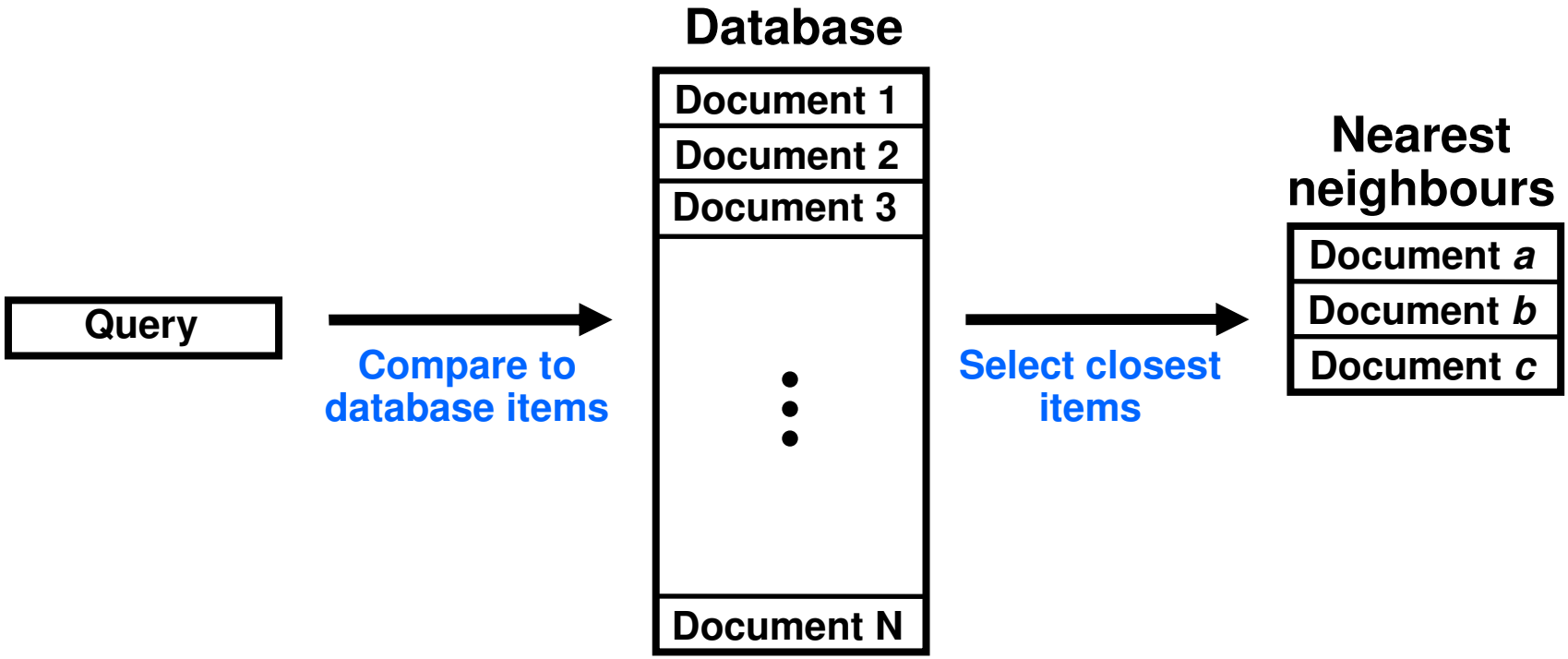
A Unified Approach to Learning Task-Specific Bit Vector Representations for Fast Nearest Neighbor Search

Vinod Nair
Yahoo! Labs, Bangalore

Joint work with Dhruv Mahajan and Sundararajan Sellamanickam

Nearest Neighbour Search

- Given a collection of data points and a new query point, find a subset of points in the collection that are closest to the query.



Many Tasks, One Subroutine

- **NN search is used as a subroutine in different tasks:**
  1. **Retrieval**
  2. **Nearest neighbour classification**
  3. **Nearest neighbour regression**

  …

Many Tasks, One Subroutine

- **NN search is used as a subroutine in different tasks:**
  1. **Retrieval**
  2. **Nearest neighbour classification**
  3. **Nearest neighbour regression**
  4. **…**
- **Different tasks require different senses of “nearest”.**

