# Neural Networks and Deep Learning
## Dual Encoder,
## Approximate Nearest Neighbor

Shumin Wu
Department of Computer Science
shumin.wu@colorado.edu
February 10, 2020

# CNN Review

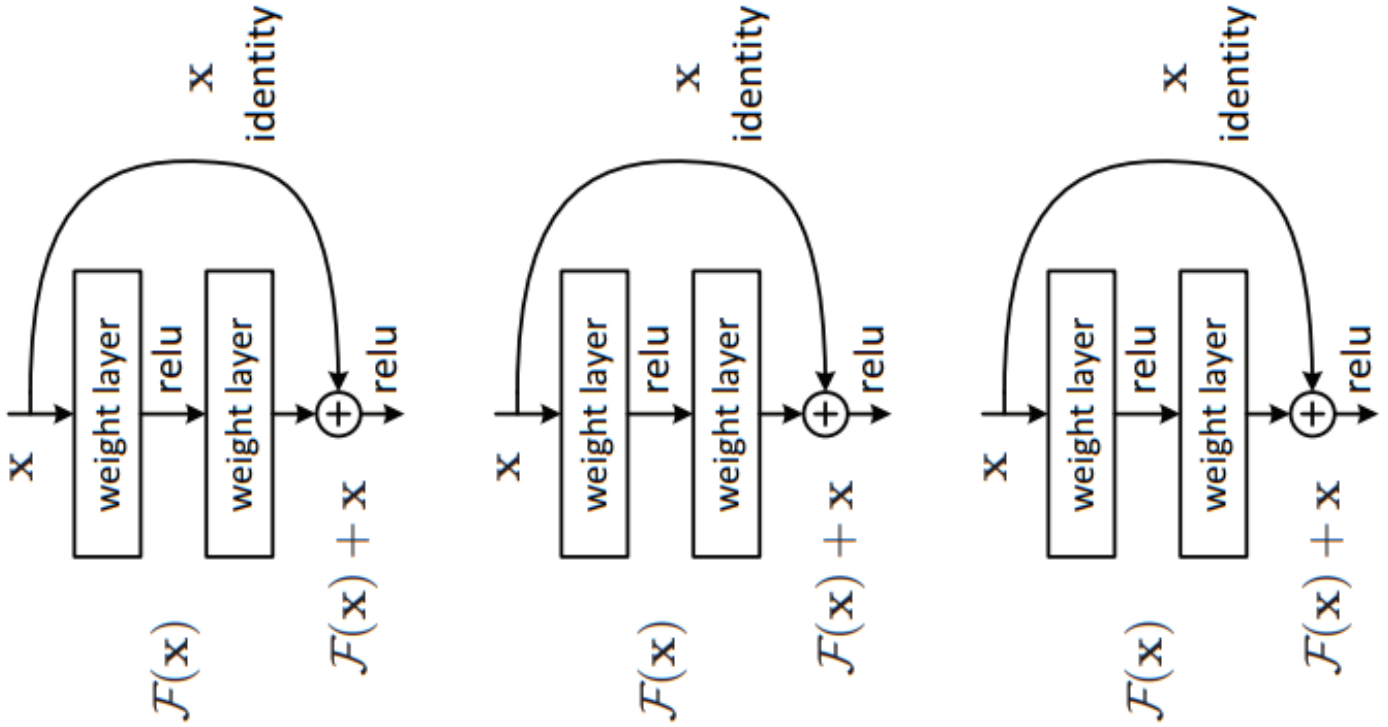
- Discrete convolution: Let $I(i,j)$ be an image, and let $K(m,n)$ be a convolution filter.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- Does not provide *translation invariance*, only *equivariance*
  - Applying a translation to the source prior to convolution is equivalent to applying the same translation to the output.
  - Pooling can be used for translation invariance (at the cost of loosing spatial information within the pooling region).

# CNN Review

- Residual learning can be effective for deep network layering (ResNet)




