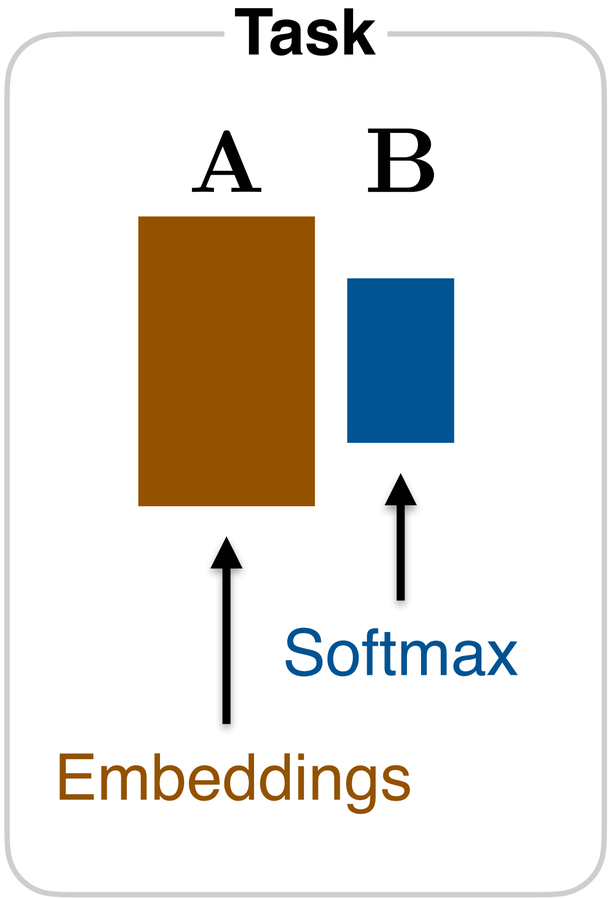
- Both indices and tables have contiguous elements of uniform size
- Like a dense GEMM but with lookup-add instead of multiply-add