Many Tasks, One Subroutine

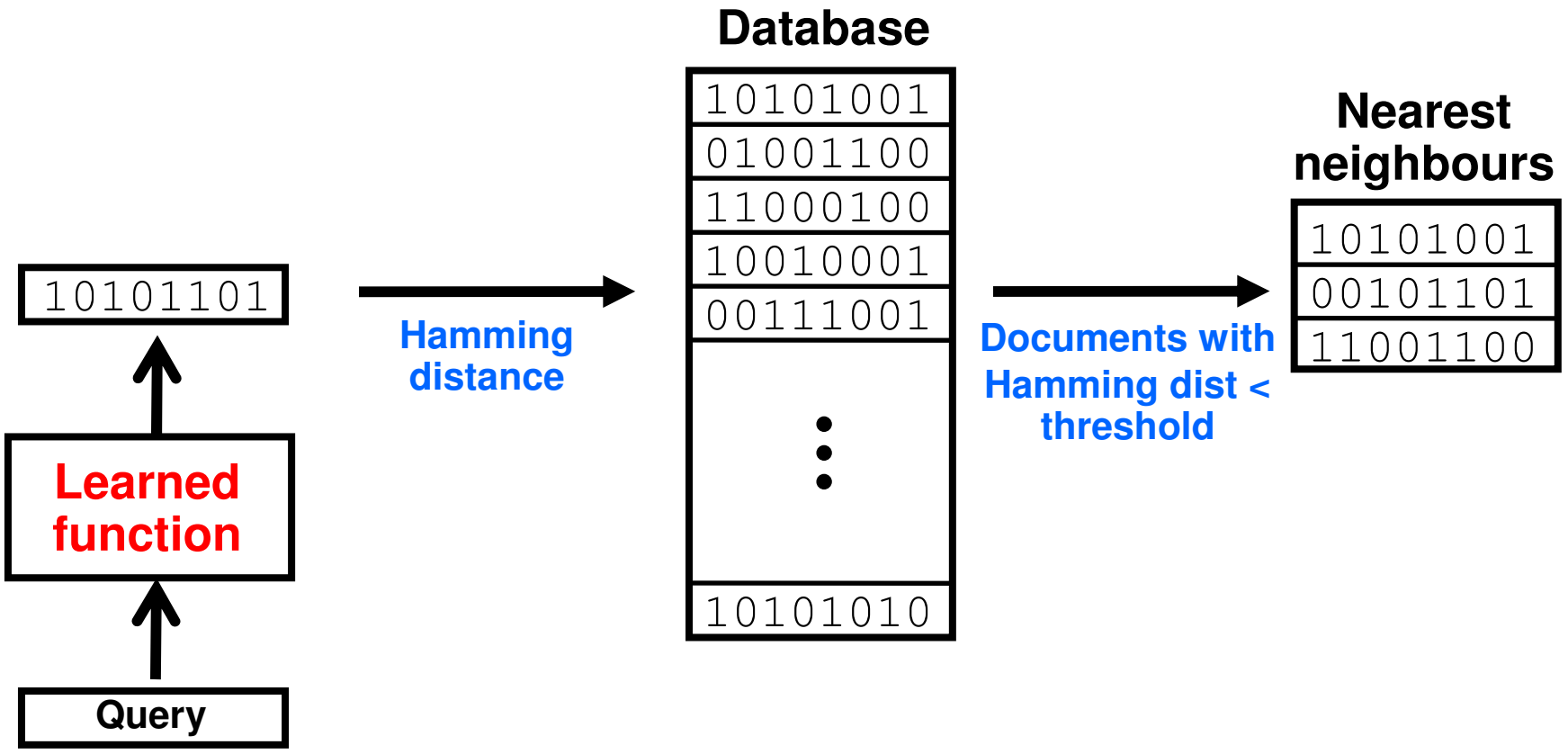
- NN search is used as a subroutine in different tasks:
    1. Retrieval
    2. Nearest neighbour classification
    3. Nearest neighbour regression
    …
- Different tasks require different senses of “nearest”.
- Examples:
    - Retrieval: Precision @ K, NDCG, MAP, etc.
    - NN classification: 0/1 loss.
    - Regression: Root mean squared error.

This Talk

- **Learn a representation of the input on which to do NN search with a fixed, predefined metric.**

This Talk

- Learn a bit vector representation and do NN search with Hamming distance as the metric.


Database
10101001
01001100
11000100
10010001
00111001
10101010
Nearest neighbours
10101001
00101101
11001100
10101101
Learned function
Query
Hamming distance
Documents with Hamming dist < threshold

The New Thing

A unified framework for learning bit vector representations for different tasks

APPROACH

Learning

Goal: **Learn a function that computes the bit vector representation of the input.**

Learning

Goal: Learn a function that computes the bit vector representation of the input.

- Posed as a learning-to-rank problem.

  Retrieval: Rank relevant documents higher.

  NN Classification: Rank documents with the same class label as the query higher than other documents.

  NN Regression: Rank documents by the absolute difference between their targets and the query's target.

Learning

Goal: Learn a function that computes the bit vector representation of the input.

- Posed as a learning-to-rank problem.

  Retrieval: Rank relevant documents higher.

  NN Classification: Rank documents with the same class label as the query higher than other documents.

  NN Regression: Rank documents by the absolute difference between their targets and the query's target.

- Use RankNet / LambdaRank (Burges et al.) to do the learning.

GENERALITY

SCALABLITY

ACCURACY

GENERALITY

Different types of nearest neighbour search problems can be cast as a ranking problem.

SCALABLITY

ACCURACY

GENERALITY

Different types of nearest neighbour search problems can be cast as a ranking problem.

SCALABLITY

Online gradient descent learning. We show results for datasets with 100K cases (2), 500K cases (2), 1.45M cases, 101 classes, 47K input dimensions.