Chaining residual building blocks

# CNN Review

- Backprop is also convolution

  Input: $\nabla_{a_{j+1}} L$

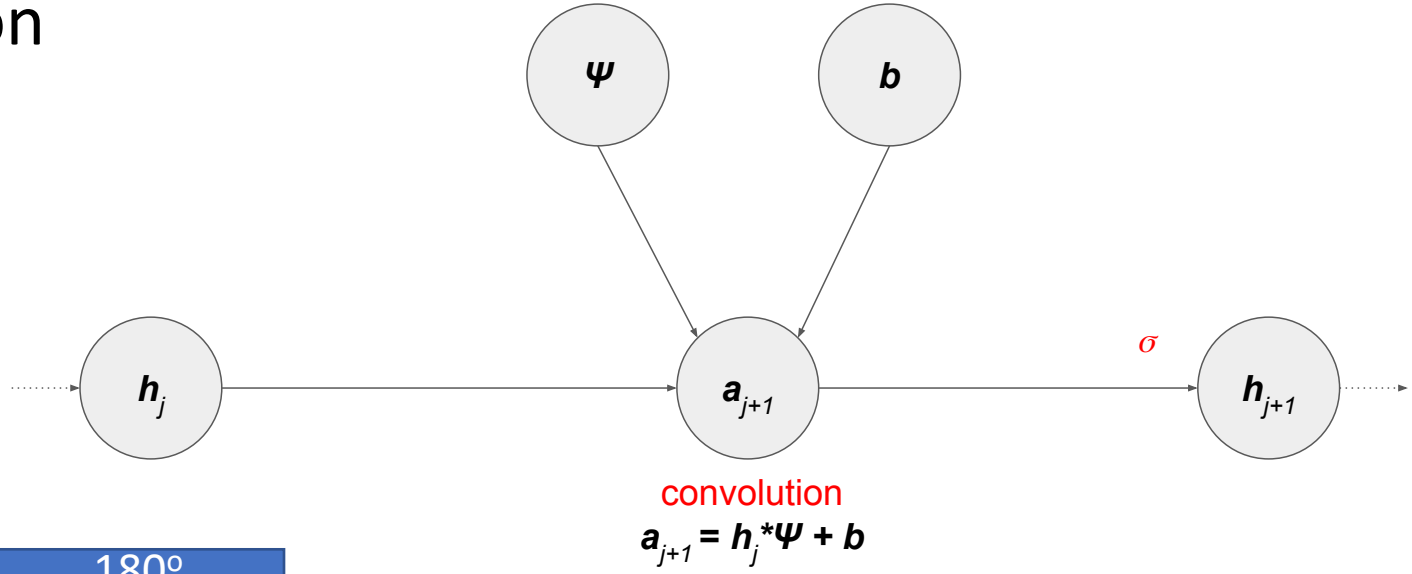
  $$\nabla_{\boldsymbol{\psi}} L = \mathbf{h}_j * \left(\nabla_{\mathbf{a}_{j+1}} L\right)$$

  $$\nabla_{\mathbf{h}_j} L = \left(\nabla_{\mathbf{a}_{j+1}} L\right) * \boldsymbol{\psi}'$$

  where:

  $$\psi'_{i,j} = \psi_{m+1-j,n+1-i}$$

180° rotation of the the filter




- Some padding may be needed to match dimension
  - Also called transposed convolution (more details on GAN)

# Neural Network Classification

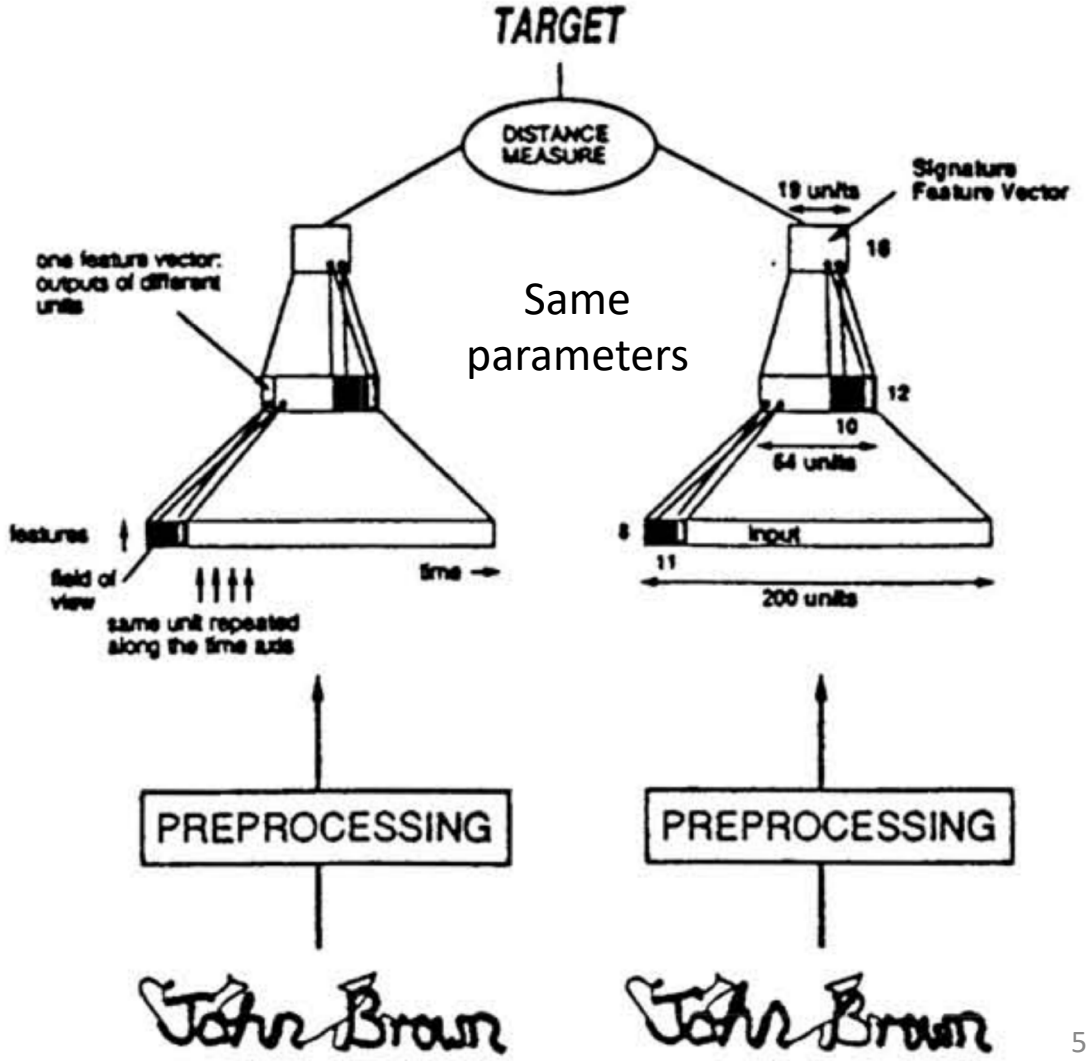
- So far: output layer contains 1 value per class w/ softmax activation
- How to add a new class to the classifier?
  - Fine tune the model with another class
- What if there are very few (maybe only 1) labeled example for the new class?
- Last layer (before output) of a neural network may be considered an “encoding”/“embedding” of the input (image, text, etc).
- What if we define the loss as the distance of encoding between inputs of the same class vs inputs of different classes?