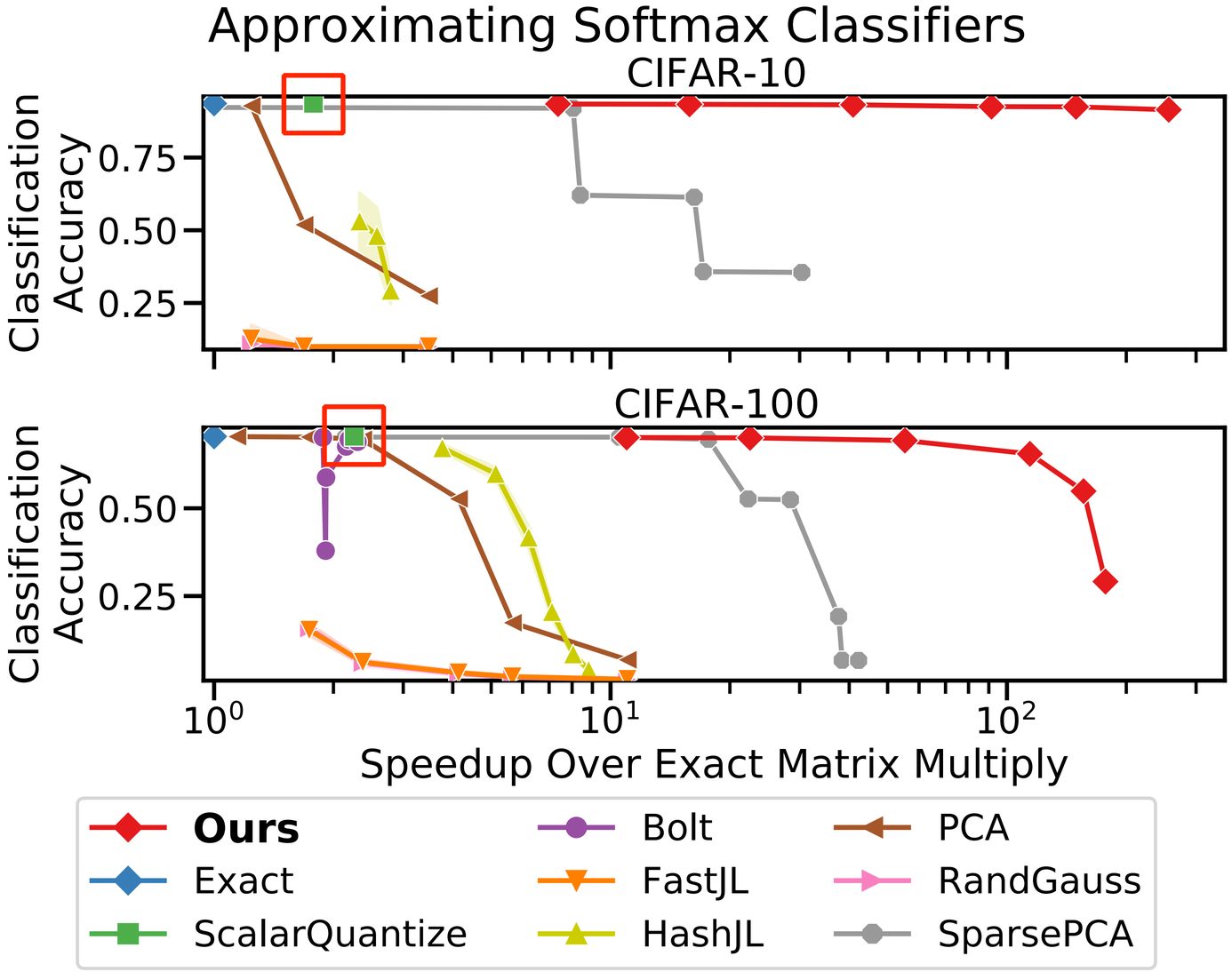
Our Second-Most-Impressive Result



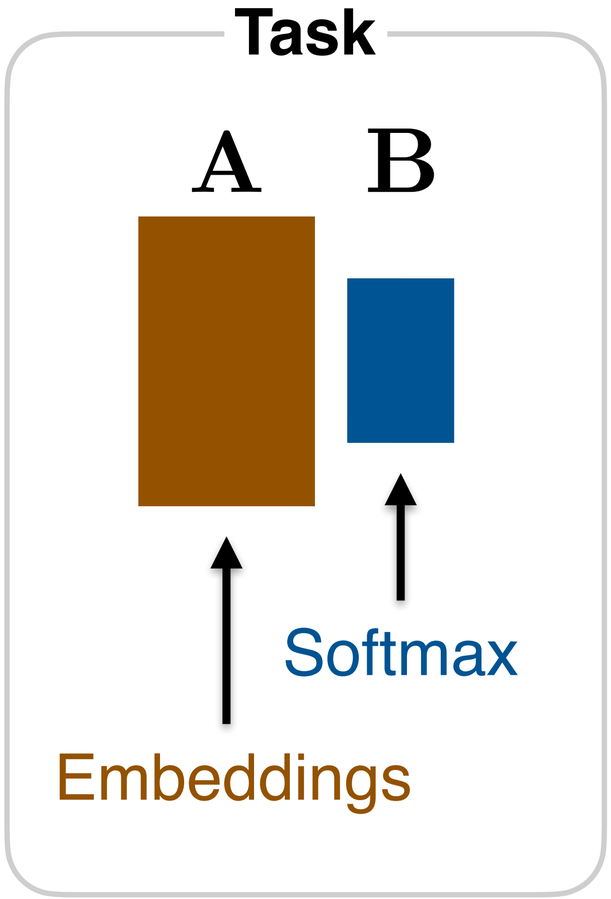


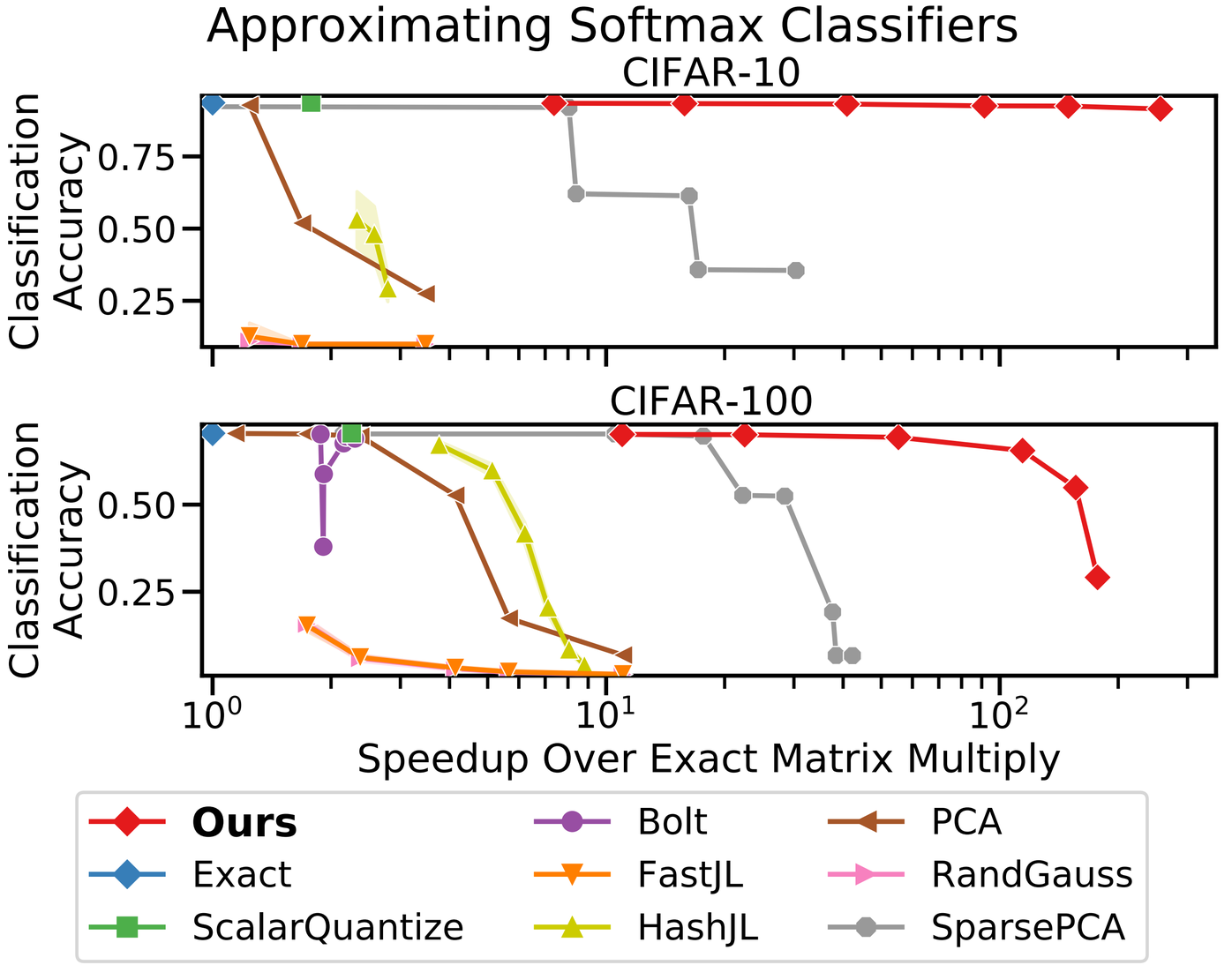
Our Second-Most-Impressive Result



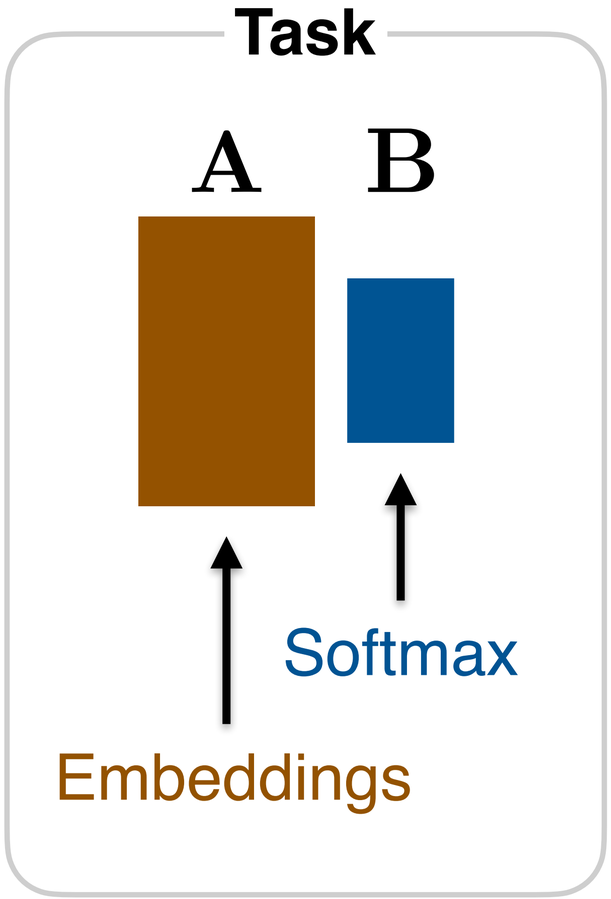


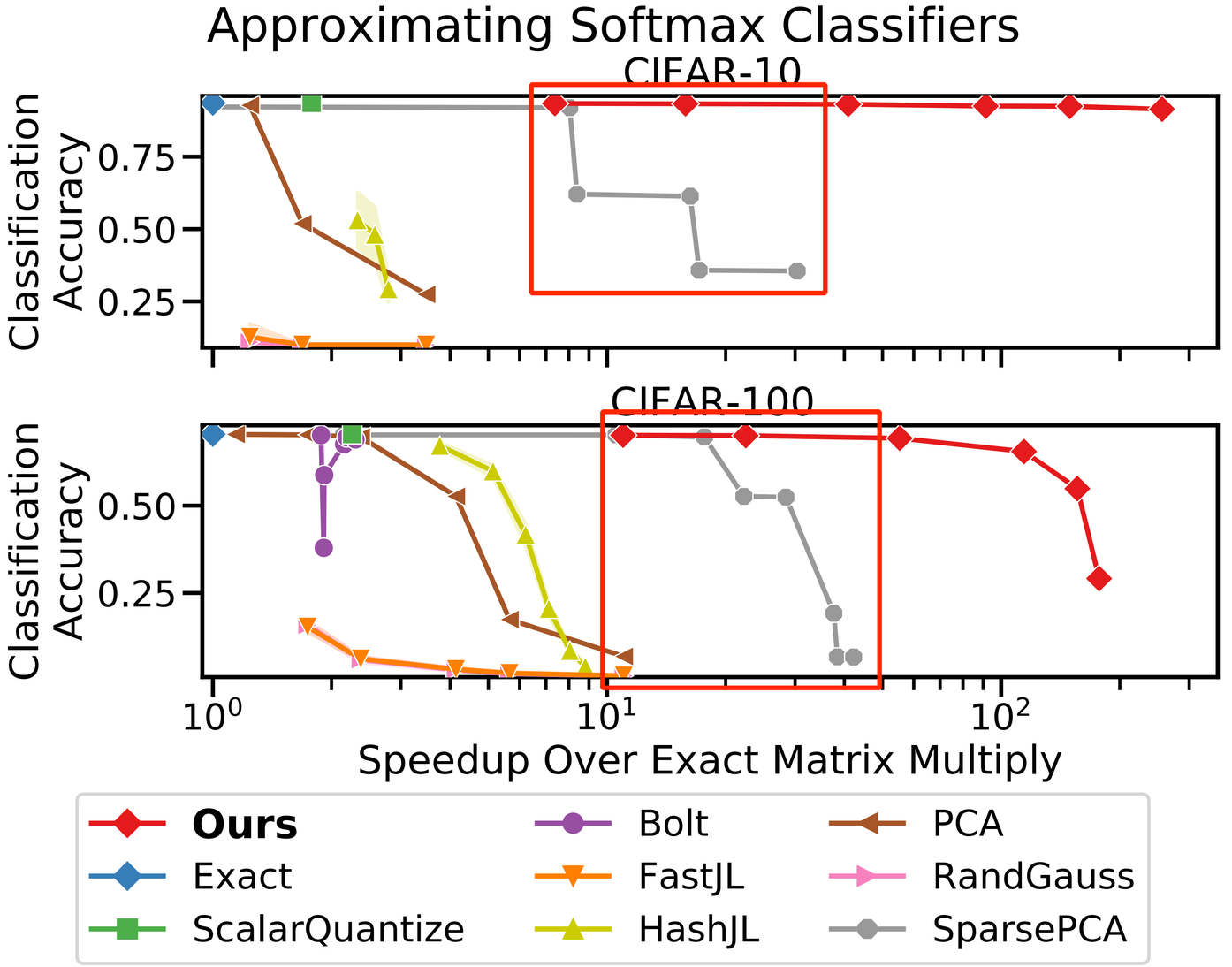
Our Second-Most-Impressive Result





Our Second-Most-Impressive Result





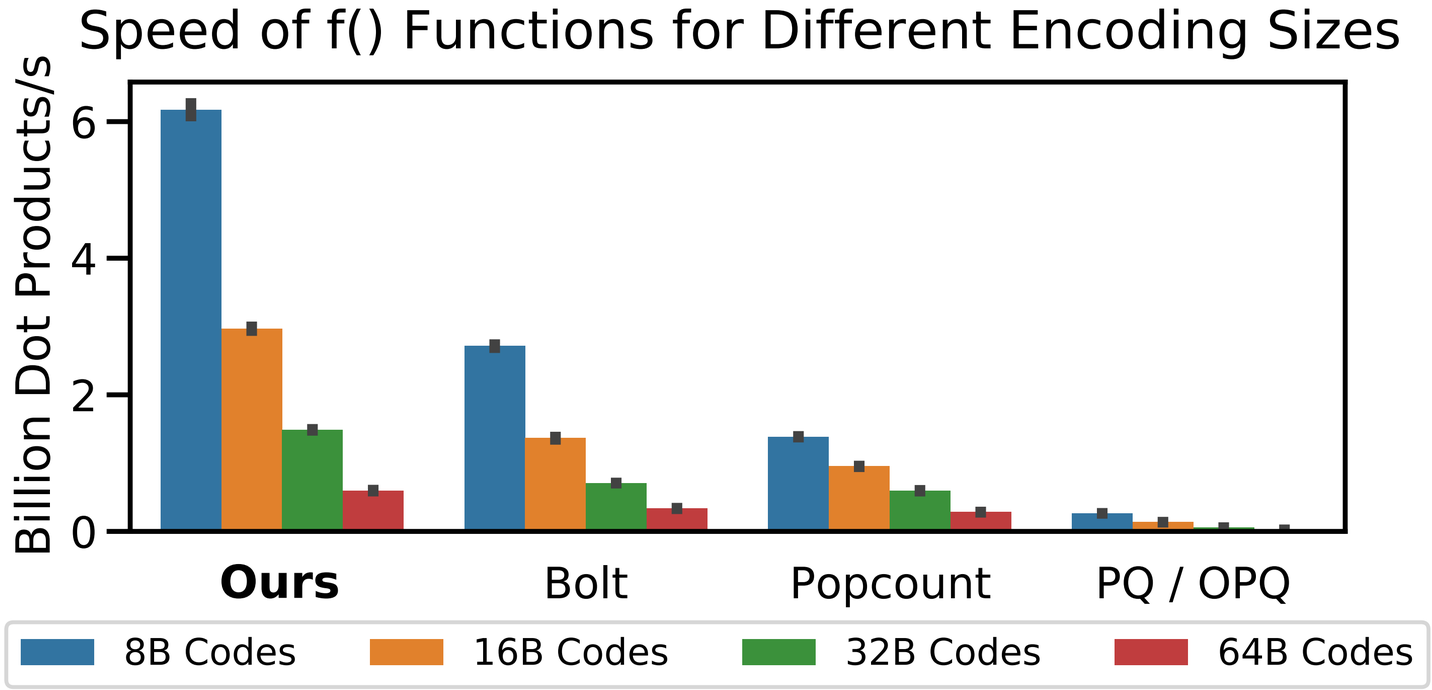
Key Ideas

1. Don’t make a vector sparse; make a sparse vector
2. “One 1” sparsity looks like dense ops, and so:
   A. Runs ridiculously fast on current CPUs
   B. Would be easy to hardware accelerate

Feel free to read more or reach out:

- Paper: “Multiplying Matrices Without Multiplying”
- Github: https://smarturl.it/maddness
- Email: davis@mosaicml.com
- Twitter: @davisblalock

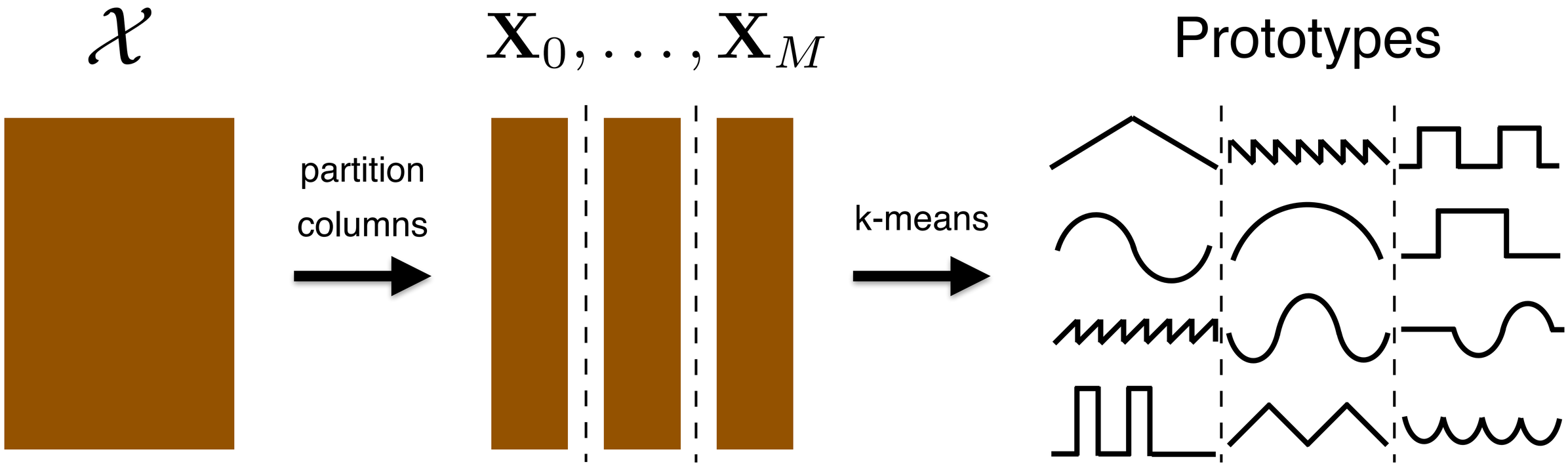
Our table lookups are really, really fast



- Up to two 256-element dot products *per cycle* in one thread
- Speedup comes from more compact representation
  - Eg., replacing 256 * 4 = 1024B with 8B; **128x space reduction**
  - Hardware supports multiply-adds 4x better than our table lookups

Background: Product Quantization[1]

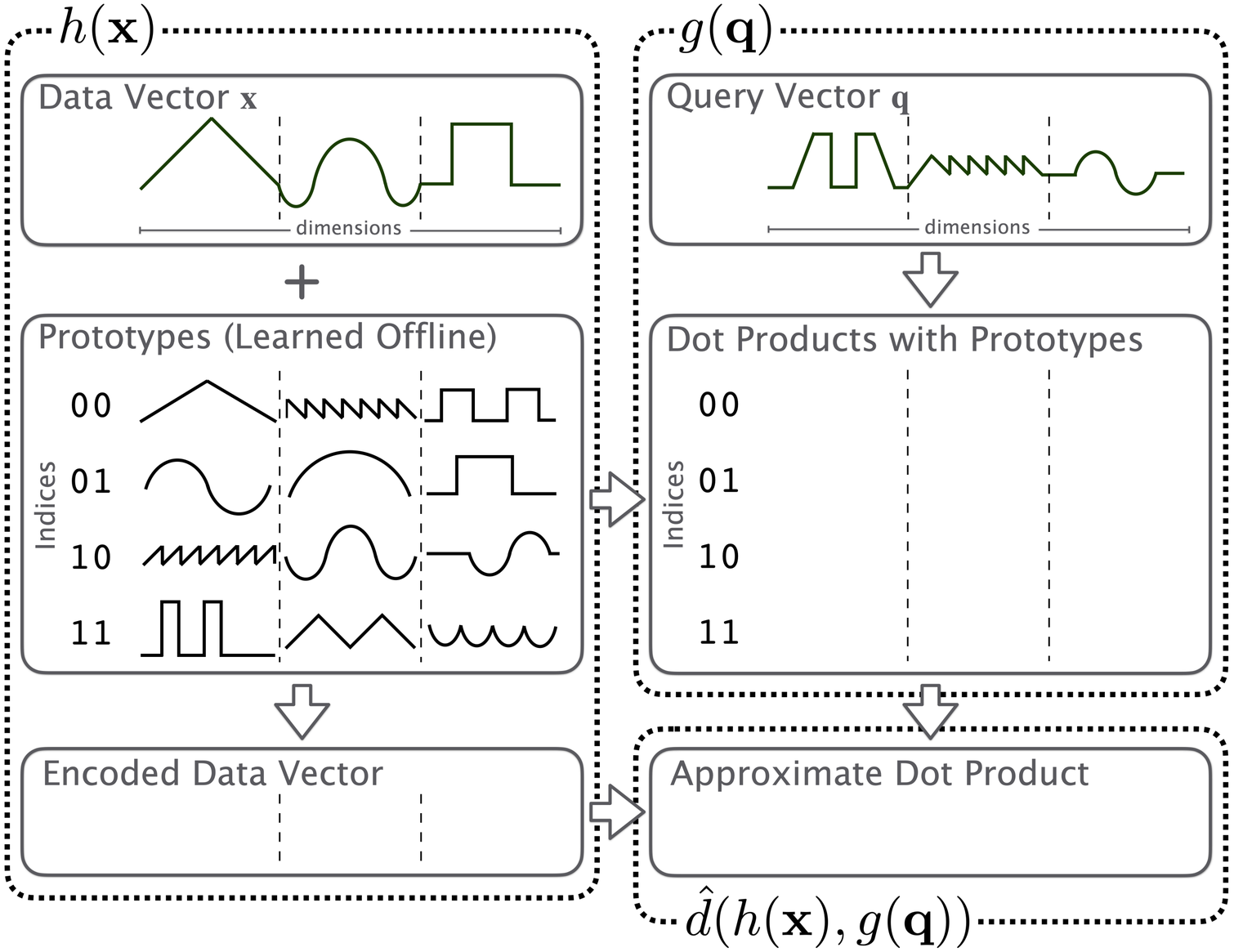
Offline training phase: run k-means in disjoint subspaces to produce prototypes



[1] “Product quantization for nearest neighbor search.” Jegou et al. 2011

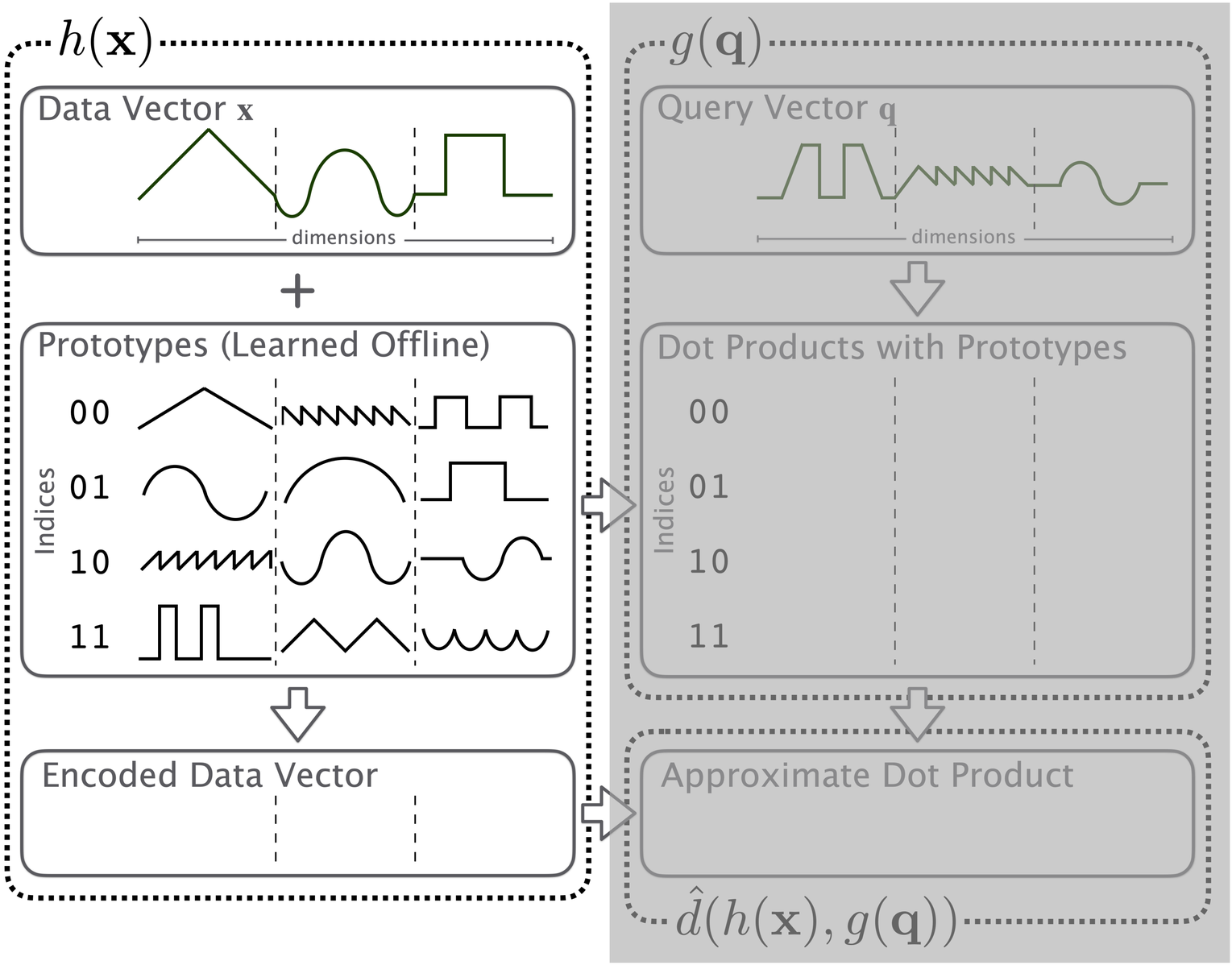
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