ACCURACY

GENERALITY

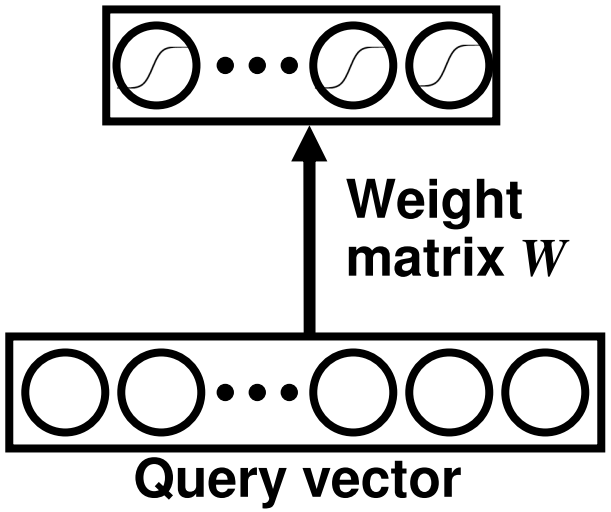
Different types of nearest neighbour search problems can be cast as a ranking problem.

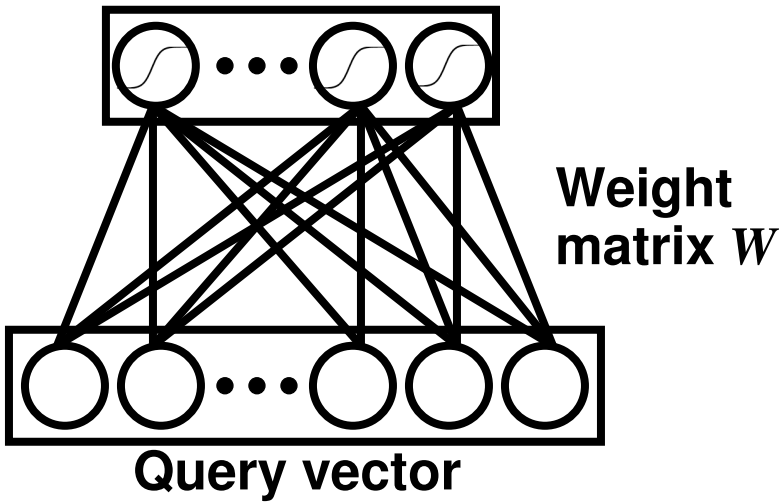
SCALABLITY

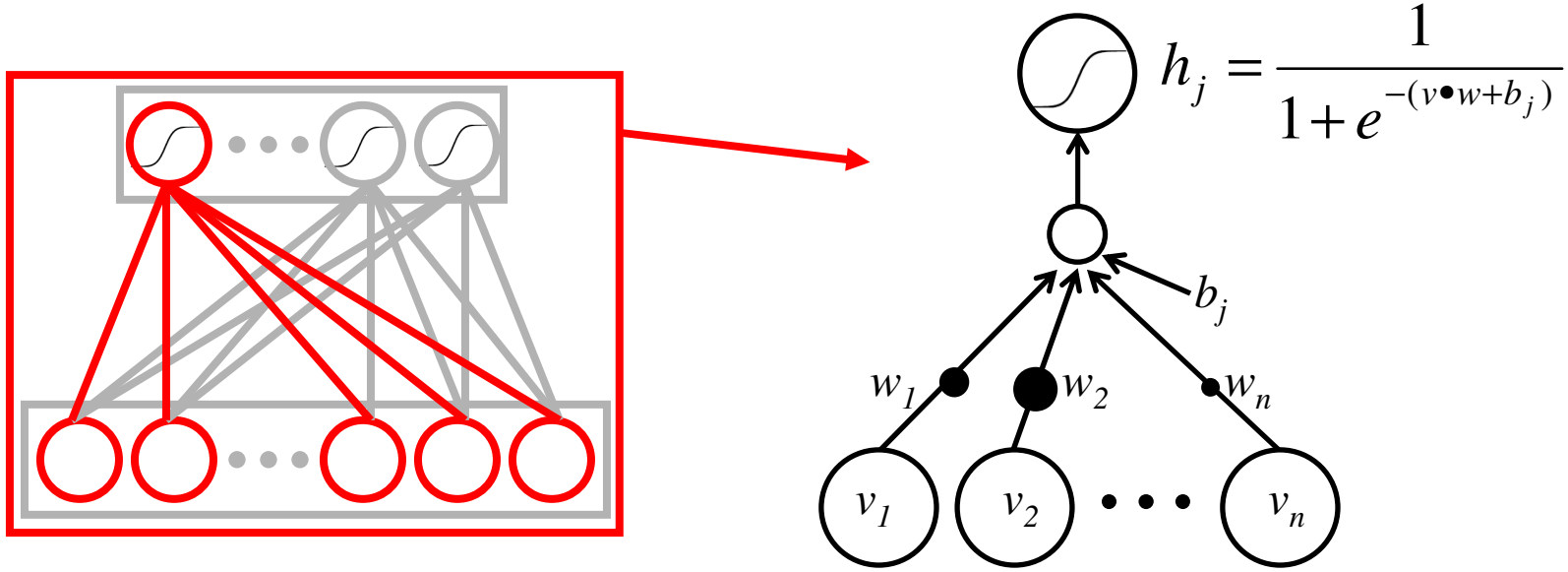
Online gradient descent learning. We show results for datasets with 100K cases (2), 500K cases (2), 1.45M cases, 101 classes, 47K input dimensions.

