BoostMap:

A Method for Efficient Approximate Similarity Rankings

V. Asthitsos, J. Alon, S. Sclaroff, G. Kollios

Presentation by B Brent Gordon
for CMSC 828S, Spring 2005

Outline

- Main Ideas and Definitions
- The Algorithm
- Examples & Remarks

Outline

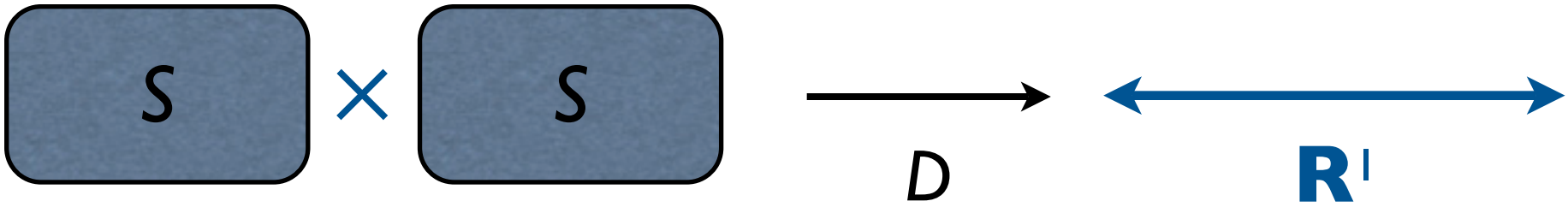
Main Ideas and Definitions

Data set *S*, with a distance function *D*.
Proximity structure function
Weak representations of proximity structure
Strong representation, from AdaBoost
Also builds on FastMap

The Algorithm

Examples & Remarks

Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$



Main Points:

- D may be expensive to compute.
- D may not satisfy triangle inequality.

Examples you might bear in mind:

- Image and Multimedia data
  - Paper experiments on hand signs & "Chamfer Distance" of Edges
  - Maybe something similar with Fingerprints
  - Paper experiments on video of ASL signing & a complicated distance based on optical flow
- Complex numerical or statistical data
  - Protein or DNA sequences, & "edit distance" or something
  - Document analysis, & some function of word counts

- Data Set S, Distance function $D : S \times S \to \mathbf{R}^1$

Idea 1. Characterize (S,D) by its *proximity structure*.

Definition (in paper). The *proximity order* on (S,D) is the 3-argument binary-valued function

$$P_r(x,y) = P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$$

In words, ± 1 as x or y is closer to r.

- Ranks distances pairwise
- No other magnitude information
- Do as you please with 0 or equality

Definition (mine). The *proximity structure* on (S,D) is the relation induced on $S^3 \times \{\pm 1\}$ by the proximity order.

Is proximity structure alone enough to answer any NN-query(?)

Remark. The concept *might* be useful with high-dimensional data that clusters into similarity classes(?)

BoostMap: Main Ideas: Proximity Structure

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$

Example.

Some proximity function values for "Cities of North America"