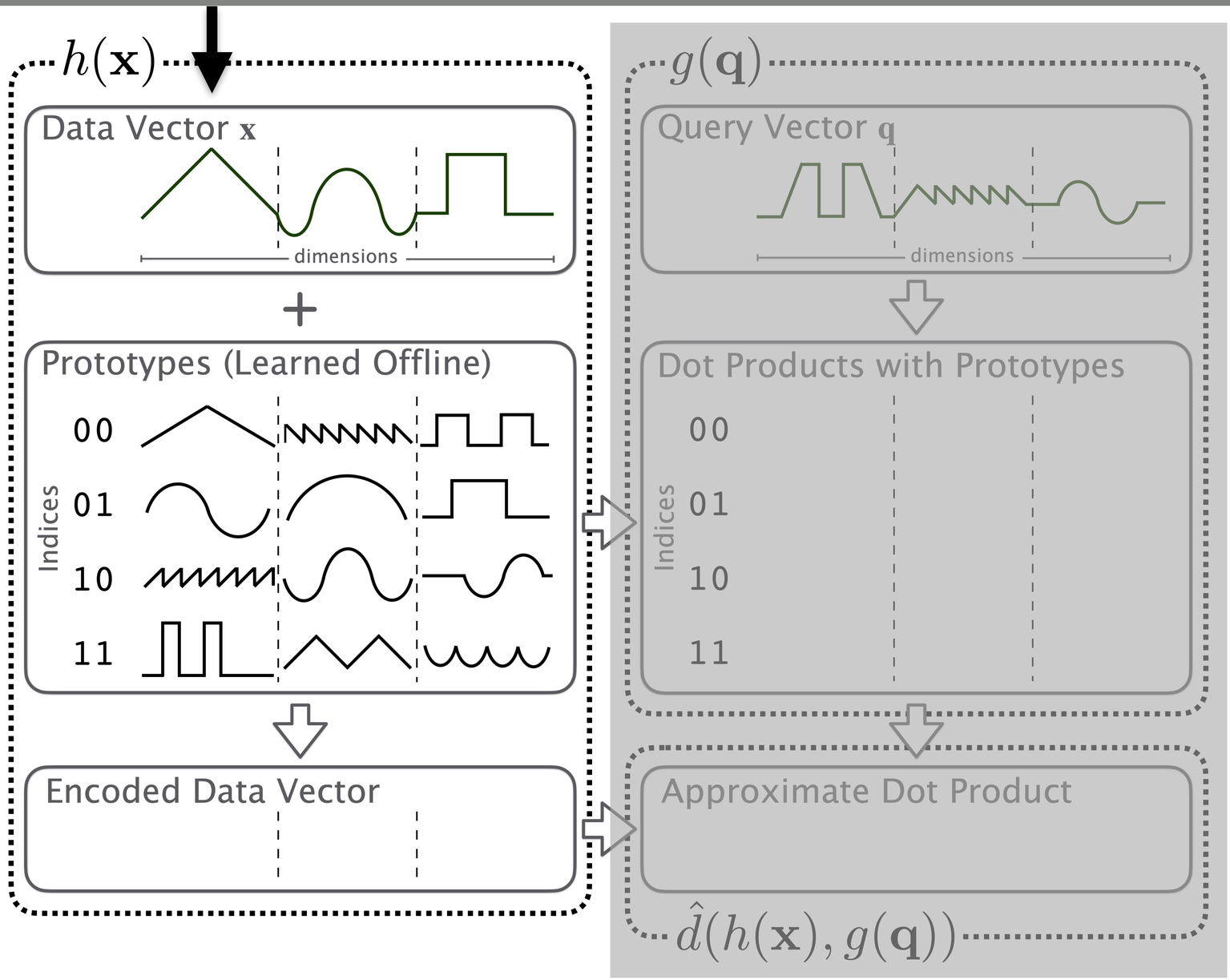
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



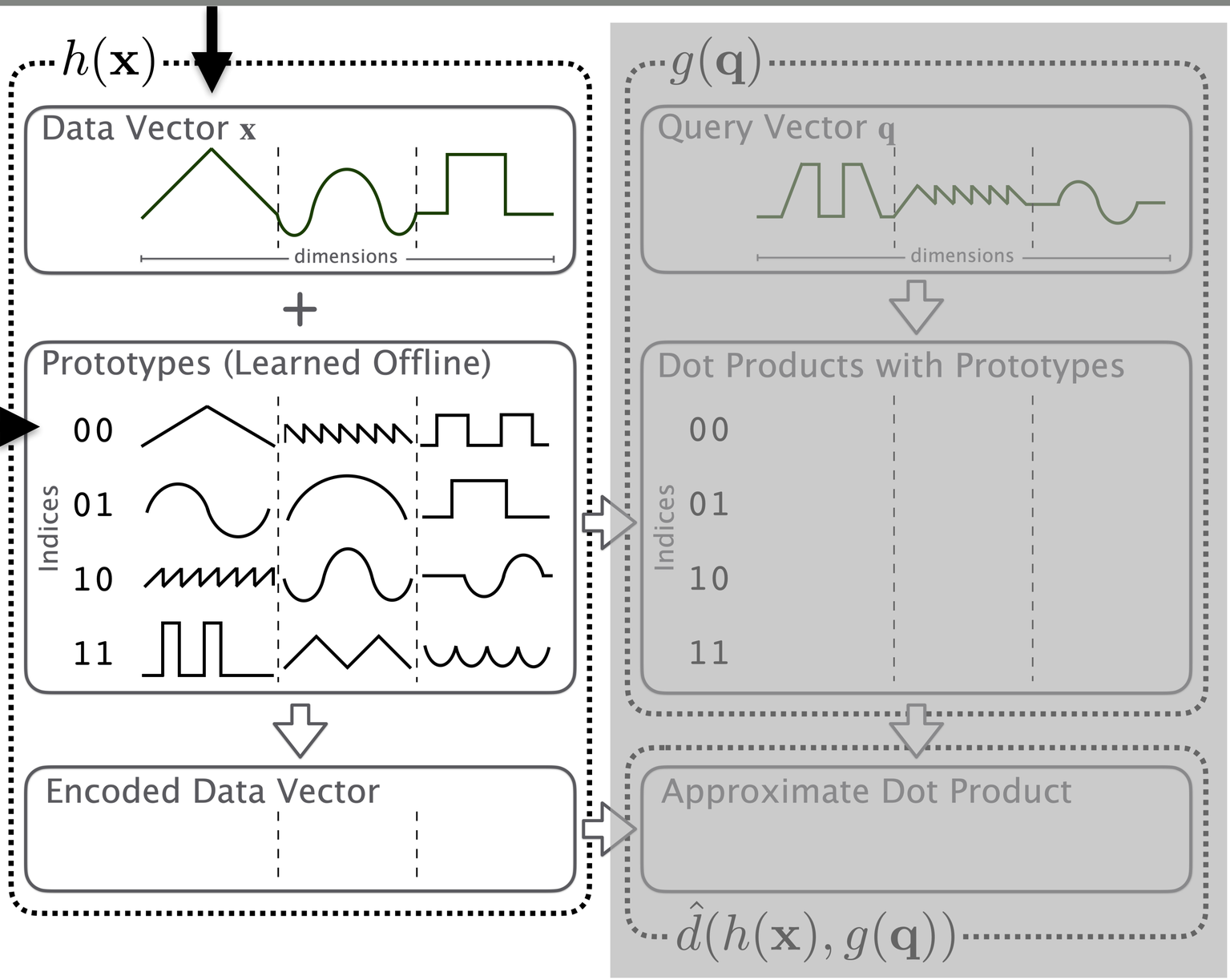
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



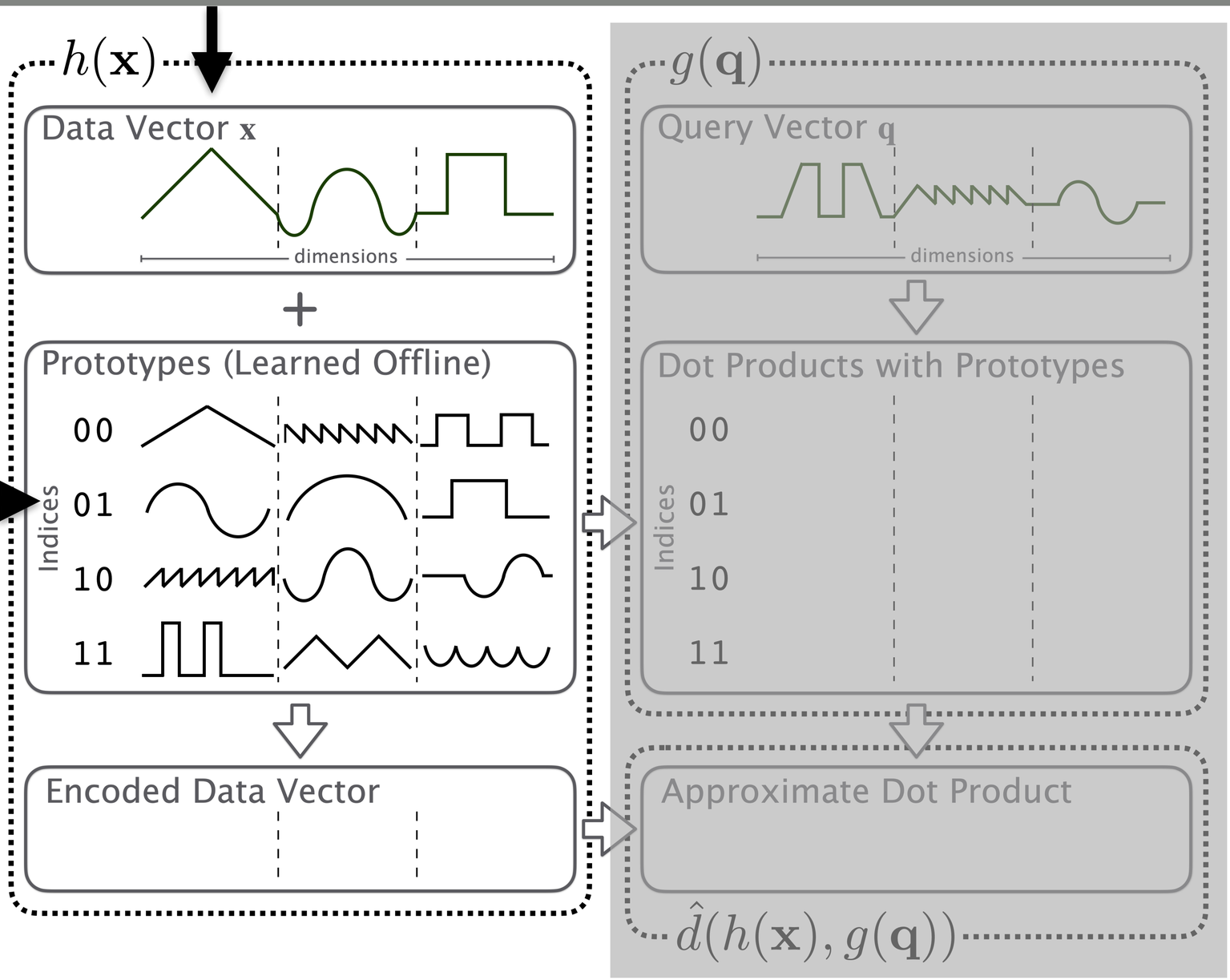
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



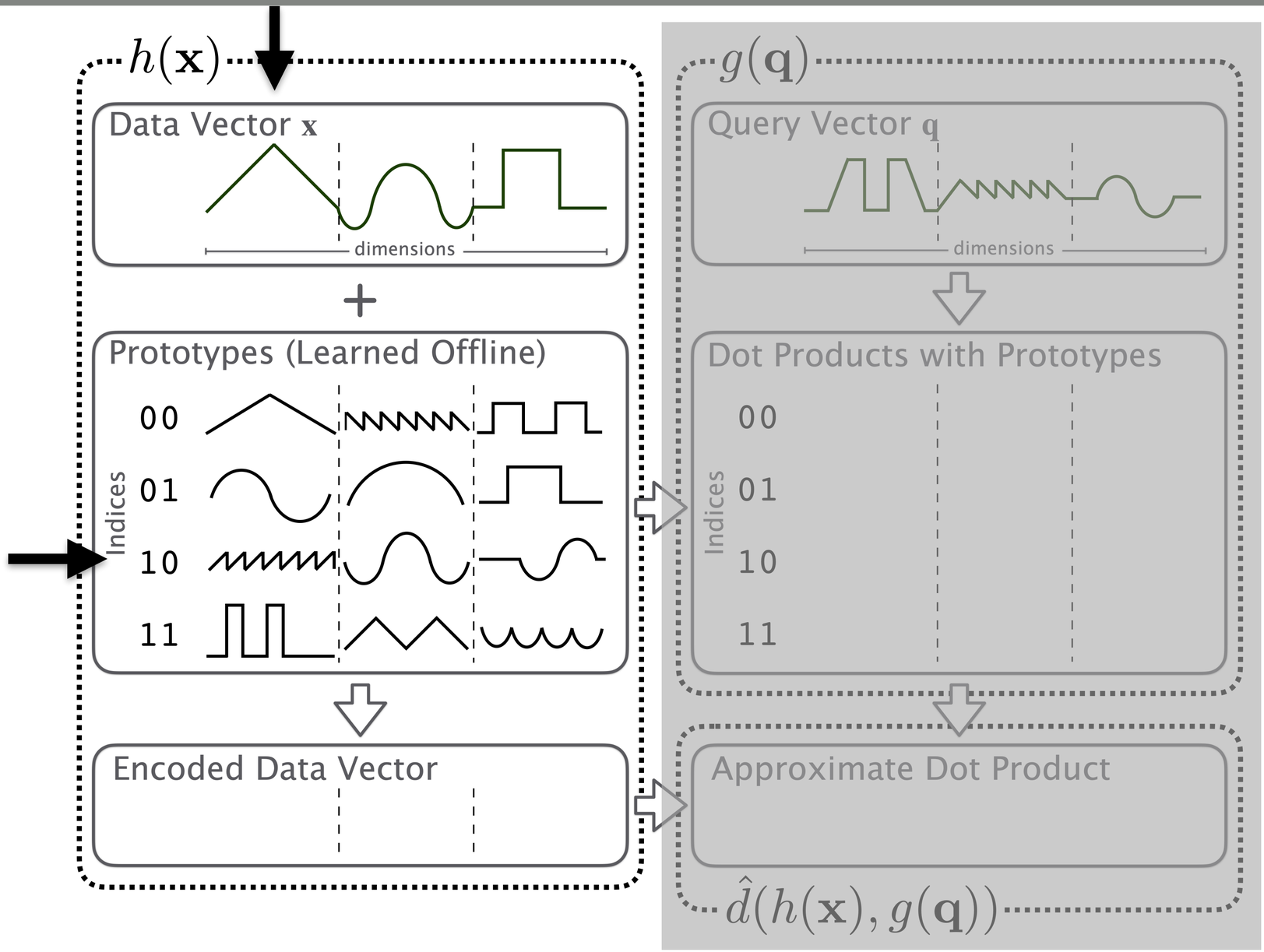
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