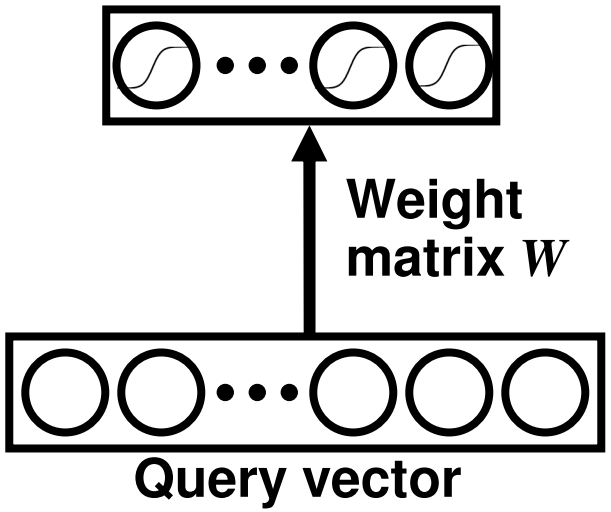
ACCURACY

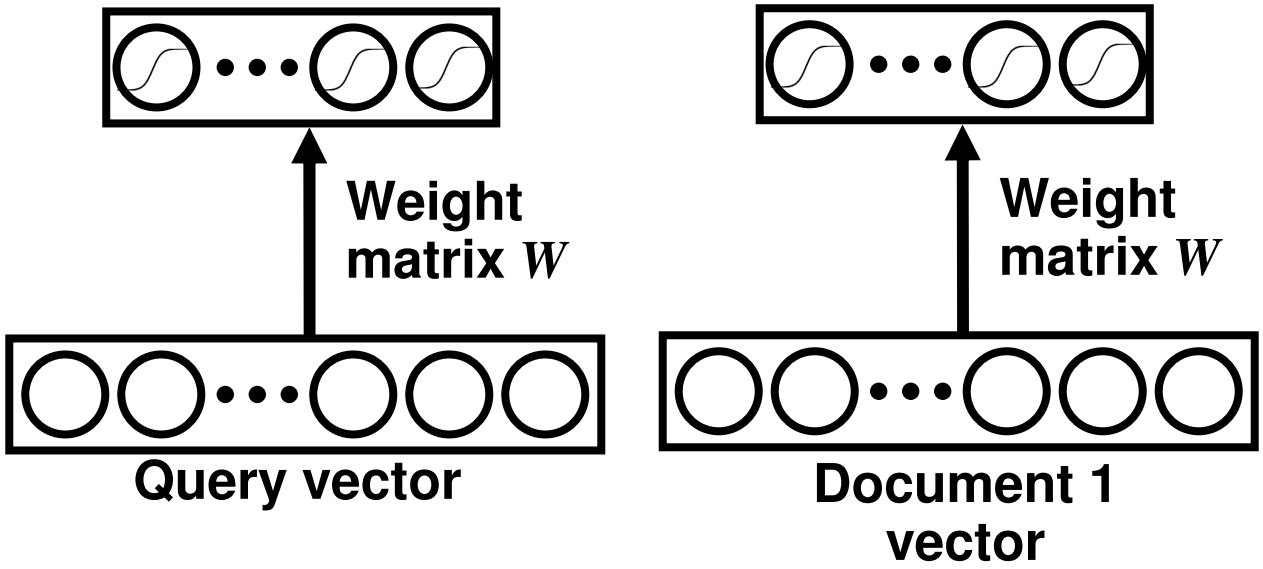
We beat several methods for NN classification and matrix projection-based retrieval.