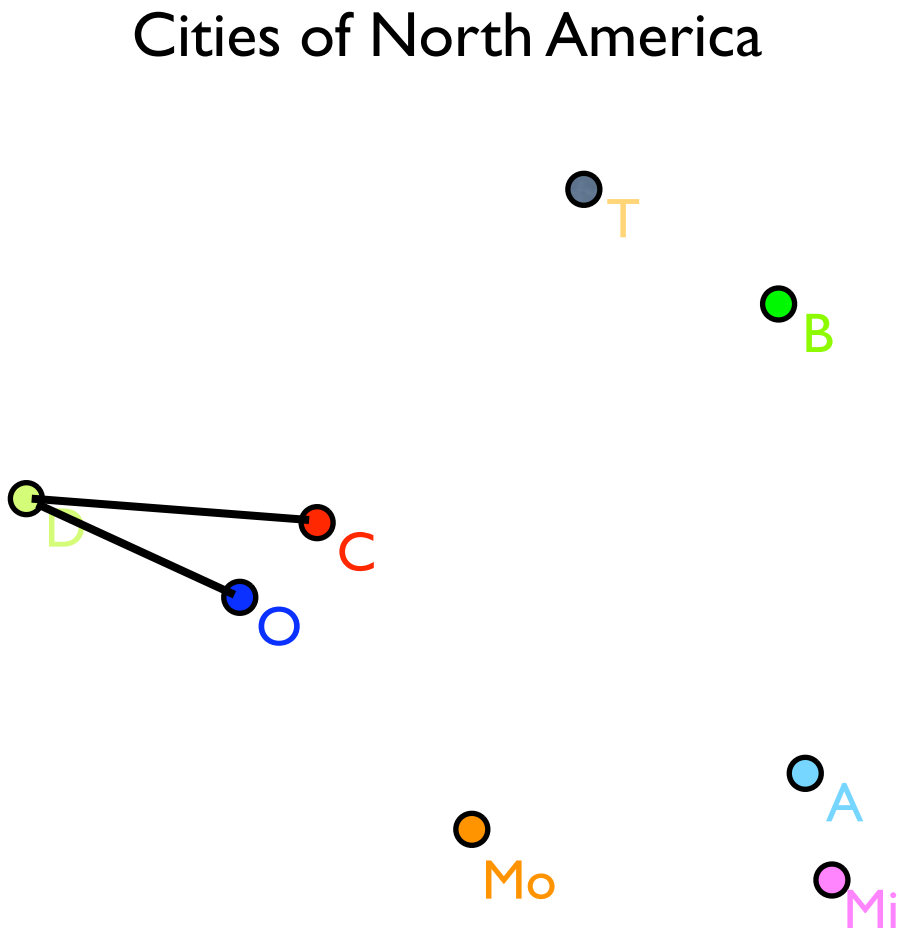
BoostMap: Main Ideas: Proximity Structure

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$

Example.

Some proximity function values for "Cities of North America"

$P(D; O,C) = +1$



BoostMap: Main Ideas: Proximity Structure

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$

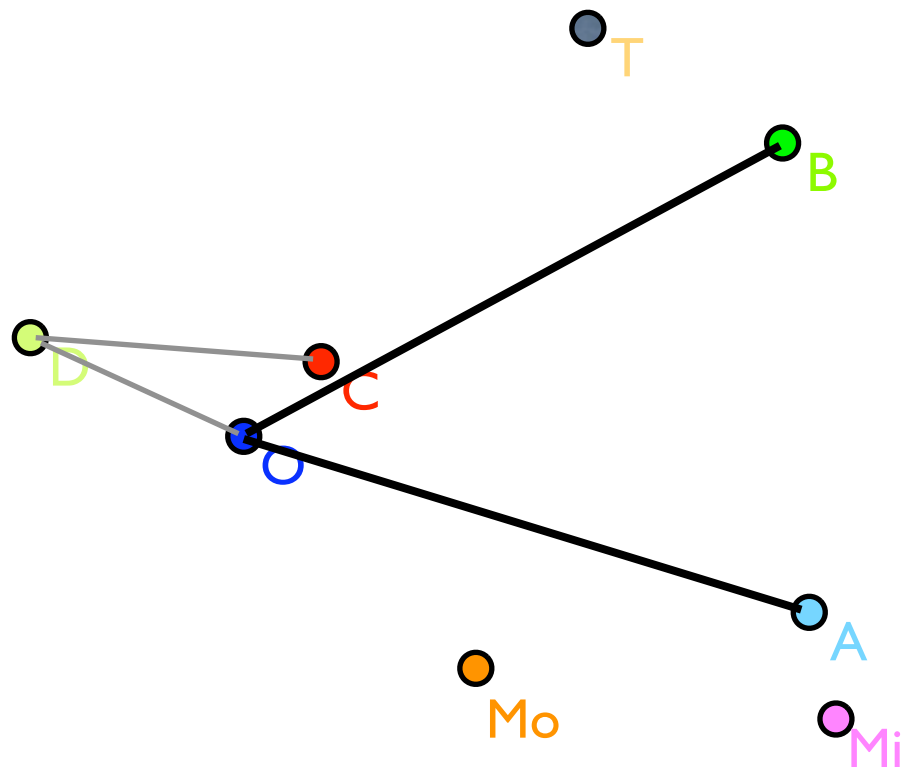
Example.

Cities of North America

Some proximity function values for "Cities of North America"

$P(D; O,C) = +1$

$P(O; B, A) = -1$



BoostMap: Main Ideas: Proximity Structure

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$

Example.