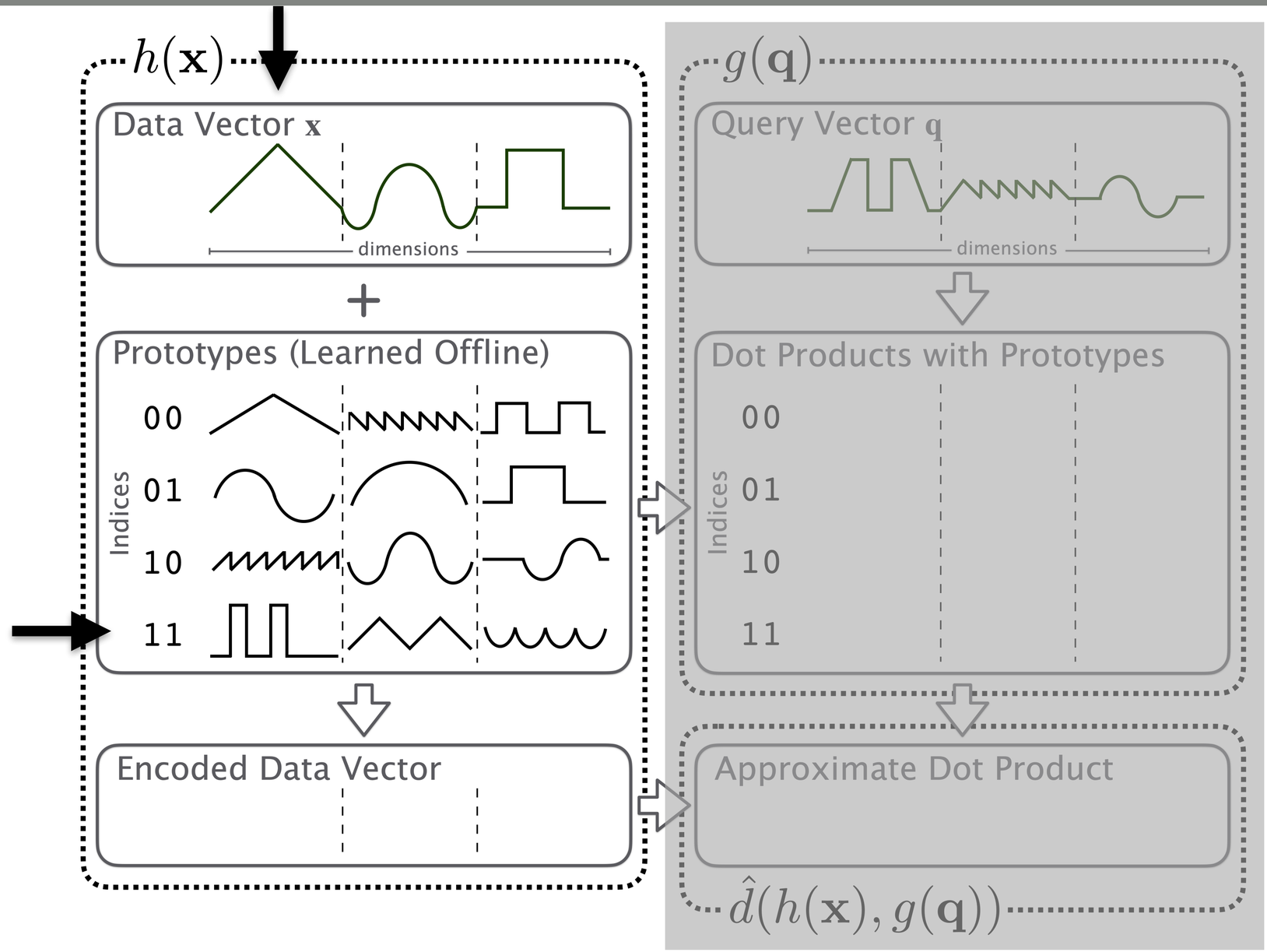
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



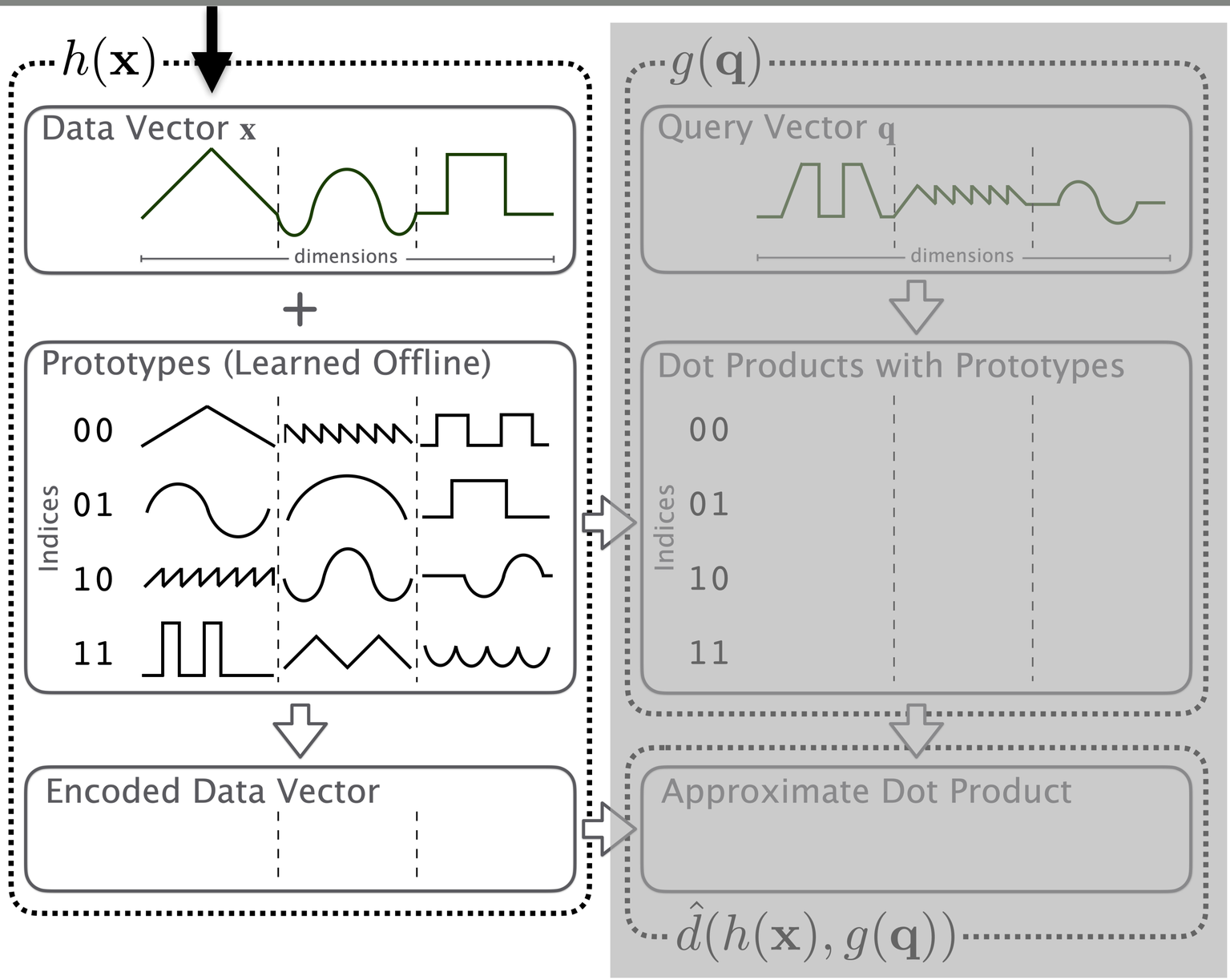
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



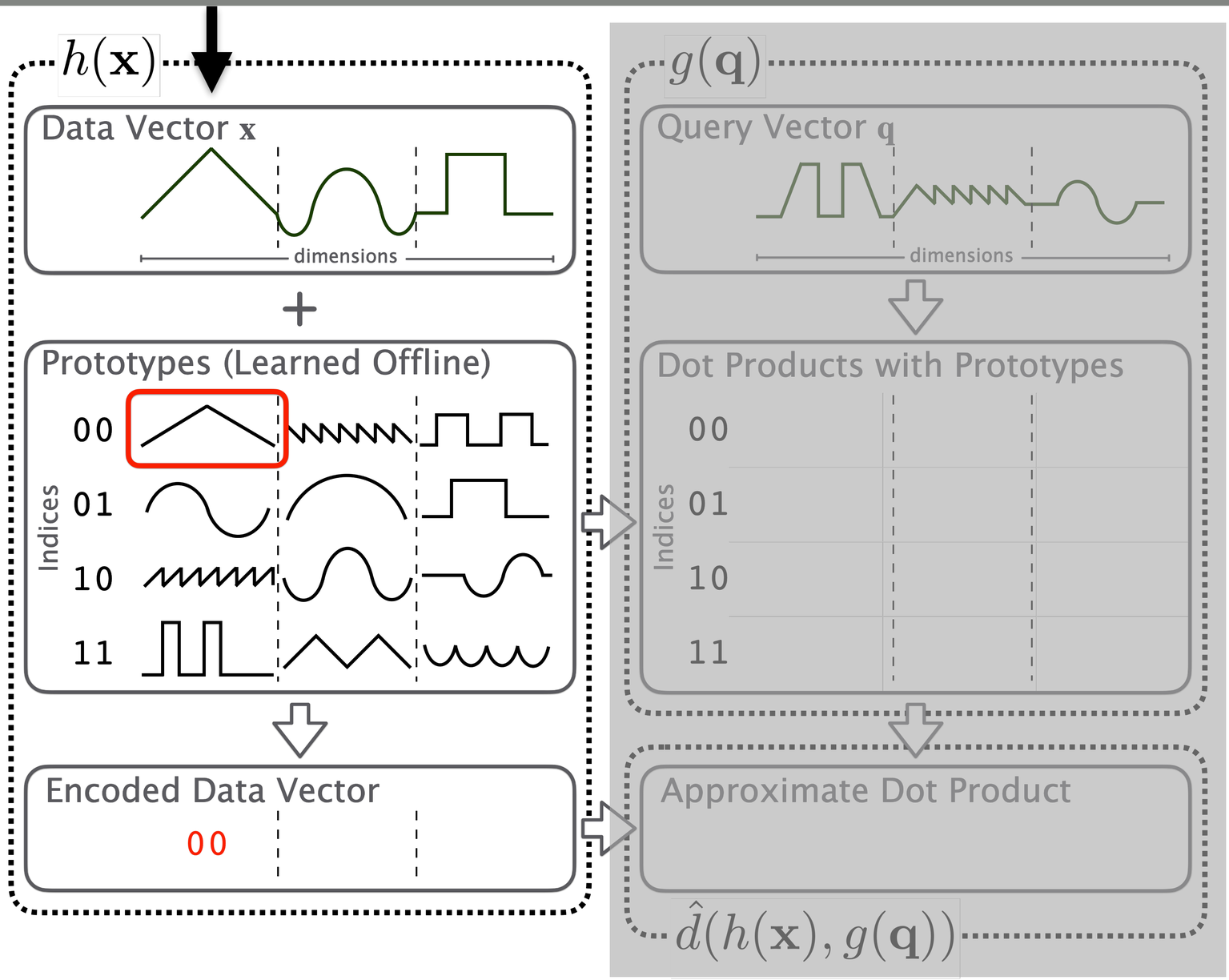
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