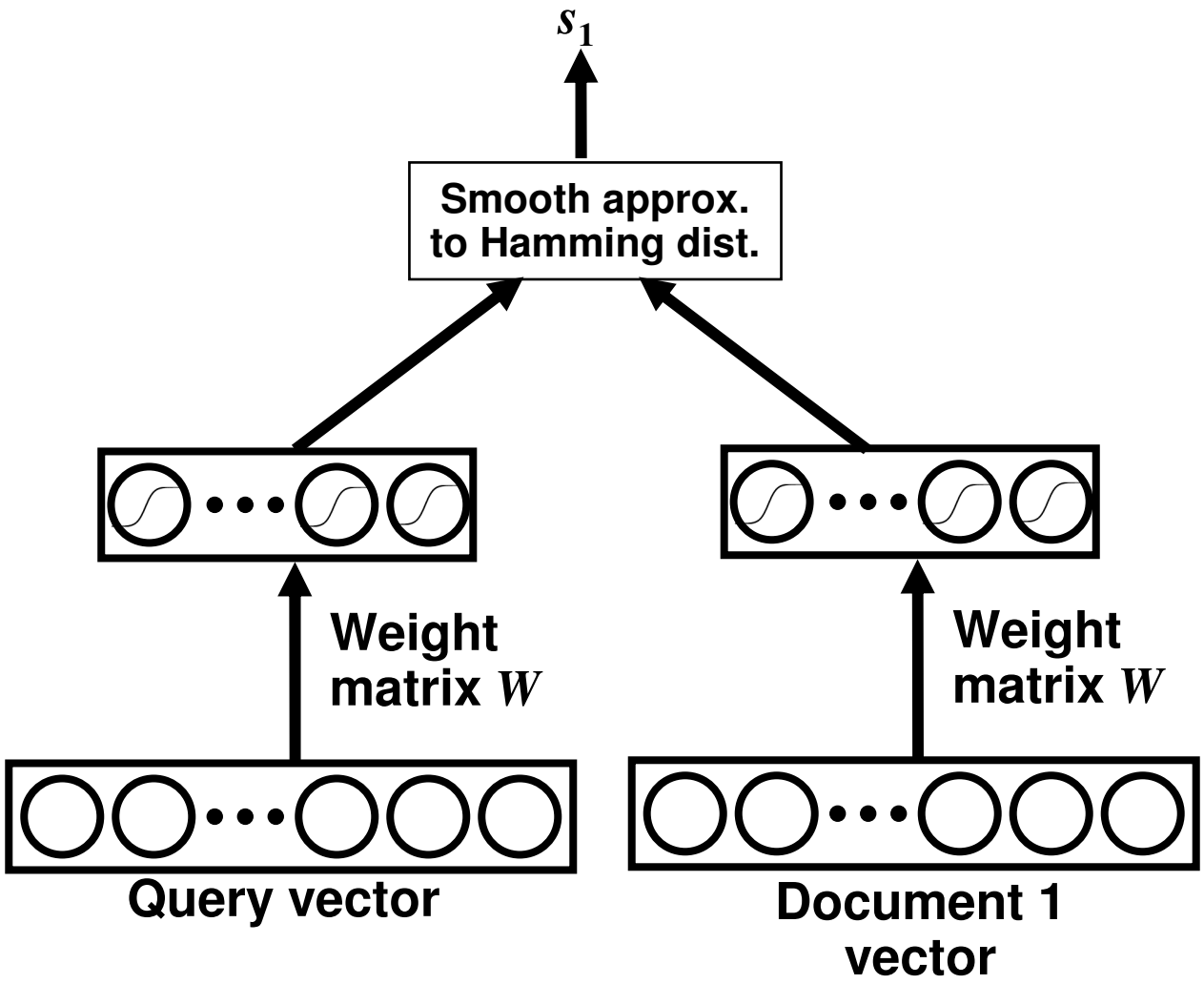

Weight
matrix W
Query vector


Weight
matrix W
Query vector


$h_j = \frac{1}{1+e^{-(v \bullet w + b_j)}}$
b_j
w_1
w_2
w_n
v_1
v_2
v_n


Weight
matrix W
Query vector

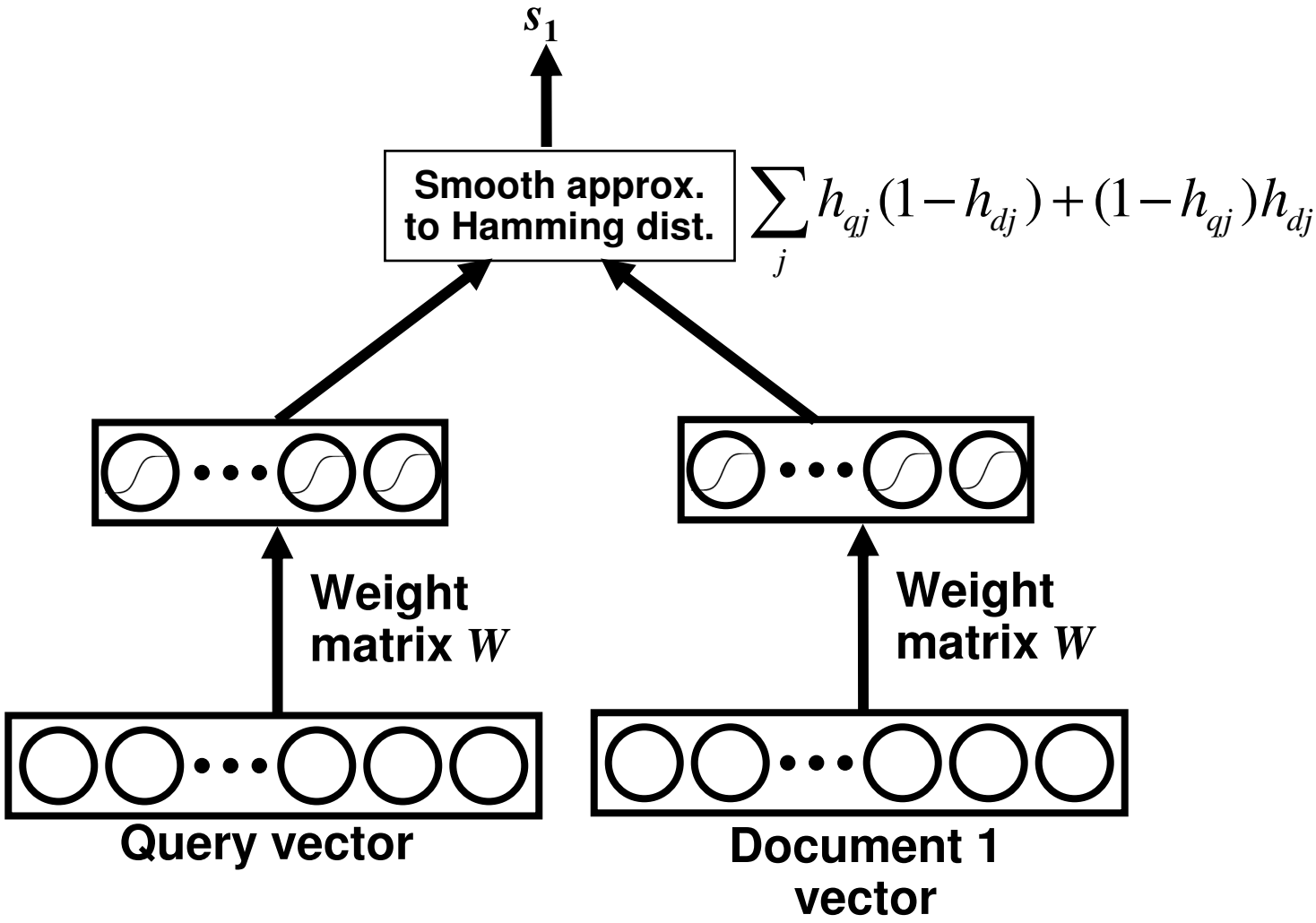

Weight
matrix W
Query vector
Weight
matrix W
Document 1
vector



s_1 = score for doc1

s_2 = score for doc2

$s_1 < s_2$ means doc1 is ranked higher than doc2



s_1 = score for doc1

s_2 = score for doc2

$s_1 < s_2$ means doc1 is ranked higher than doc2

RankNet Training