# Siamese Network/Dual Encoder

- Signature verification

  Signature Verification using a Siamese Time Delay Neural Network, Bromlet et al, 1994

- Widely used for facial recognition
  - DeepFace
  - FaceNet
- Also in NLP
  - Does not need to have identical architecture/share all parameters (ex: question/answer pairs)
  - Universal Sentence Encoder and its multilingual extension



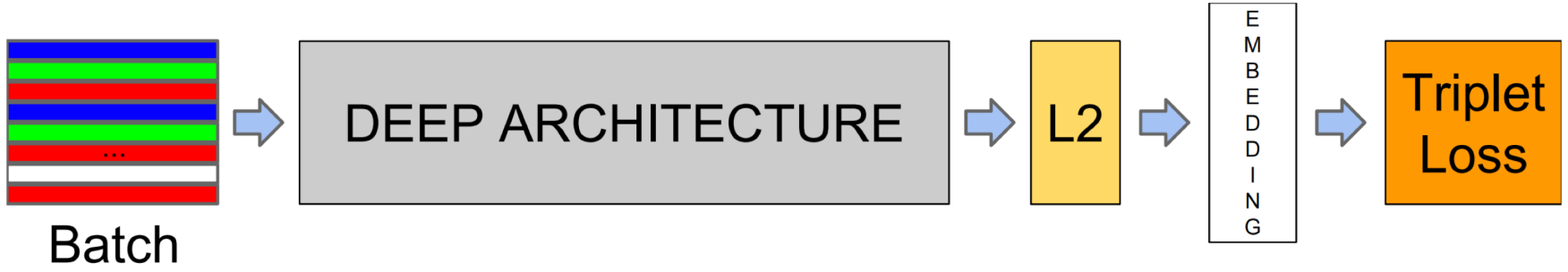
# Facial Recognition

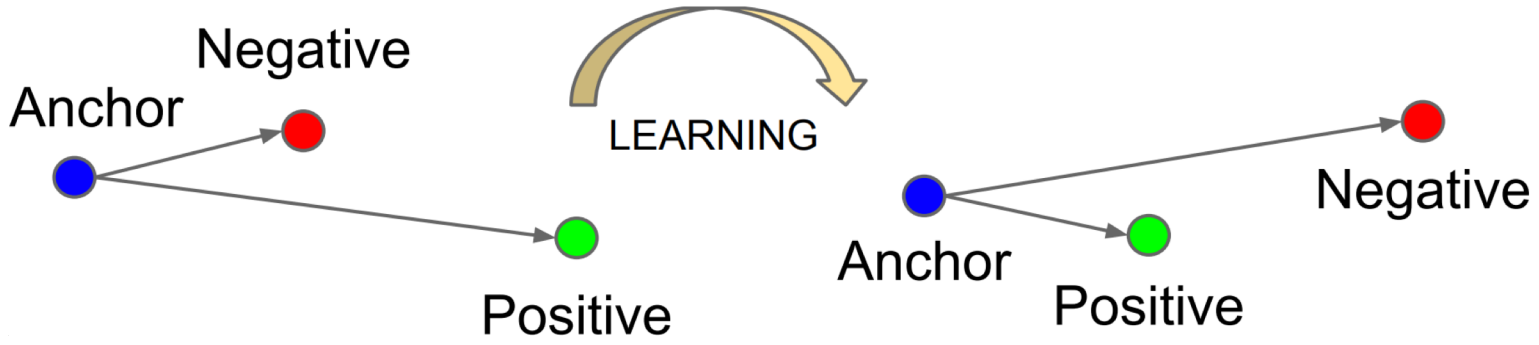
- Find the nearest face in database and accept/reject based on distance value
  - absolute or relative to the next closest face
- Can “recognize” a new face by adding a single reference photo w/o retraining the model
  - Also called one-shot learning[1], originally w/ a generative model.