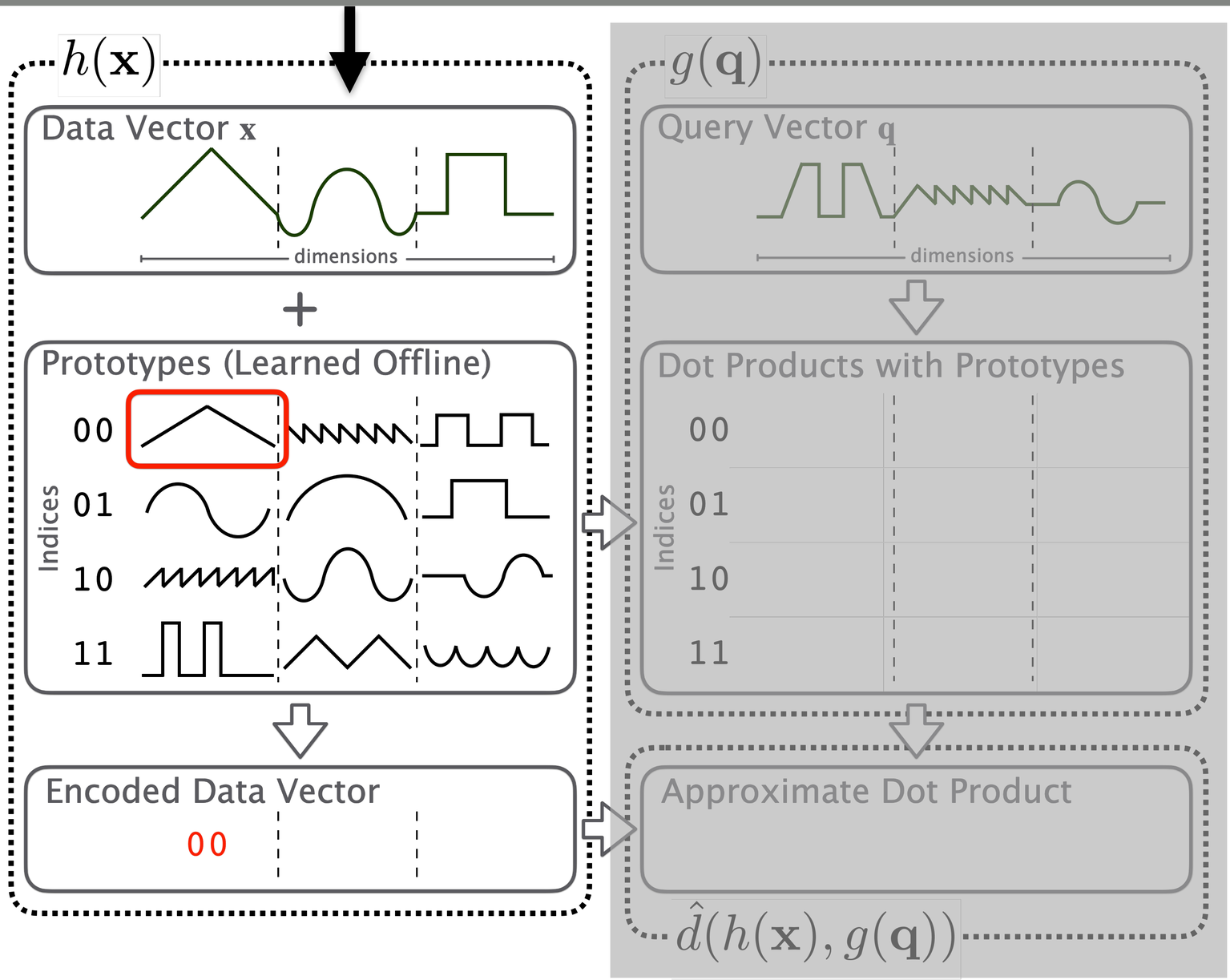
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



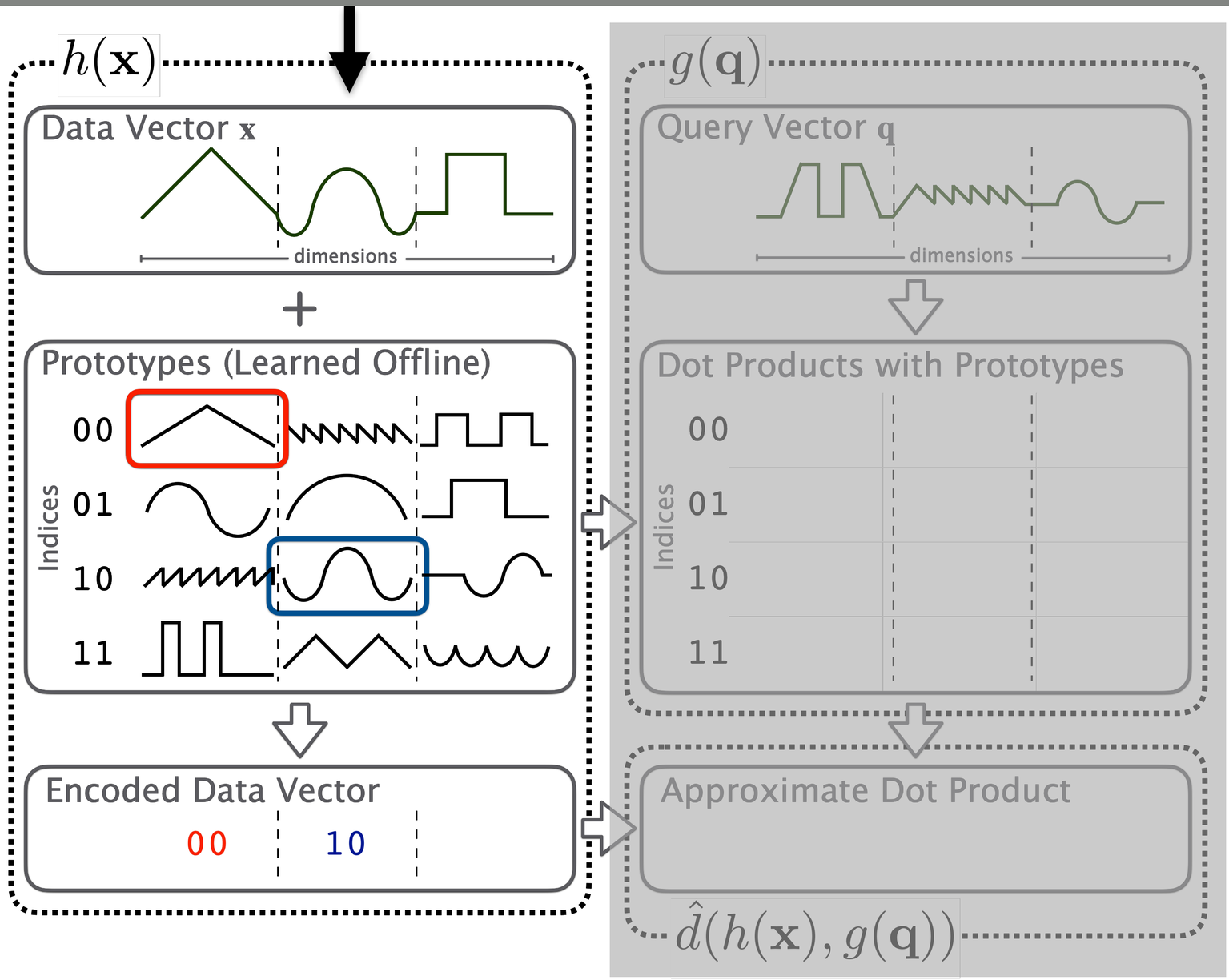
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



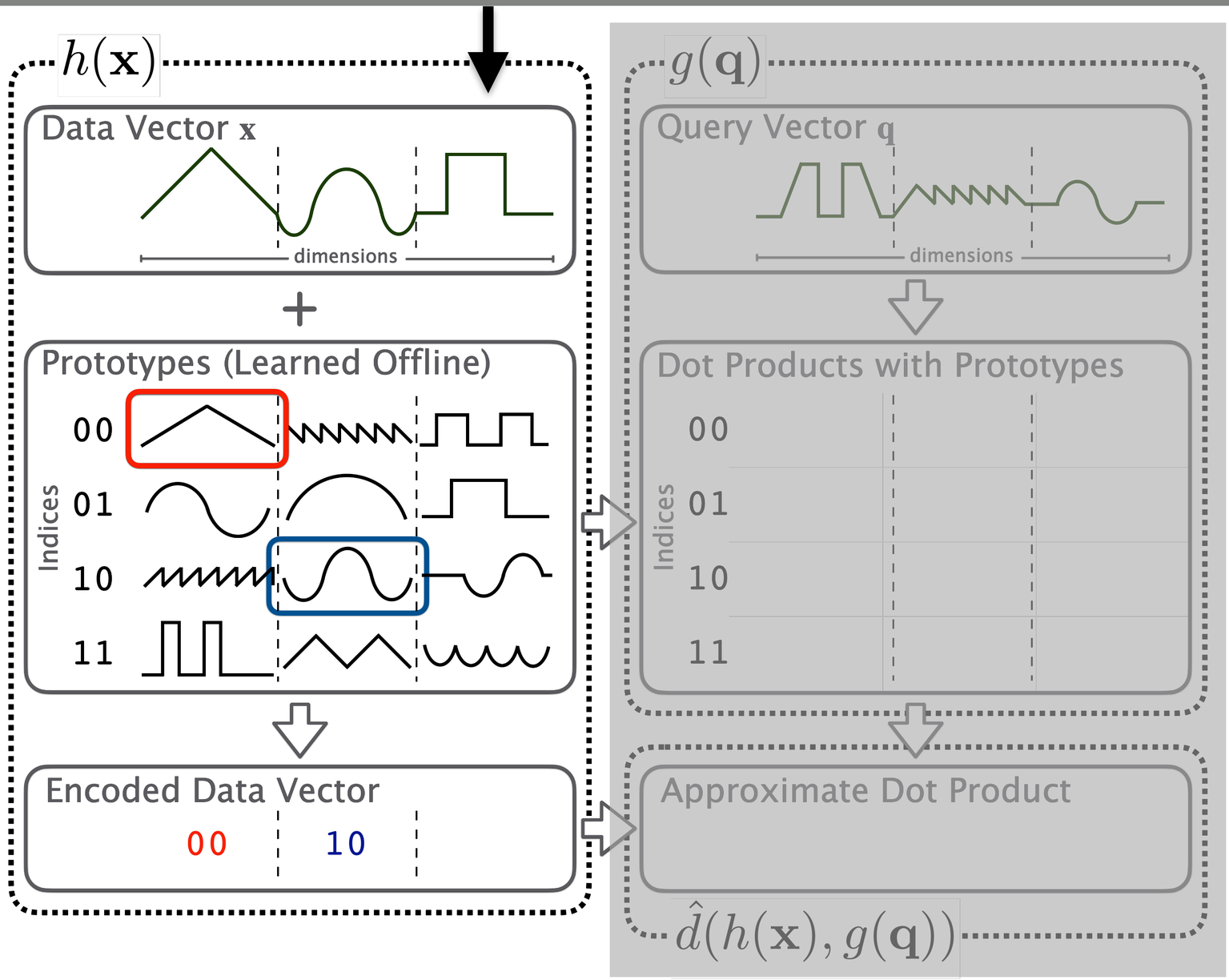
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



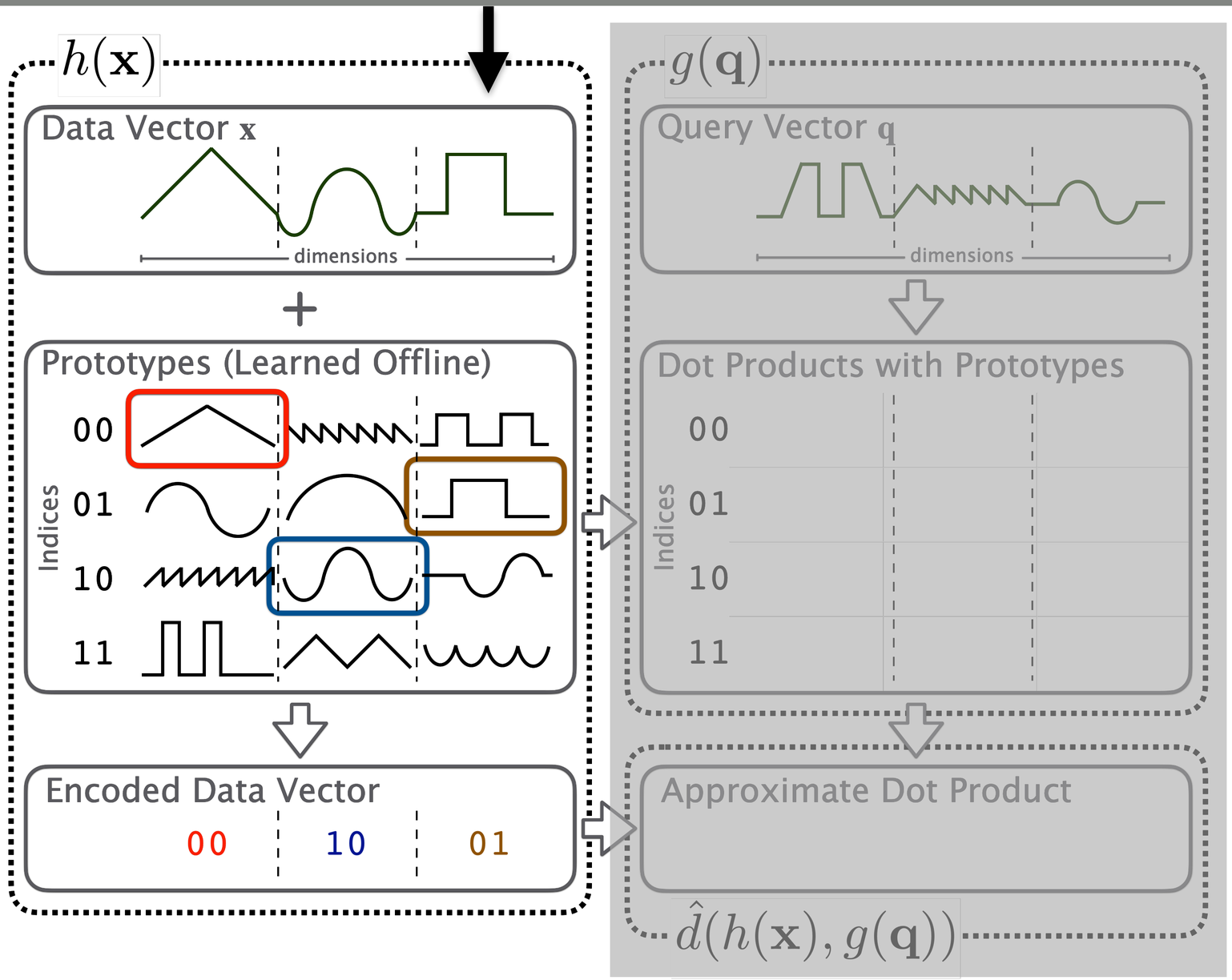
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