- **RankNet is trained to do pairwise ranking on documents pairs for a given query.**

$$P(d_1 \triangleright d_2) = \frac{1}{1 + e^{(s_1 - s_2)}}$$

RankNet Training

- **RankNet is trained to do pairwise ranking on documents pairs for a given query.**

$$P(d_1 \triangleright d_2) = \frac{1}{1 + e^{(s_1 - s_2)}}$$

- **Training objective is to maximize the log probability of the correct pairwise ranking.**

RankNet Training

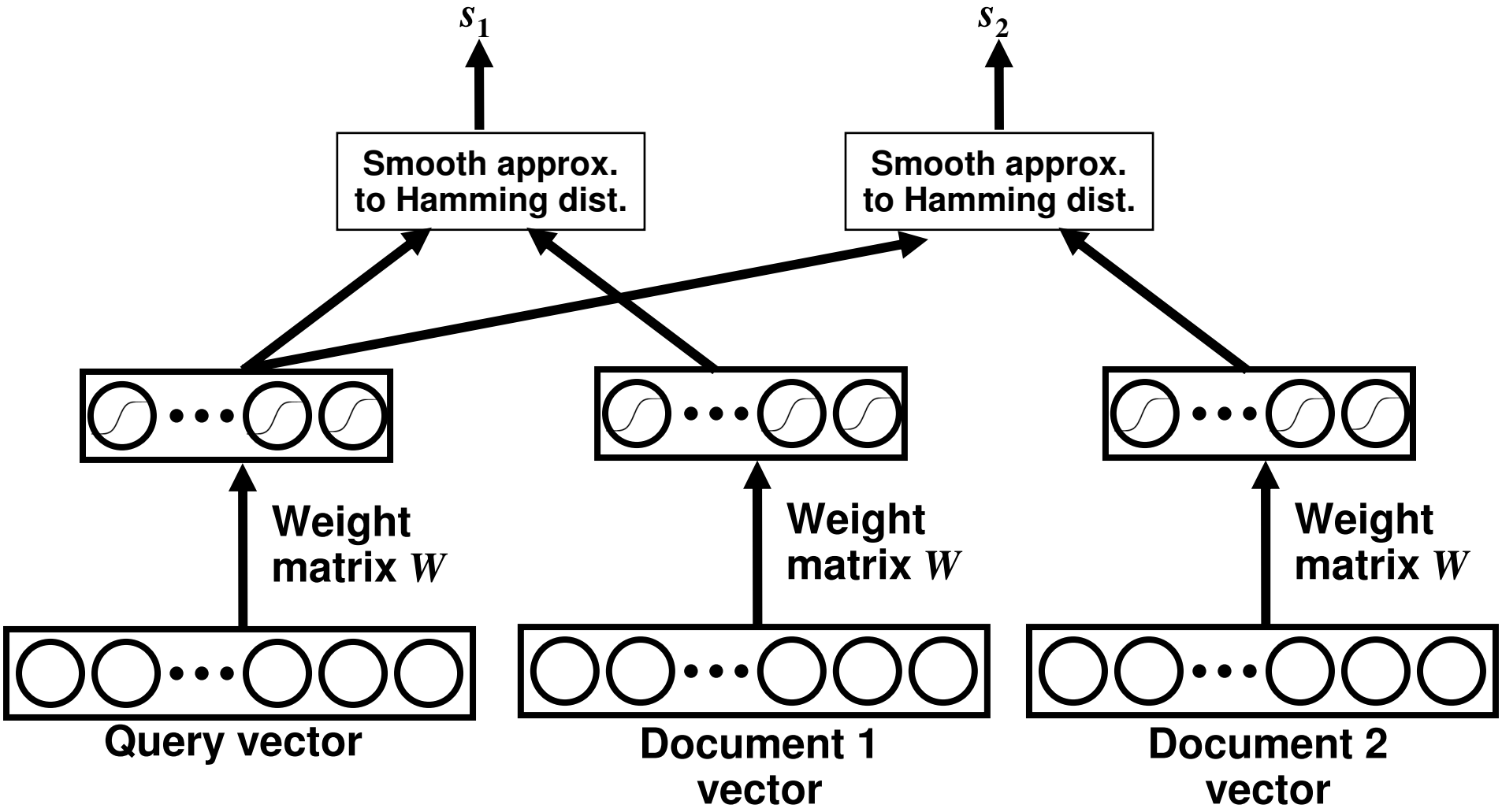
- **RankNet is trained to do pairwise ranking on documents pairs for a given query.**