[1]Li Fei-fei and Rob Fergus and Pietro Perona, “One-Shot Learning of Object Categories”, IEEE 2006

# FaceNet

- Training w/ triplet loss:



# FaceNet Triplet Loss

Let: $x_i^a = \text{anchor},\ x_i^p = \text{positive example},\ x_i^n = \text{negative example}$

$f(x) = \text{encoding function}$

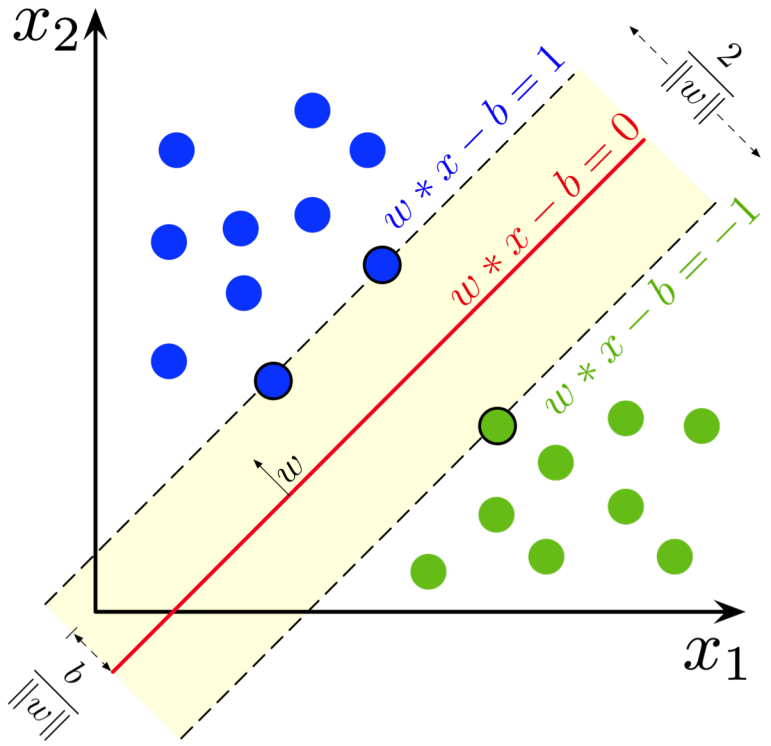
Objective: $||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau$$

Loss : $$\sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]$$

# FaceNet Triplet loss

- Triplet loss $\alpha$ is a margin similar to Support Vector Machine (but is fixed instead of maximized)




Source: [Wikipedia]

# Choosing Training Triples

- Random sample
  - May rarely encounter “difficult” negative examples
- Find the furthest positive example and the closest negative example
  - Difficult over the entire training data
    - Data need to be re-encoded/re-indexed with each model update
  - Oversample possibly erroneously labeled or poor quality examples
- FaceNet
  - Select examples from a large batch (thousands of examples)
  - Ensure enough examples per class (40), each acting as the anchor
  - Select semi-hard negative examples
    - More distant than positive examples, but not separable by $\alpha$

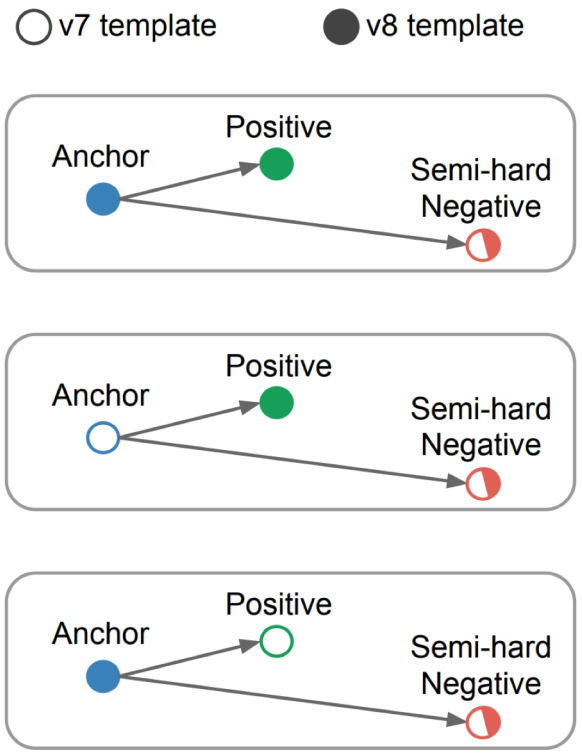
# FaceNet False Accept

# FaceNet False Reject

Only 5 pairs are correctly labeled in the test data, can you tell which ones?