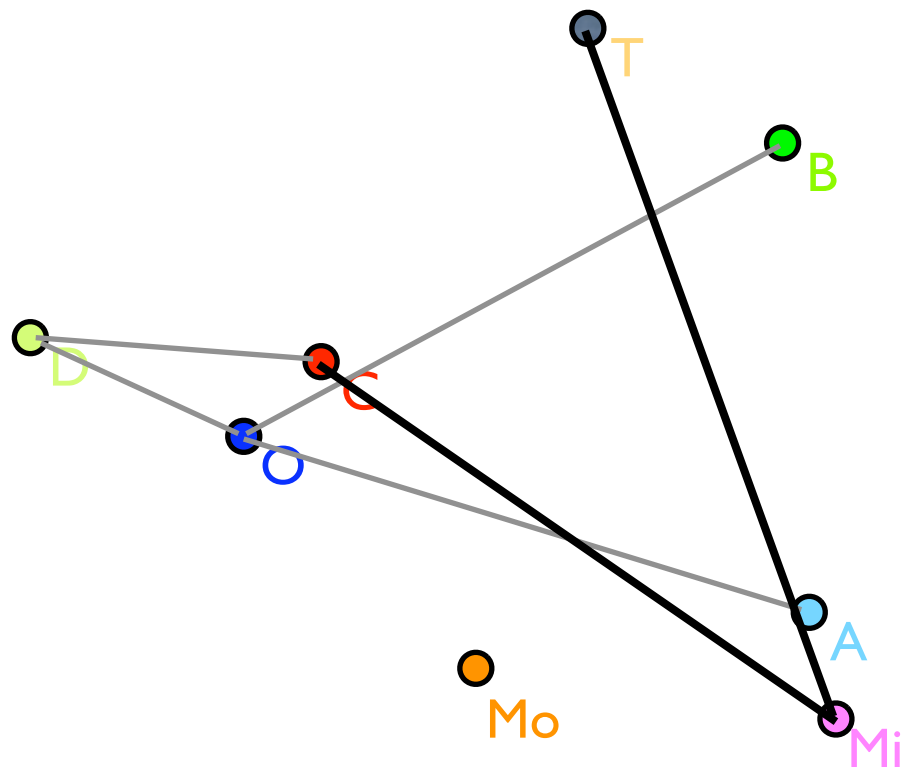
Some proximity function values for "Cities of North America"

$P(D; O,C) = +1$

$P(O; B, A) = -1$

$P(Mi; T, C) = -1$

Cities of North America



- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$

Ingredient 2. Find enough *weak representations* of the proximity ordering

Definition (mine). A *weak representation* of a proximity ordering is a 1-dimensional embedding

$$f : S \rightarrow \mathbf{R}$$

such that

Prob [$P(r; x,y) = \text{sgn}(|f(y) - f(r)| - |f(x) - f(r)|)$]

is greater than random.

- In machine learning parlance, a *weak classifier*.

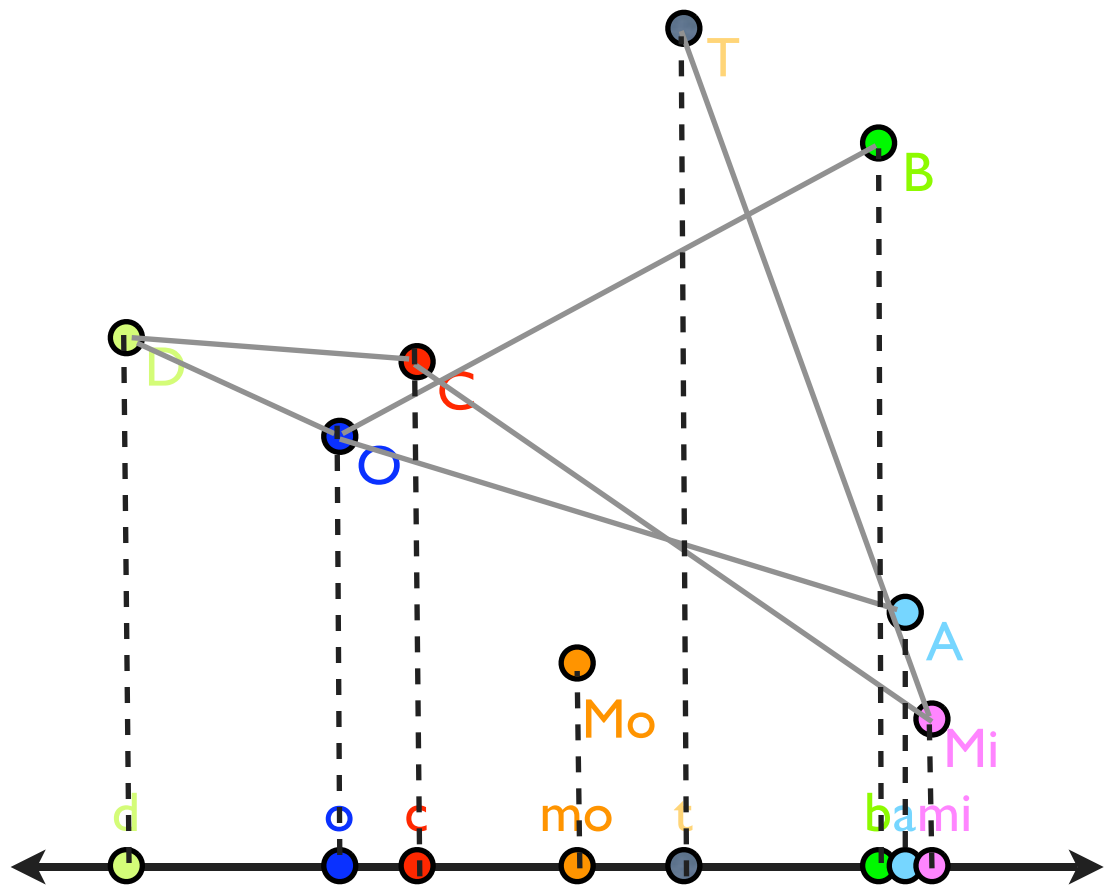
BoostMap: Main Ideas: Weak Representations