$$P(d_1 \triangleright d_2) = \frac{1}{1 + e^{(s_1 - s_2)}}$$

- **Training objective is to maximize the log probability of the correct pairwise ranking.**

- **Optimization is done using stochastic gradient descent using triplets {query, doc1, doc2}.**

$$P(d_1 \triangleright d_2) = \frac{1}{1 + e^{(s_1 - s_2)}}$$



LambdaRank

- A way of optimizing non-smooth, non-differentiable objective functions.

  Examples: NN classification 0/1 loss, NDCG.

- Empirically it has been shown to converge to a local minimum of various non-smooth IR metrics.

- Rescales RankNet gradient in such a way that it is guaranteed to be the gradient of some (unspecified) objective function.

EVALUATION

Overview

- **Two types of tasks: 1) classification, and 2) retrieval.**

- **Methods: LSH, Spectral Hashing, RankNet, LambdaRank.**

- **Approximate ordering of methods by accuracy:**
  1. **LambdaRank**
  2. **RankNet**
  3. **Spectral Hashing**
  4. **LSH**

Classification Datasets

Dataset	Train set size	Test set size	Dimensionality	Classes
MNIST	60000	10000	784	10
MCAT	150344	4362	11429	7
Cover	522911	58101	54	7
RCV1	531742	15913	47236	101

MNIST

- **60K training cases, 10K test cases, 784 dim., 10 classes.**

Method	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
LSH	58.73%	52.15%	24.61%	13.20%	7.71%	4.64%
SpecHash	33.63%	19.67%	10.10%	6.82%	5.74%	4.63%
RankNet	12.64%	8.50%	5.54%	4.20%	4.01%	3.55%
LambdaRank	12.32%	7.57%	4.85%	4.02%	2.37%	1.63%

L2 kNN = 3.09%, NCA = 2.45%, LMNN = 1.72%.

MCAT

- **150344 training cases, 4362 test cases, 11429 dim., 7 classes.**

Method	8 bits	16 bits	32 bits
LSH	67.35%	60.39%	42.85%
SpecHash	24.97%	8.02%	5.36%
RankNet	1.70%	1.38%	1.42%
LambdaRank	1.54%	1.47%	1.31%

L2 kNN = 3.67%, NCA= 7.66%.

Cover

- **522911 training cases, 58101 test cases, 54 dim., 7 classes.**

Method	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
LSH	42.07%	33.96%	20.65%	12.33%	9.55%	7.64%
SpecHash	42.17%	34.30%	27.29%	14.68%	9.46%	7.44%
RankNet	29.00%	26.54%	21.62%	18.50%	15.40%	9.58%
LambdaRank	30.36%	21.60%	14.24%	10.88%	7.74%	5.52%

L2 kNN = 6.25%, NCA = 4.01%.

RCV1

- **531742 training cases, 15913 test cases, 47236 dim., 101 classes.**

Method	8 bits	16 bits	32 bits	64 bits
LSH	84.63%	75.20%	63.79%	50.95%
SpecHash	75.24%	59.46%	-	-
RankNet	45.60%	18.56%	16.84%	13.34%
LambdaRank	25.21%	15.93%	14.41%	12.58%

L2 kNN = 29.67%, NCA = 45.79%.

Retrieval Datasets

Dataset	Train set size	Test set size	Dimensionality
INRIA SIFT	100000	1000000	128
Tiny Images	1450000	100000	512

INRIA SIFT

- **100K training cases, 1M test cases, 128 dim.**

Precision x 10^{-4} @ Hamming distance < 2 from the query.