# Encoding Stability

- Every trained model may yield an incompatible encoding function
  - Production headaches
- FaceNet Harmonic Triplet Loss
  - Mixes examples from 2 models
  - Retrain embedding layer, then whole model
  - New model corrects some examples
  - Allows old index serving new encoding



# Finding the Nearest Neighbor(s)

- Brute force: compute pairwise distance between query and every element in the database
  - $O(nd)$ for database size of $n$ and encoding dimension $d$
- Approximate Nearest Neighbor
  - Tree-based
  - Hashing-based
  - Graph-based
  - And others…
- Encoding from Neural Network
  - Dense encoding weights
  - Distance function is a metric

# Distance Metric

Satisfies:

$$d(x, y) = 0 \Leftrightarrow x = y$$
$$d(x, y) = d(y, x)$$
$$d(x, z) \leq d(x, y) + d(y, z)$$

Examples of non-metric distance:

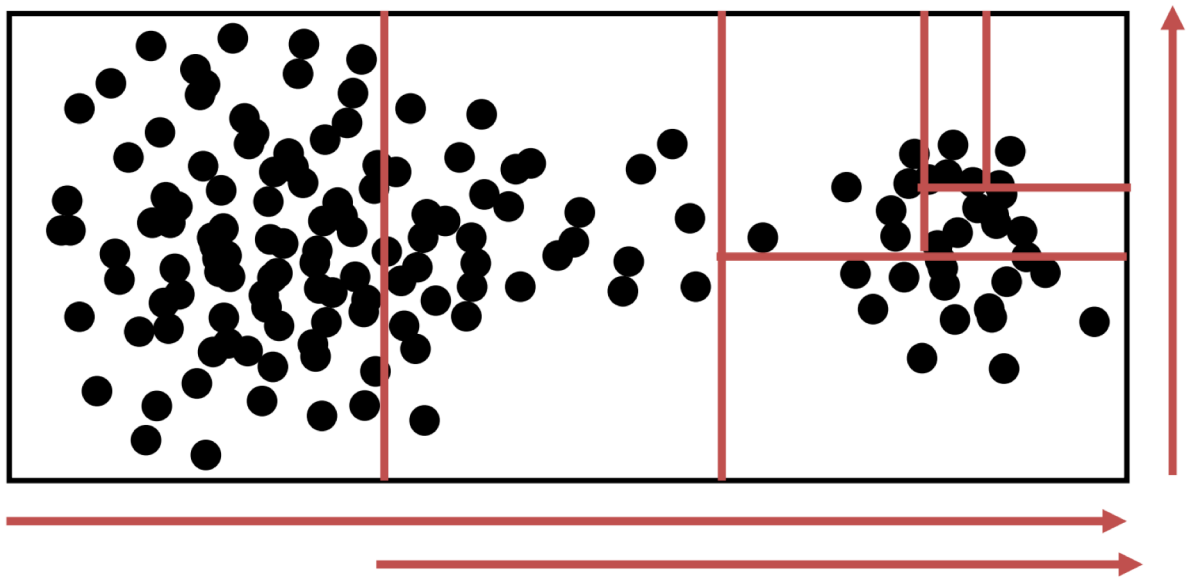
- Kullback-Leibler divergence:

$$KL(p, q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

- Jaro-Winkler distance
  - Character edits for typo-correction

# k-d Tree Building

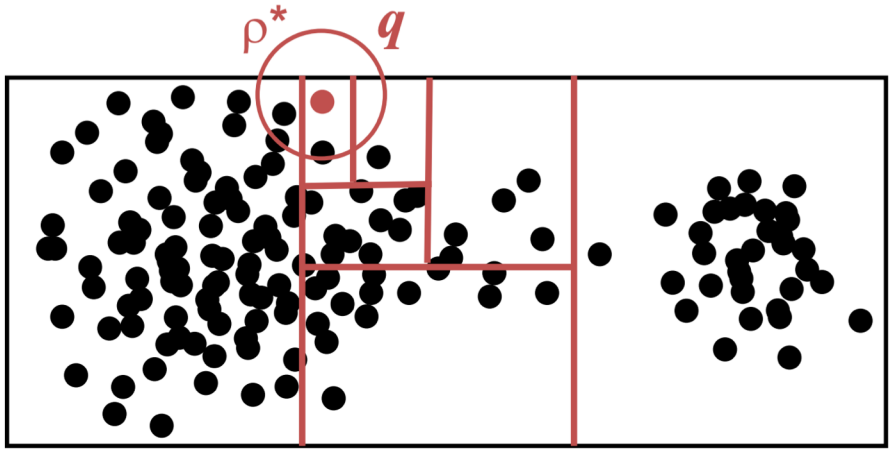
- Split along hyperplane parallel to axis
- Start w/ dimension of largest variance
- Split along median point until space contains 1 point (leaf)