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$
- Weak representation S to $\mathbf{R}$. $Q(s; u,v) = \text{sgn}(|v - s| - |u - s|)$.

Examples.

$P(D; O,C) = +1 = Q(d; o,c)$
$P(O; B, A) = -1 \neq Q(o; b, a)$
$P(Mi; T, C) = -1 \neq Q(mi; t, c)$
Overall, better than random.

Cities of North America



BoostMap: Main Ideas: Weak Representations

- Data Set S, Distance function $D : S \times S \rightarrow \mathbf{R}^1$
- Proximity function $P(r; x,y) = \text{sgn}(D(y,r) - D(x,r))$
- Weak representation S to $\mathbf{R}$. $Q(s; u,v) = \text{sgn}(|v - s| - |u - s|)$.

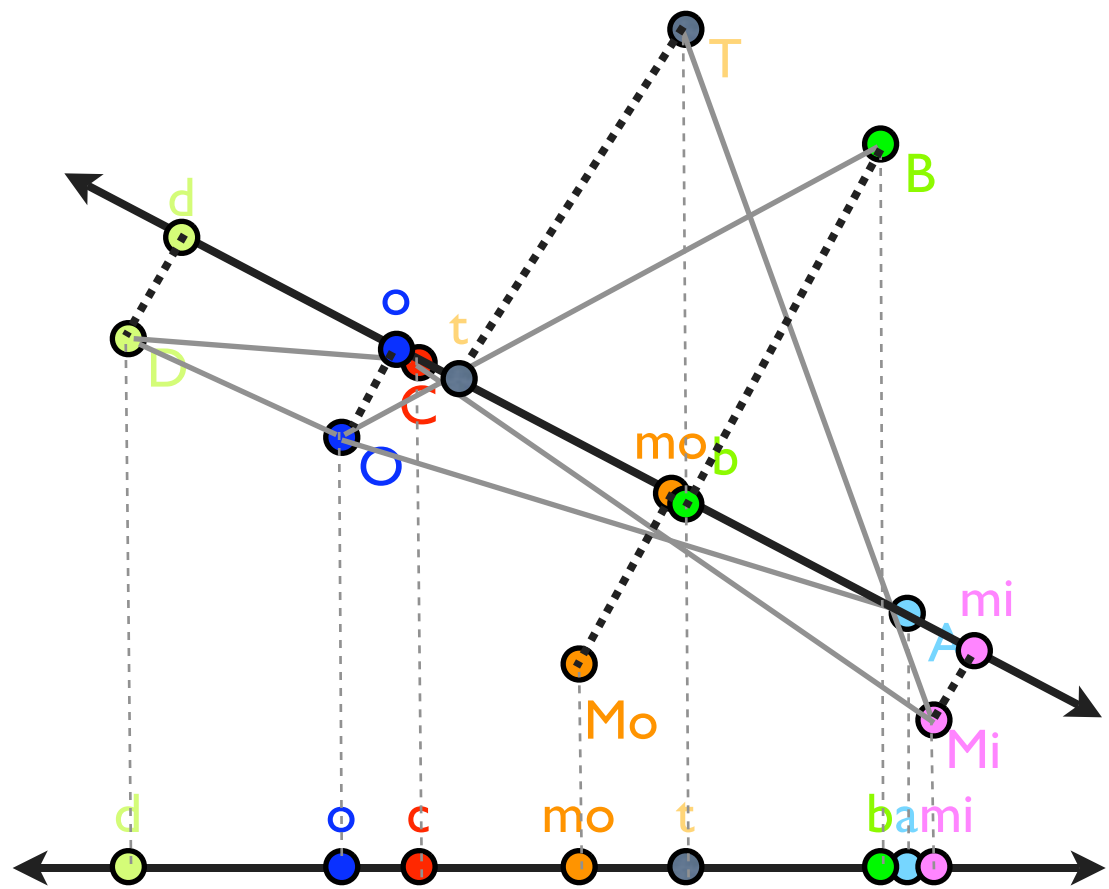
Examples.