Method	8 bits	16 bits	32 bits
LSH	6.11	44.70	476.29
SpecHash	14.26	200.04	1462.76
RankNet	15.00	184.61	1687.21
LambdaRank	17.16	266.63	1805.25

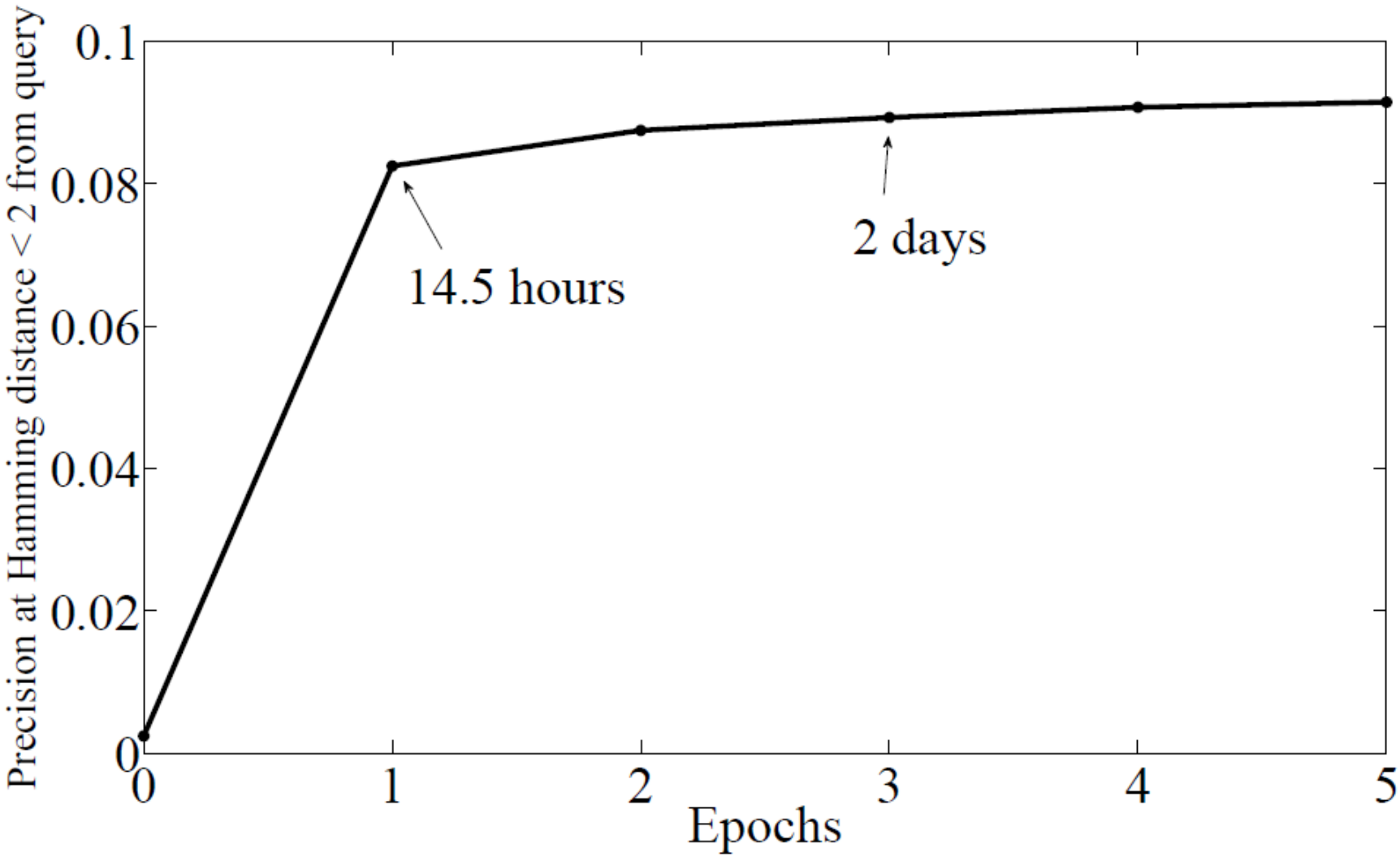
Tiny Images

- **1.45M training cases, 100K test cases, 512 dim.**

Precision x 10^{-5} @ Hamming distance < 2 from the query.

Method	8 bits	16 bits	32 bits
LSH	6.93	34.58	410.41
SpecHash	28.73	211.39	3396.62
RankNet	36.85	430.98	7979.02
LambdaRank	42.93	578.99	9065.89

Training Time for Tiny Images



Conclusions

- **A unified way of learning bit vector representations for different tasks.**
- **Preliminary results for NN regression.**
- **Possible application to speeding up neighbourhood based methods for Collaborative Filtering.**

Dataset	Lambda Rank	L_2 kNN	NCA
MNIST	23.99	756.81	121.19
MCAT	8.85	385.13	707.84
Covertype	765.83	1744.83	1785.33
RCV1 Subset	127.78	5457.93	10706.11

Table 6: CPU time (seconds) required to find the nearest neighbors for the entire test set by different methods on the various datasets.