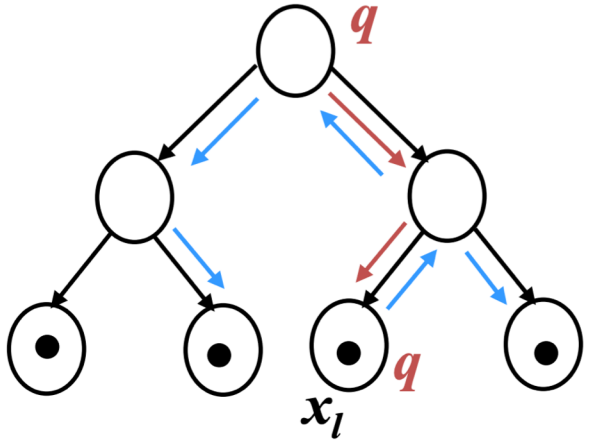


Source: Large-Scale Machine Learning, EECS 6898

# k-d Tree Searching

- Given a query $q$, find leaf partition and compute distance
- Backtrack to neighboring partitions w/ possibly closer distance
- Successively update smallest distance to reduce search radius



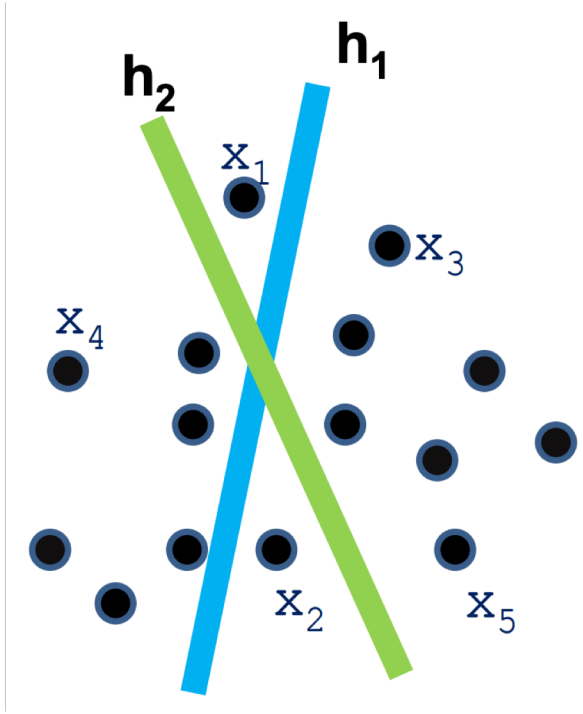




Source: Large-Scale Machine Learning, EECS 6898

- Works well in low dimensions (10-15)

# ANN Hash

Linear projection based partitioning




| X | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $y_1$ | 0 | 1 | 1 | 0 | 1 |
| $y_2$ | 1 | 0 | 1 | 0 | 1 |
| … | … | … | … | … | … |
| $y_m$ | … | … | … | … | … |
| | 010… | 100… | 111… | 001… | 110… |

Source: Large-Scale Machine Learning, EECS 6898

# ANN Hash Training

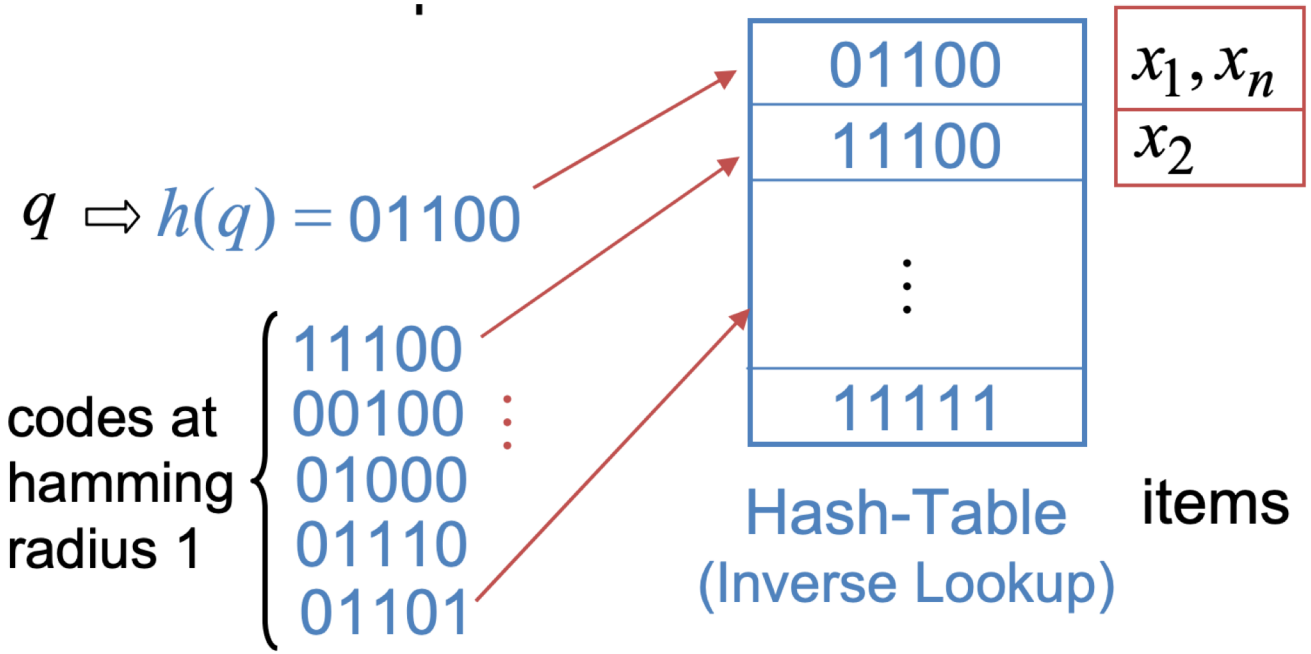
- Need a model to convert input to code sequence
- Learn weights $w_{1..m}$ and $b_{1..m}$ for $m$ hashing functions