$P(D; O,C) = +1 = Q(d; o,c)$
$P(O; B, A) = -1 \neq Q(o; b, a)$
$P(Mi; T, C) = -1 \neq Q(mi; t, c)$
Overall, better than random.

C A

$P(D; O,C) = +1 = Q(d; o,c)$
$P(O; B, A) = -1 \neq Q(o; b, a)$
$P(Mi; T, C) = -1 \neq Q(mi; t, c)$
Overall, better than random.

Cities of North America



BoostMap: Main Ideas: Strong Representation

- Data Set S, Distance function $D : S \times S \to \mathbf{R}^1$
- Proximity function $P(r; s,t) = \text{sgn}(D(t,r) - D(s,r))$
- Weak representation S to $\mathbf{R}$. $Q(w; u,v) = \text{sgn}(|v - w| - |u - w|)$.

Idea 3. Use AdaBoost to construct a *strong representation* to $\mathbf{R}^k$ from a set of weak representations.

Definition (mine). A *strong representation* of a proximity ordering on S is a k-dimensional embedding

$F = (f_1, f_2, ..., f_k) : S \to \mathbf{R}^k$ such that

(i) The projection to each coordinate axis, i.e., each coordinate function f_i, is a weak representation, and

(ii) There are positive real numbers $a_1 \ldots a_k$ such that the weighted L_1 distance $d(u,v) = \sum a_i |v_i - u_i|$ satisfies

$$\text{Prob}\ [\ P(r; s,t) = \text{sgn}(\ d(F(t) - F(r)) - d(F(s) - F(r))\ )\ ]$$

with high probability.

BoostMap: Main Ideas: Strong Representation

- Data Set S, Distance function $D : S \times S \to \mathbf{R}^1$
- Proximity function $P(r; s,t) = \text{sgn}(D(t,r) - D(s,r))$
- Weak representations $f : S \to \mathbf{R}$. $Q(w; u,v) = \text{sgn}(|v - w| - |u - w|)$.
- Strong representation $F = (f_1, f_2, ..., f_k) : S \to \mathbf{R}^k$. $d(u,v) = \sum a_i |v_i - u_i|$.

Comments. Let $h(u) = \sum a_i u_i$.

- In machine learning parlance, $H = h \circ F$ would be a strong classsifier.
- A "high probability" could be 95%, or 98% or 99%.
- If 100%, call H *proximity preserving*.
- More about choosing f's and finding a's, in next section.

BoostMap: Main Ideas: Strong Representation

- Data Set S, Distance function $D : S \times S \to \mathbf{R}^1$
- Proximity function $P(r; s,t) = \text{sgn}(D(t,r) - D(s,r))$
- Weak representations $f : S \to \mathbf{R}$. $Q(w; u,v) = \text{sgn}(|v - w| - |u - w|)$.
- Strong representation $F = (f_1, f_2, ..., f_k) : S \to \mathbf{R}^k$. $d(\mathbf{u},\mathbf{v}) = \sum a_i |v_i - u_i|$; $H(s) = \sum a_i f_i(s)$.

Step 4. Hope the outcome is publishable.