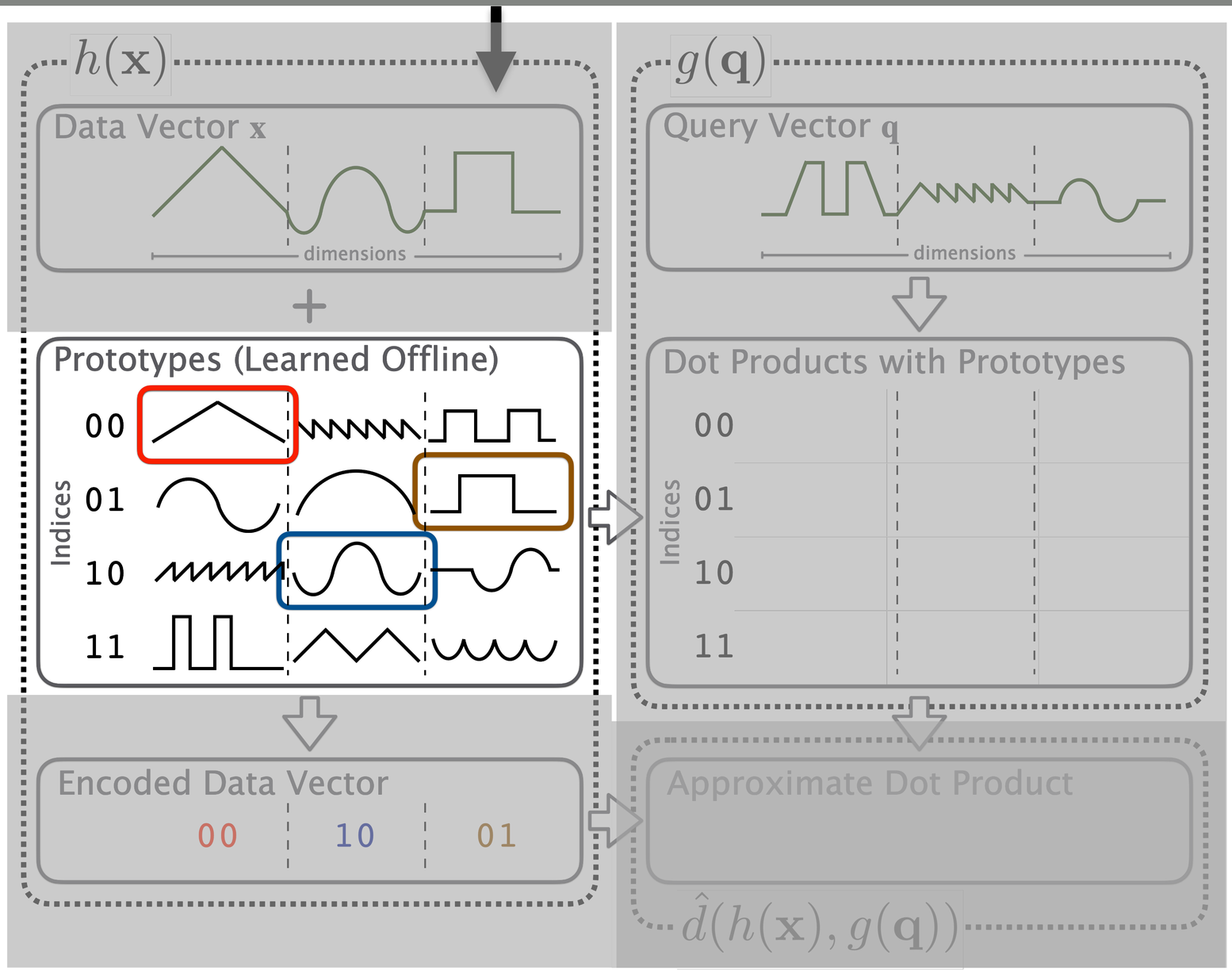
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



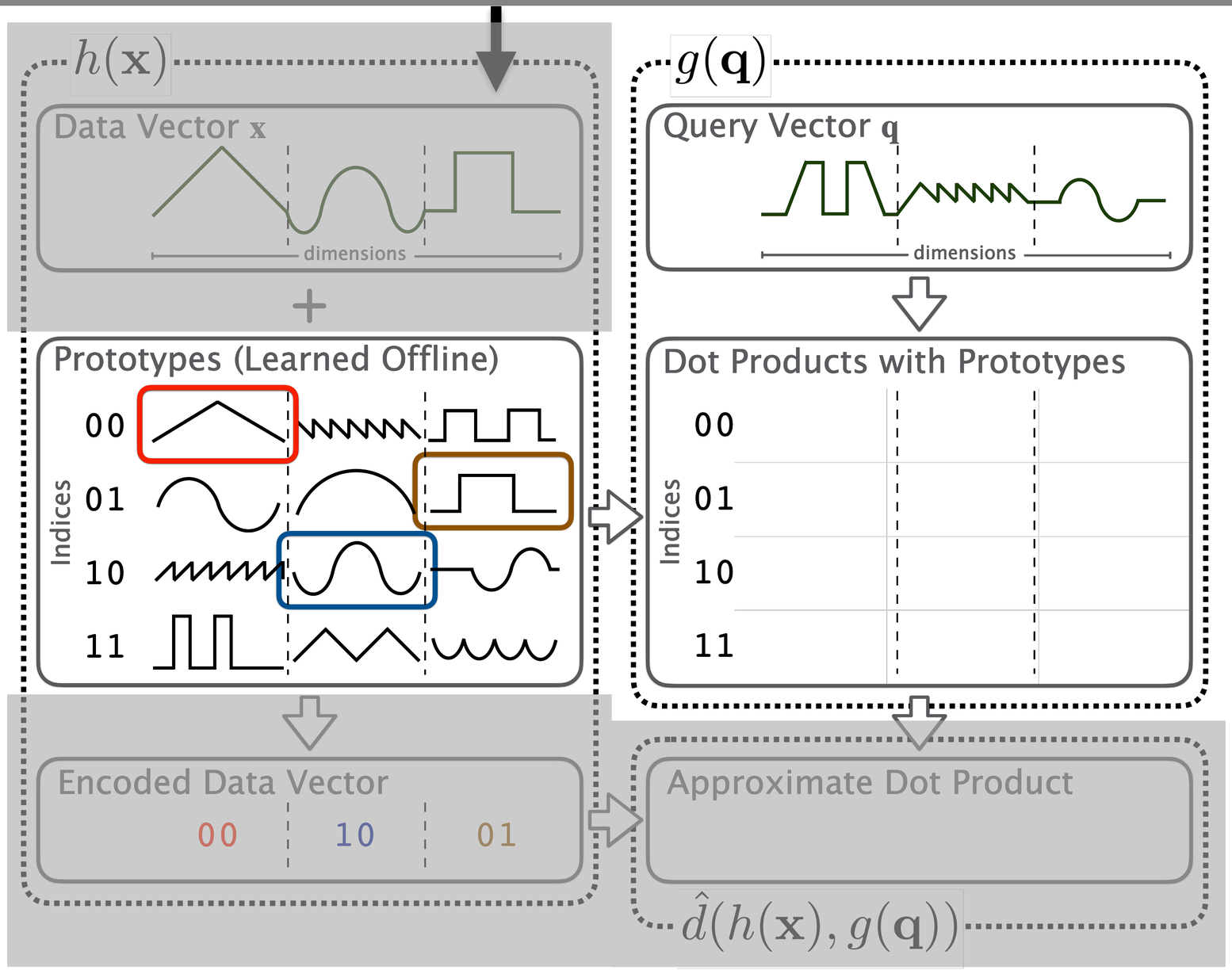
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



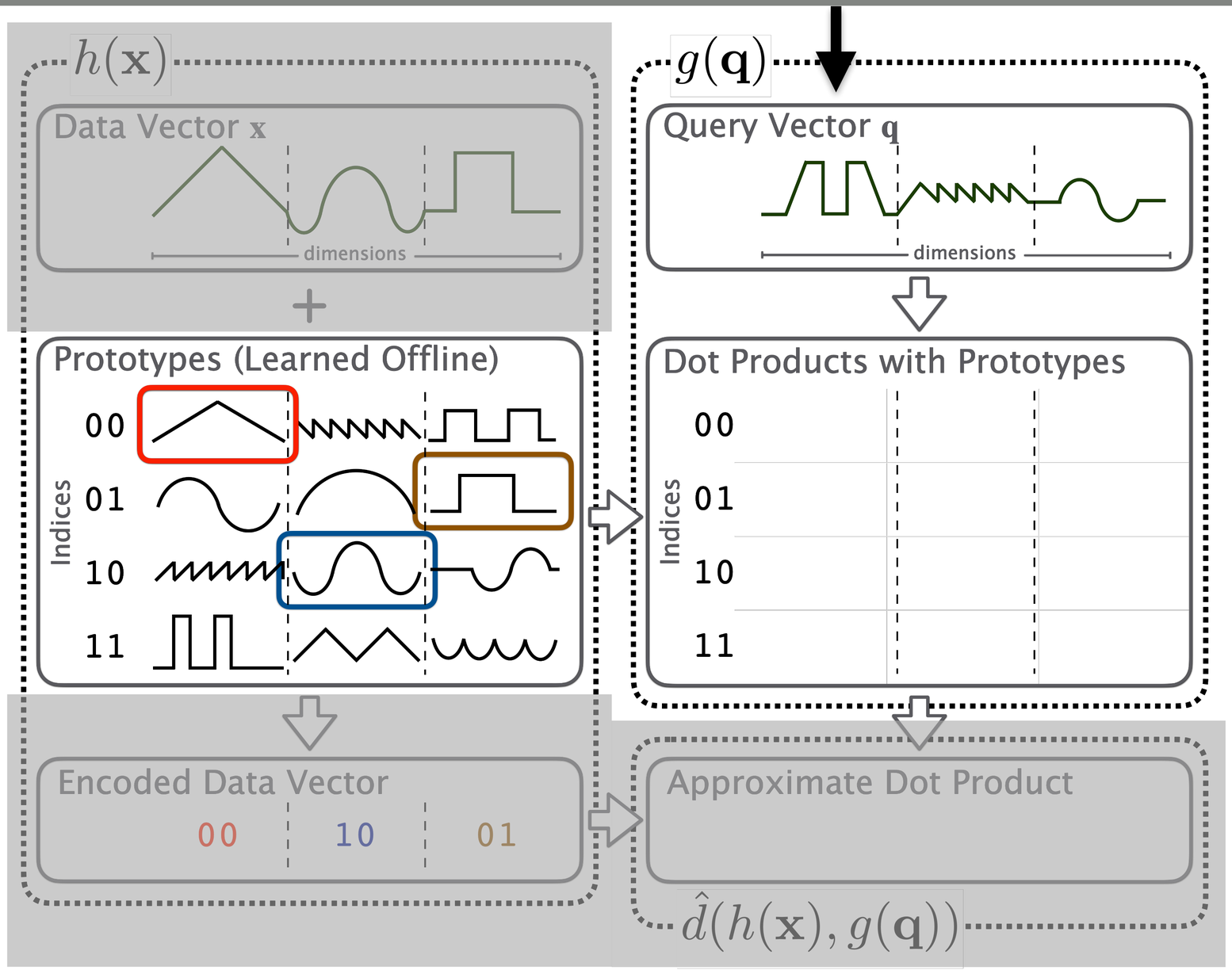
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



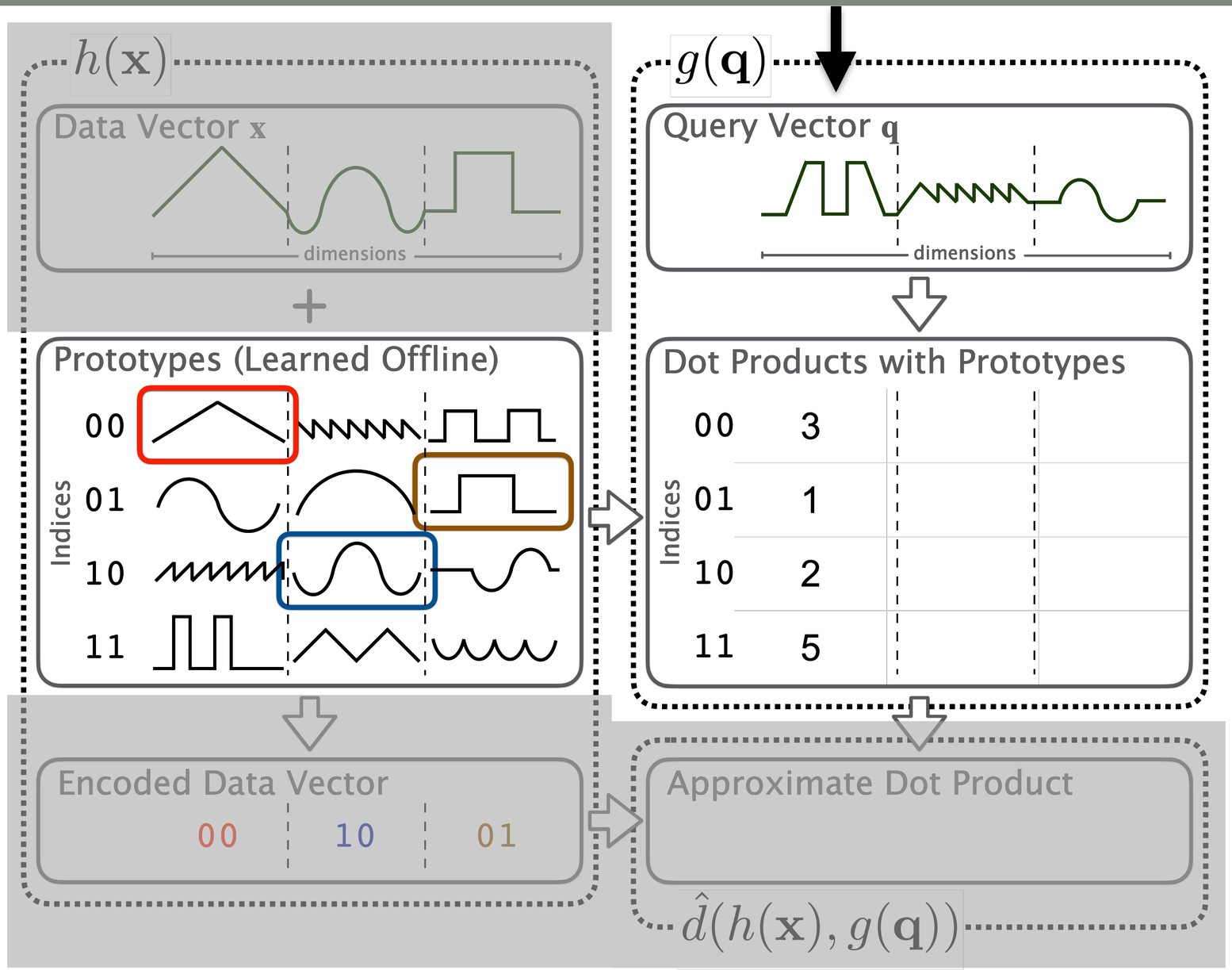
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