$$h_k(x) = sgn(w_k^T x_k + b_k) \quad h_k(x_k) \in -1, 1$$

Typically each $w_k$ and $b_k$ are masked for a few dimensions of the space.

# ANN Hash Lookup

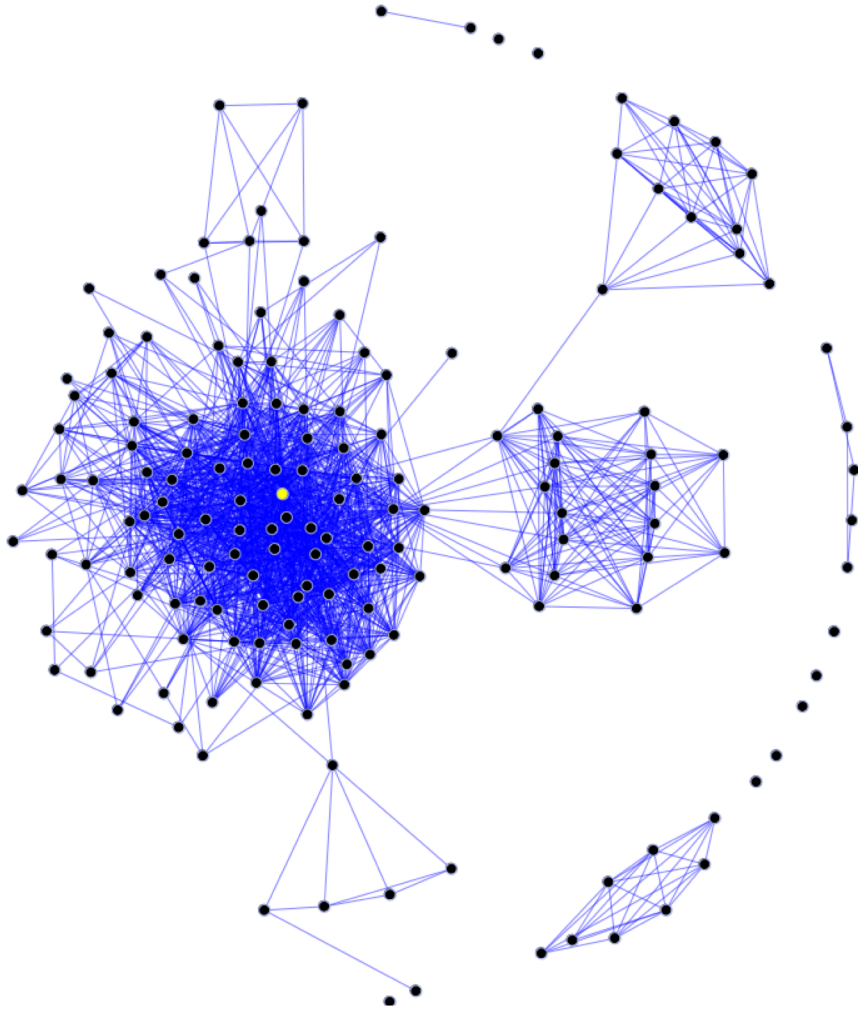
- Build an inverted index of hash code to input values
- Look up based on hash code of query
  - Exact or a small hamming radius away
- Spare in high dimension
  - Use multiple hash functions/tables to increase recall
- Rerank candidates in original encoding space
  - Expensive w/ imbalanced hash bucket