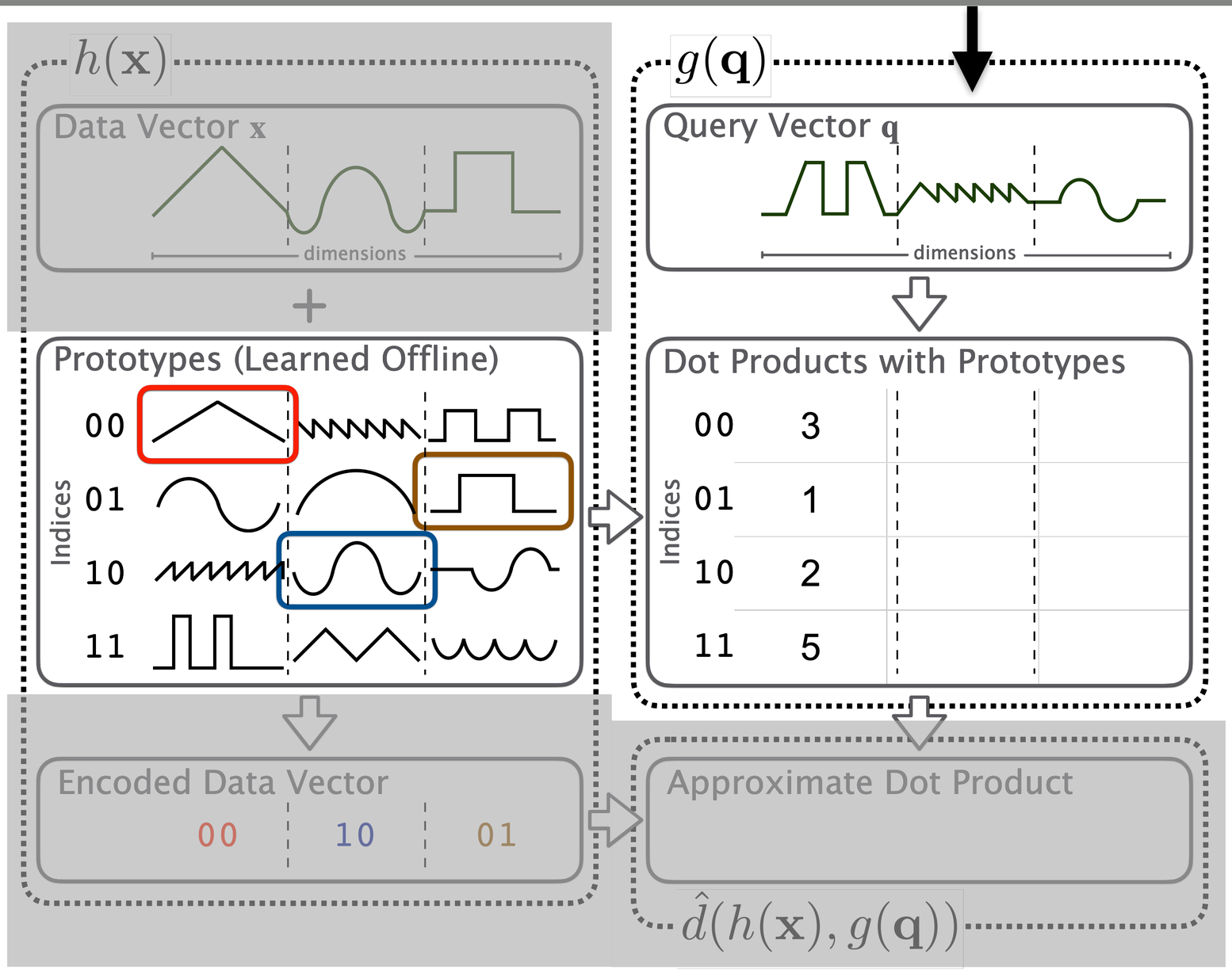
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



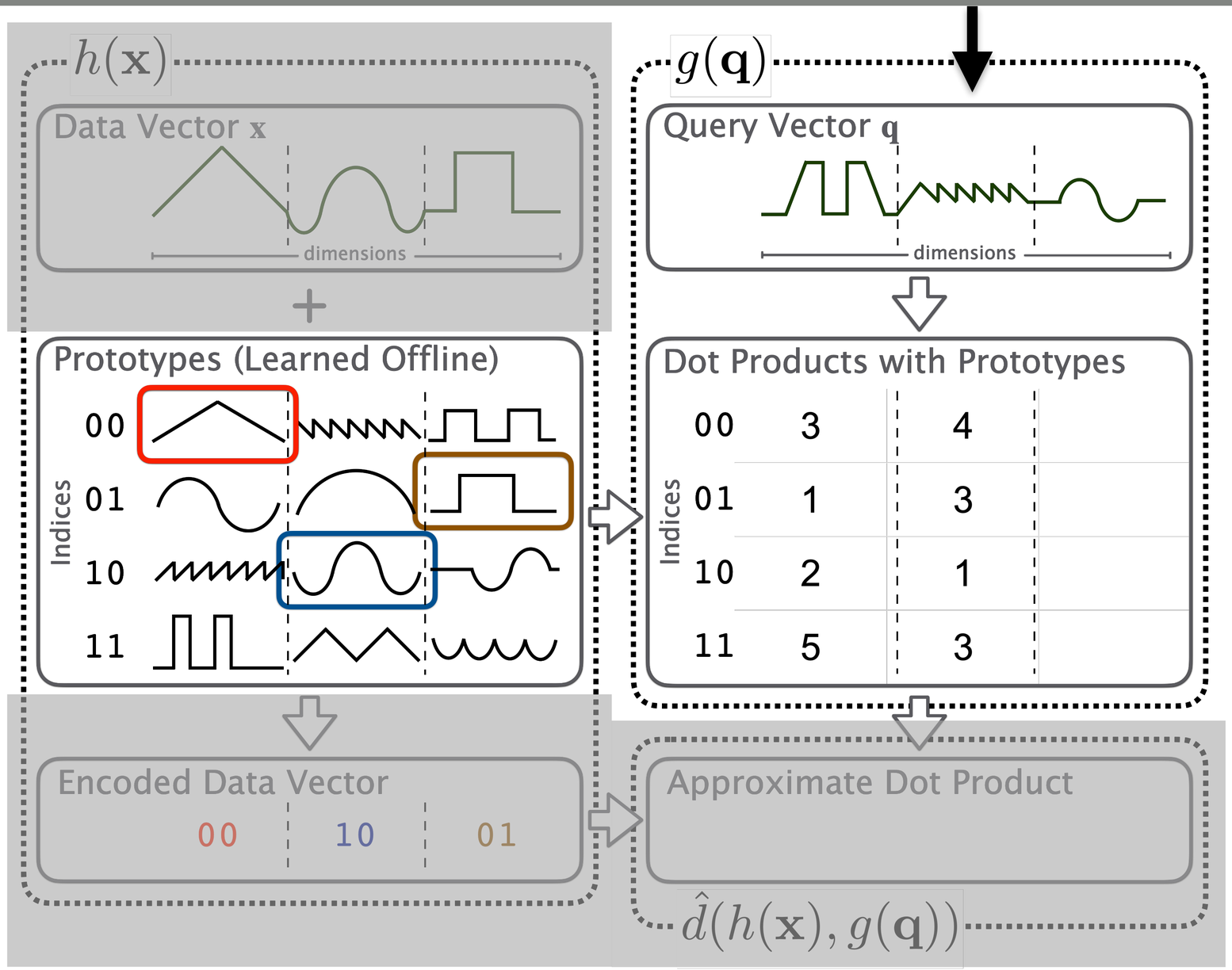
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



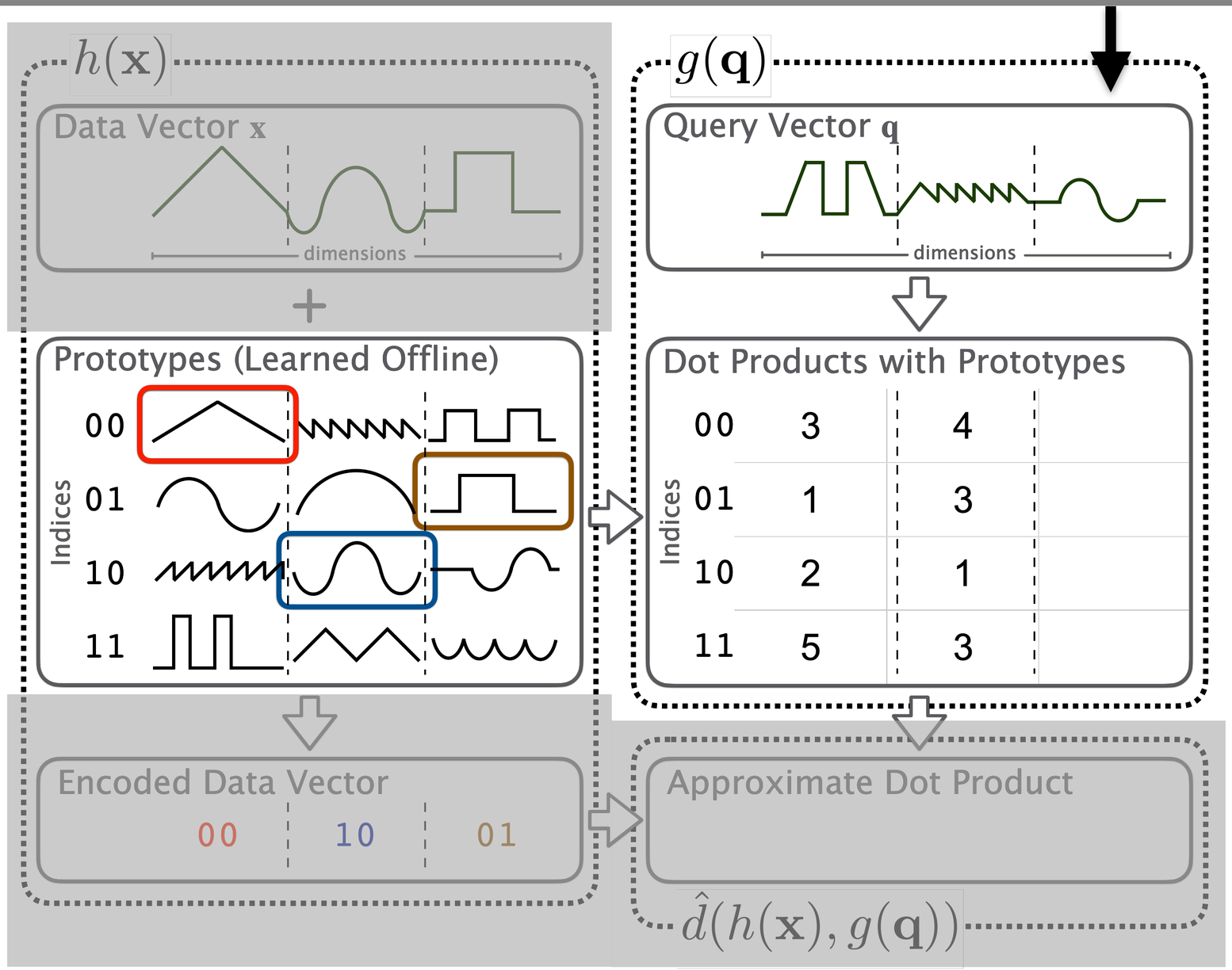
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



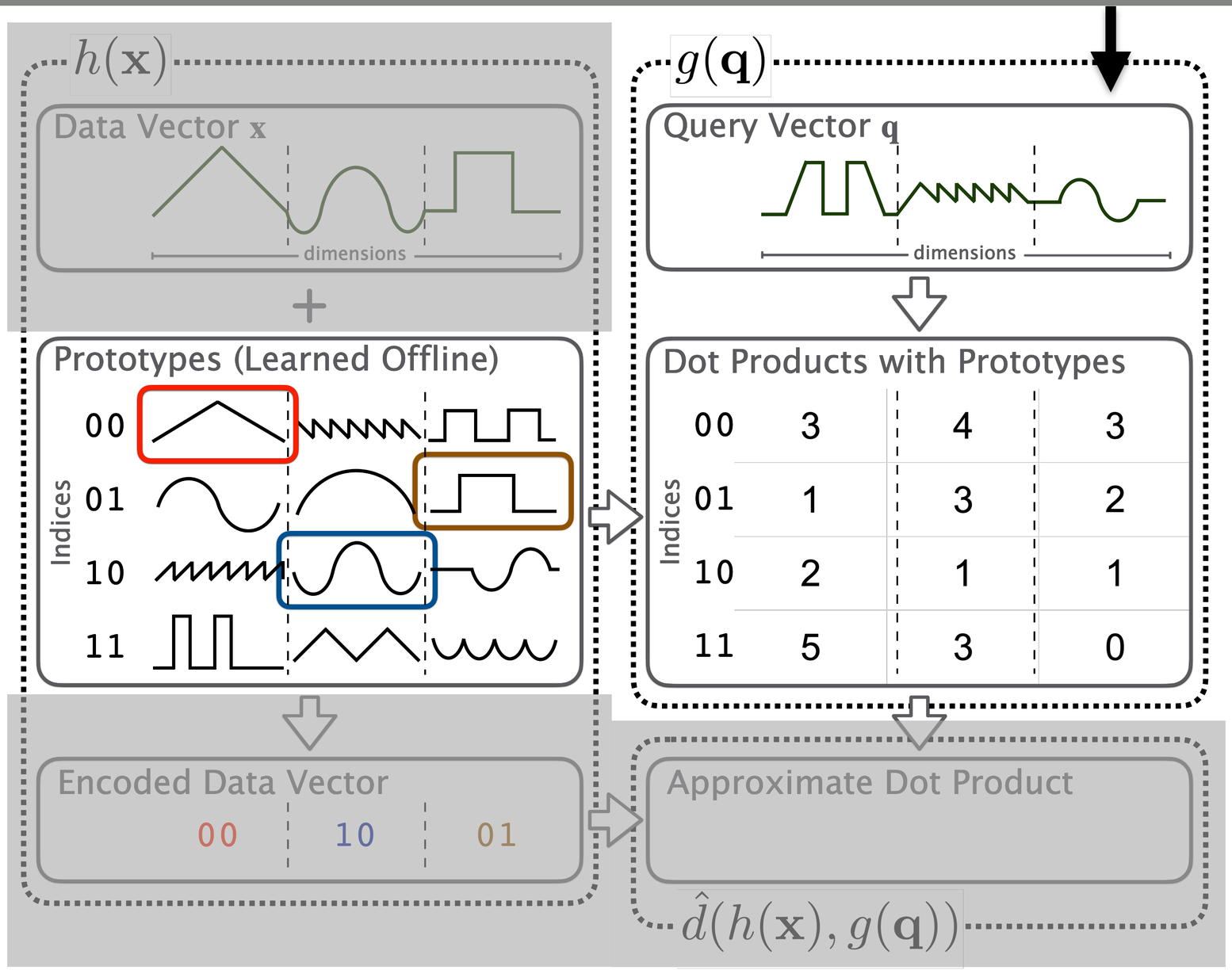
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



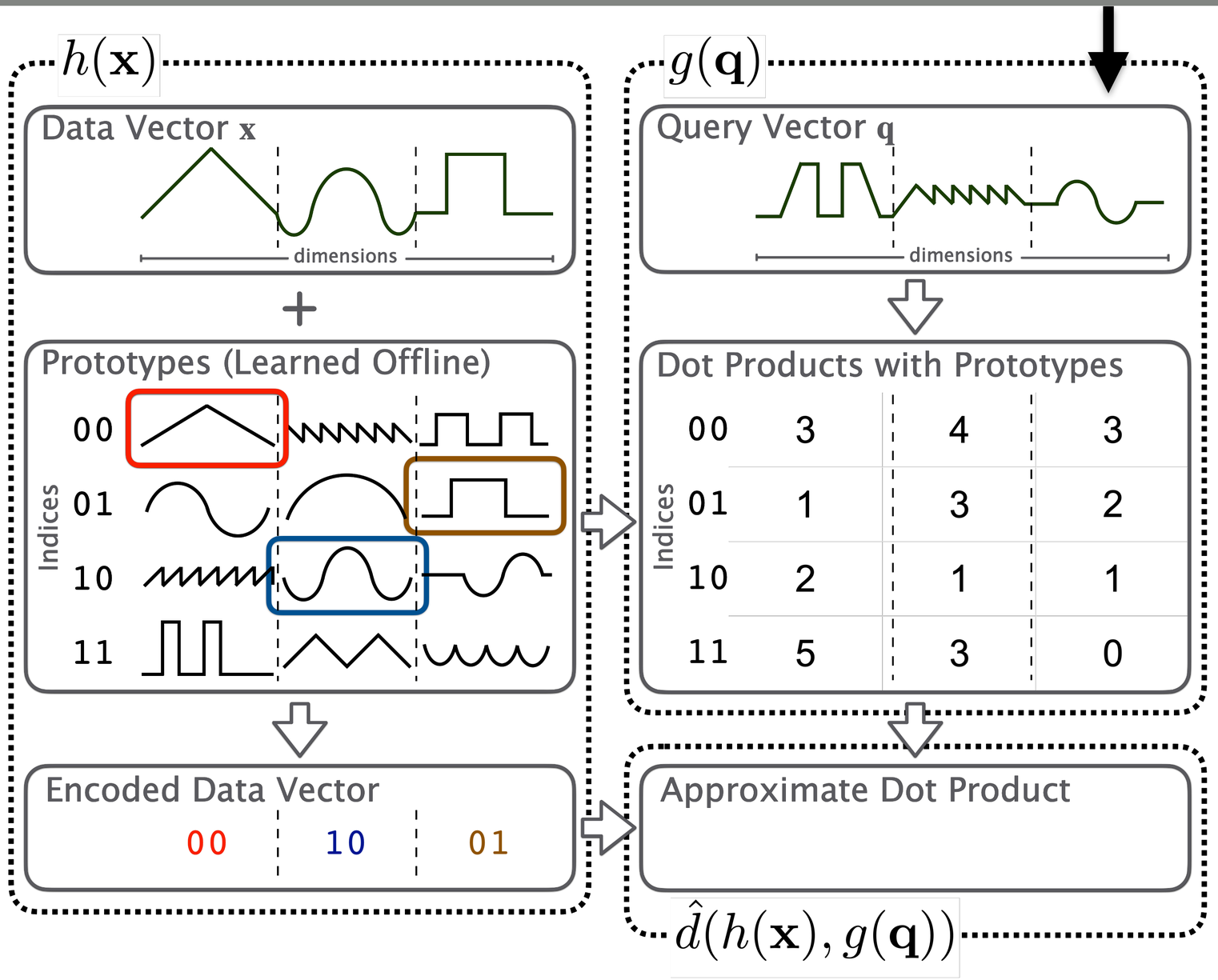
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



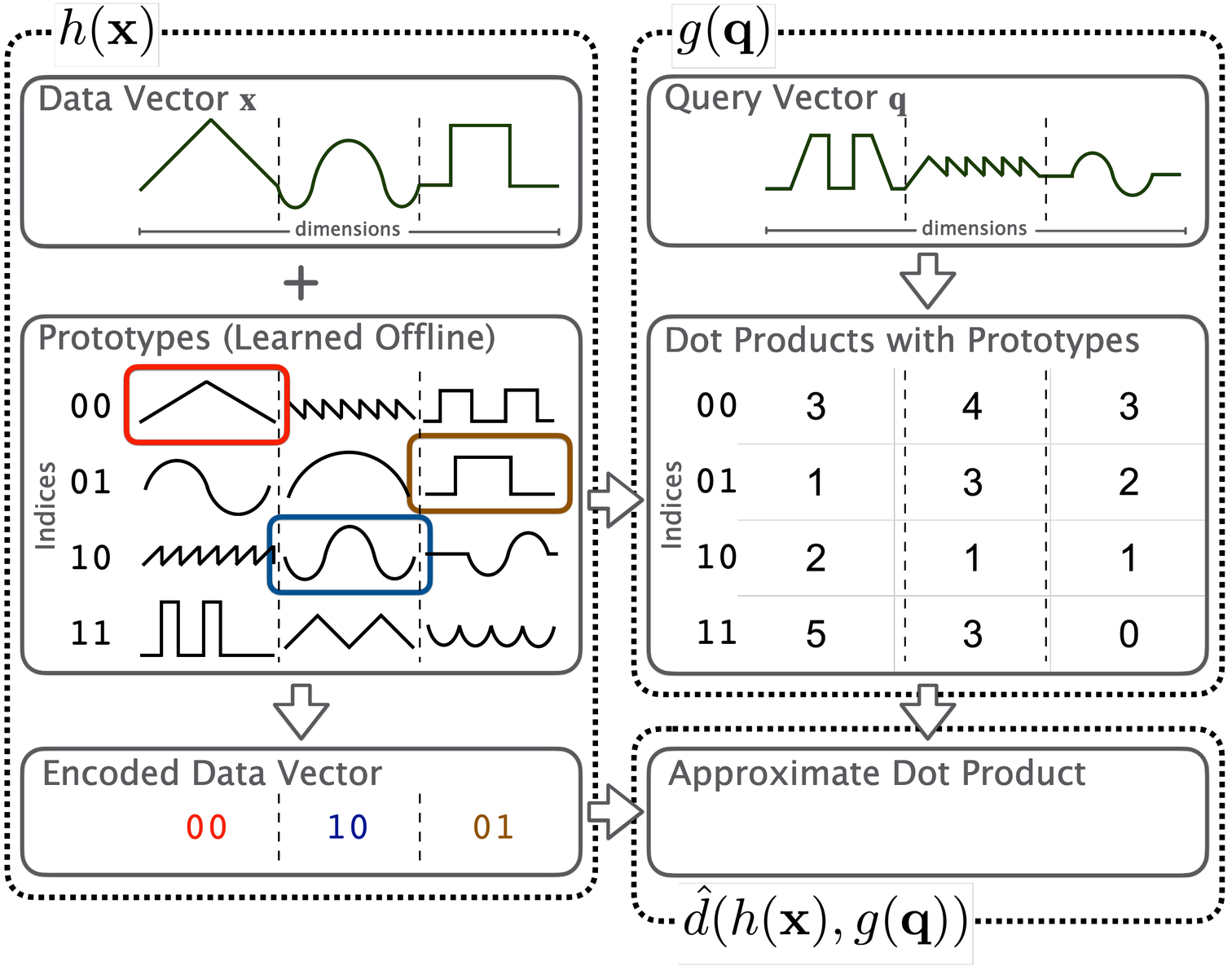
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



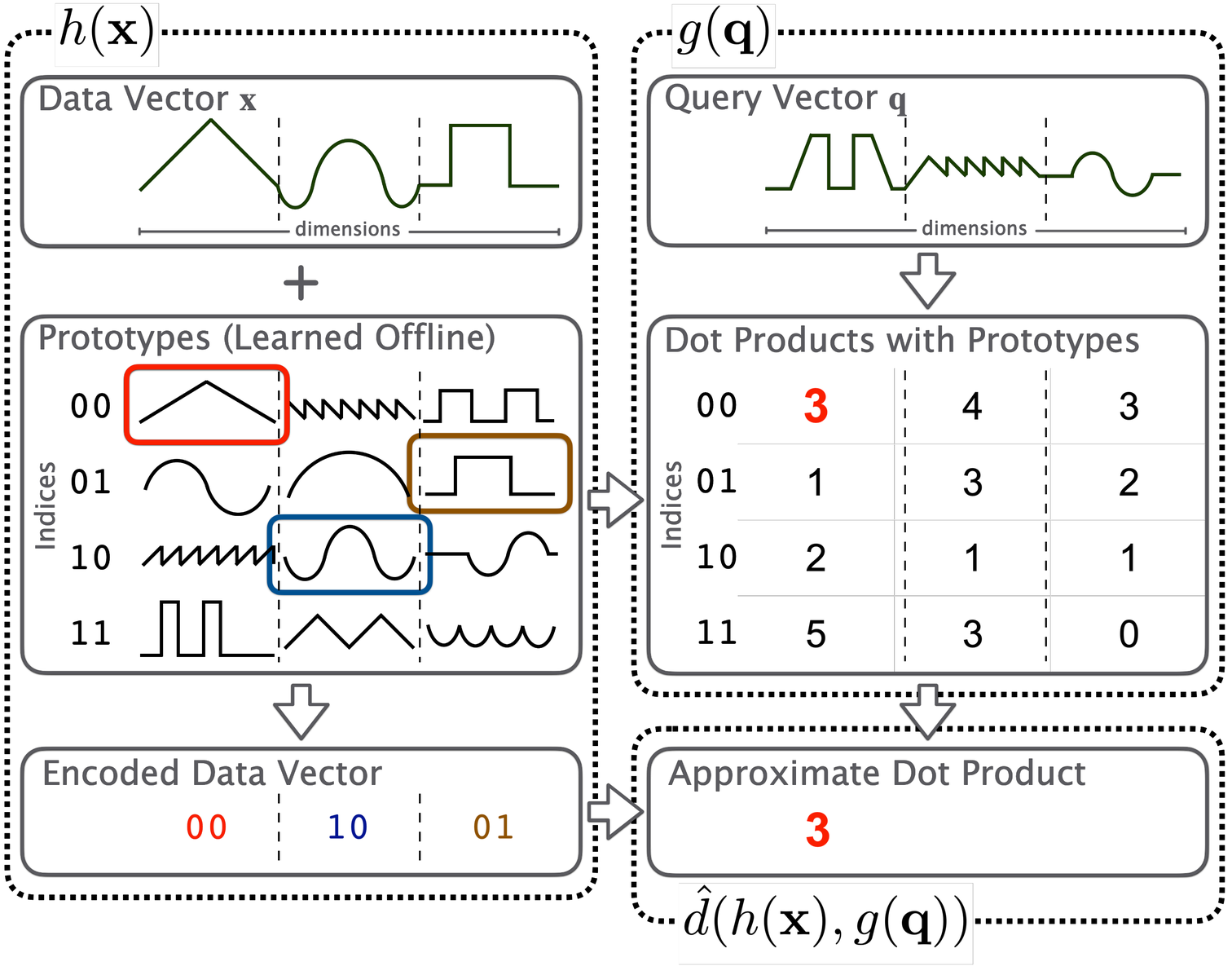
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



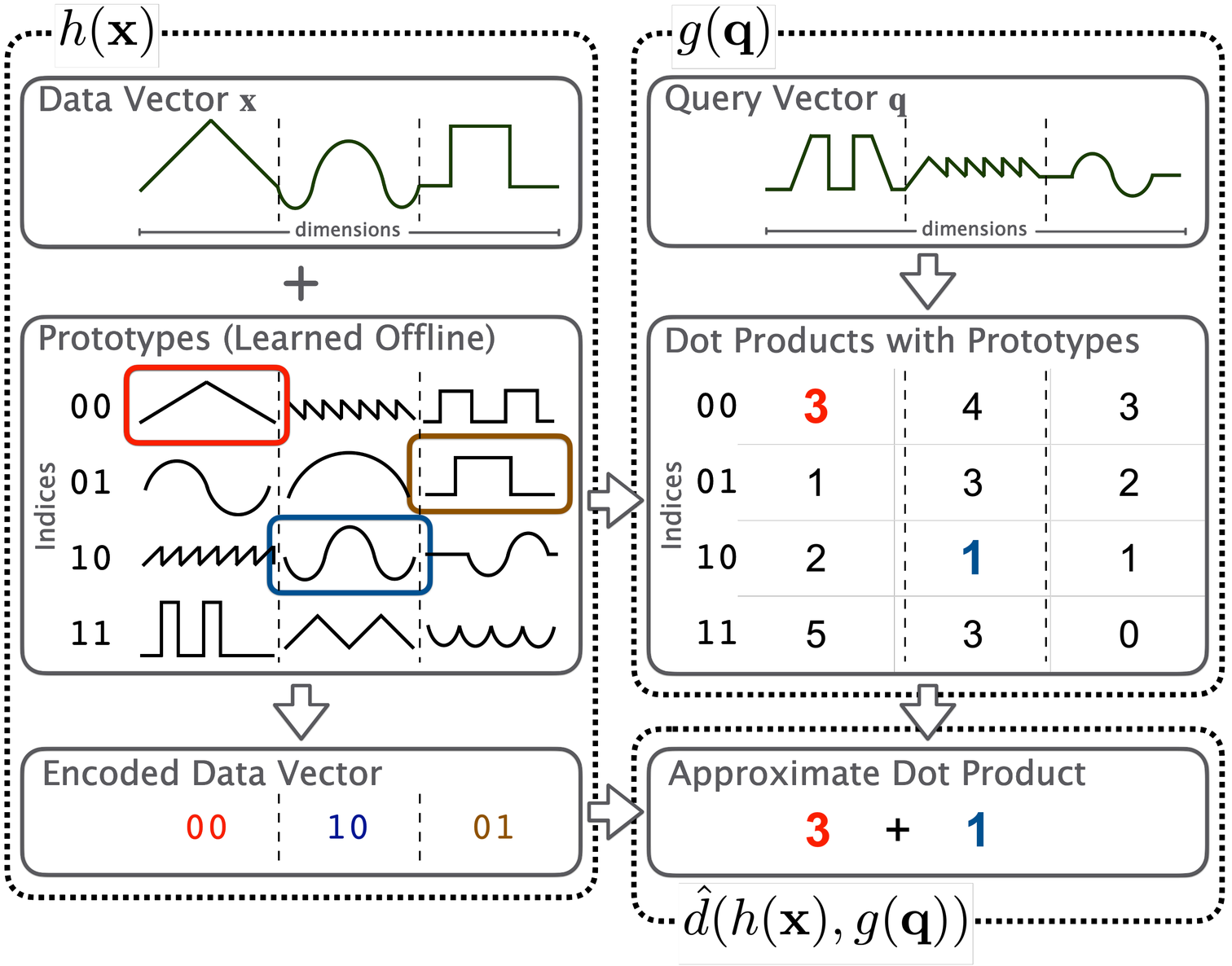
Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!



Background: Product Quantization

- **Online** product computation:
  - Encode each row of A as indices of nearest prototypes
  - Encode column of B as table of dot products
  - Table lookups!