Source: Large-Scale Machine Learning, EECS 6898

# Small World Graph

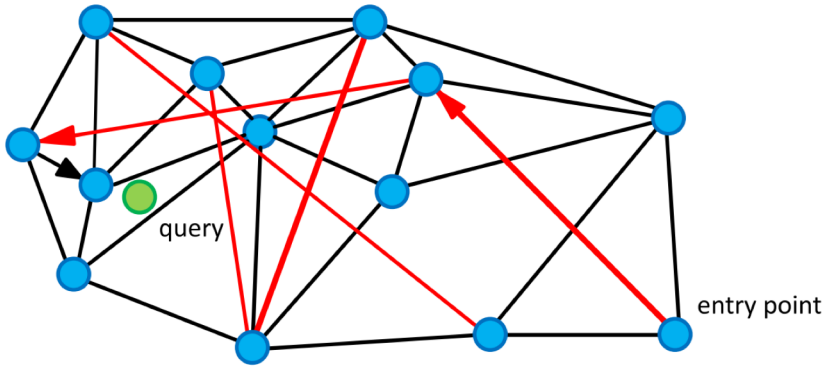
- Sparse edges w/ low degrees of separation between vertices
  - Example: social network, web link, etc
  - Distance does not need to be metric



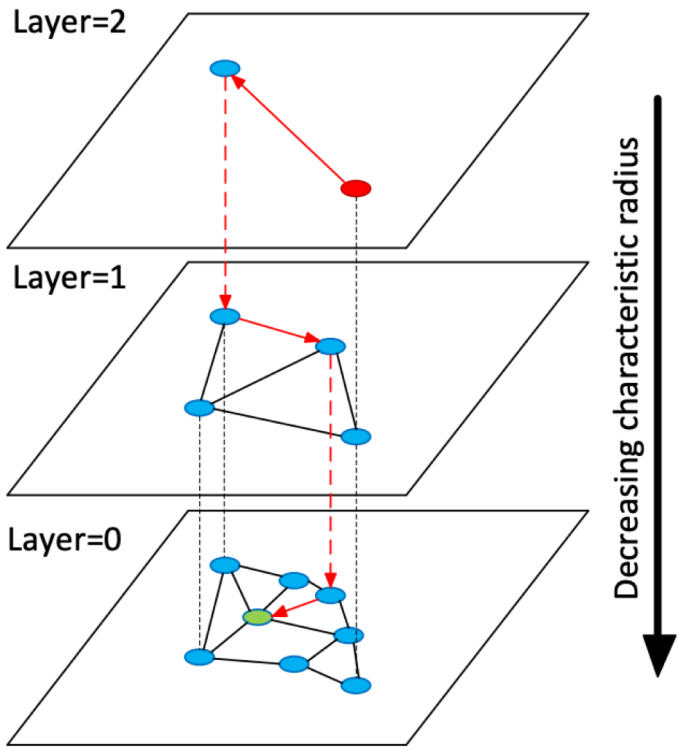
Source: wikipedia

# Navigable Small World ANN Search

- Start w/ an entry point
- Find neighbor w/ closest distance
- Iterate until distance stops improving



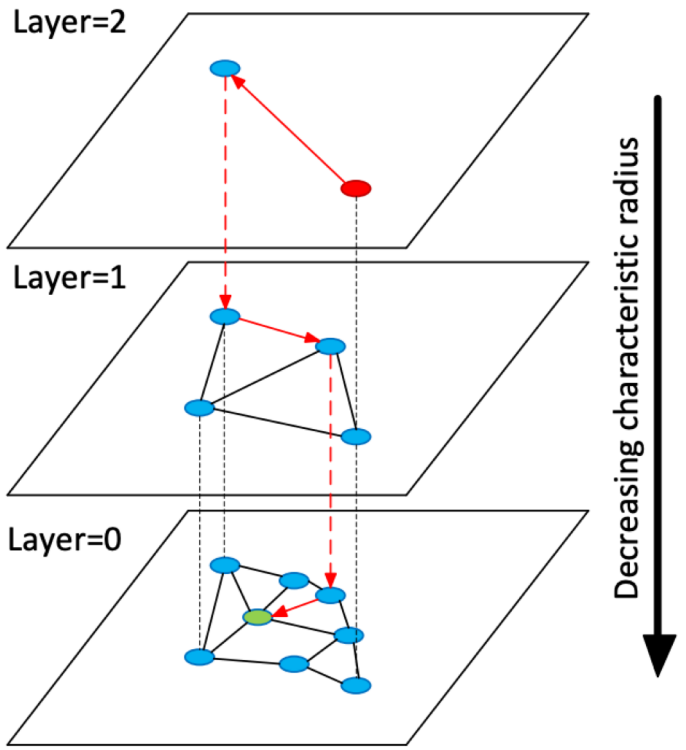

Source: [Approximate nearest neighbor algorithm based on navigable small world graphs]




Source: [Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs] (HNSW)

# HNSW Graph Building

- Perform search and keep track of neighbors at each layer
- Randomly pick highest layer to insert
  - Exponential decay
  - Always insert in all lower layers
- For each layer, find K nearest neighbors and insert links
  - Search locality at each layer
  - May need to delete existing edges if max degree of vertex is too large
- Memory proportional to degree $d$ of vertex (typical 6-48) at layer 0
  - For $n$ inputs: $d \times 2 \times log_2 n \times n$



# ANN Benchmarks

http://ann-benchmarks.com/