Remark. If you know about both *FastMap* (Faloutsos & Lin, 95) and *AdaBoost* (Freund & Schapire, '96), it's a very natural question to ask if the latter can be applied to the former. This paper seems to be the result of developing that idea.

Outline

- Main Ideas and Definitions
- The Algorithm
  - Input
  - 1 reference-point functions
  - 2 reference-point functions: 3 viewpoints
  - Process, output
  - Complexity, remarks.
- Examples & Remarks

AdaBoost Input. AdaBoost expects:

- Labeled training (testing, verification) data
- A set of weak classifiers (weak representations)

For BoostMap:

Make training data:

(a) Choose—randomly—a largish set of ordered triples (r, s, t) of data from S.

(b) Label each with $P(r;s,t) = \text{sgn}(D(t,r) - D(s,r))$.

(c) "Largish" might be $O(N)$, some small multiple of N.

To make weak classifiers, start with a subset C of objects from S

(a) Called *candidate set* in paper.

(b) Maybe $O(k)$, if desired embedding dimension k is known

(c) Precompute all distances $D(c,s)$, $c \in C$, $s \in S$

$C \subset S$, subset to use for constructing weak representations.

$Q(w; u,v) = \text{sgn}(|w - v| - |w - u|)$.

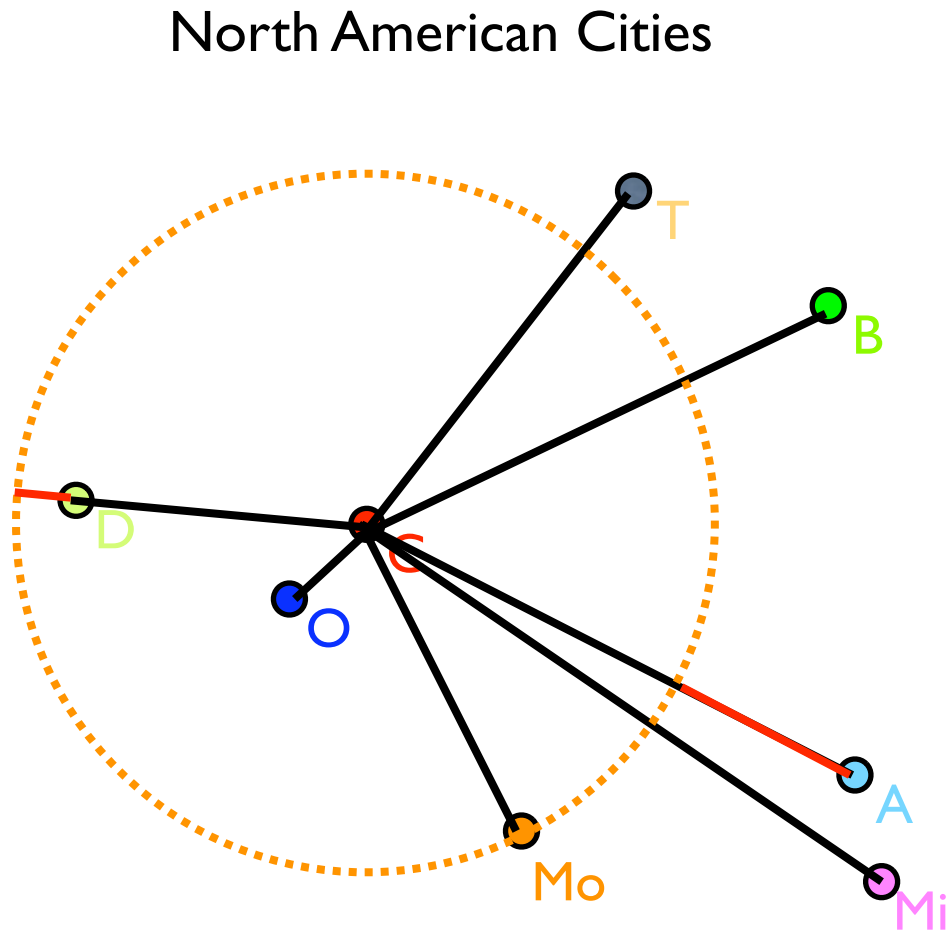
1-pivot weak representations $f : S \to \mathbf{R}$

For $c \in C$, define $f_c : S \to \mathbf{R}$ by

$$f_c(s) = D(c,s).$$

- For all triples of form $(c; s,t)$,
  $P(c; s,t) = Q(f_c(c); f_c(s), f_c(t))$.
- Is f_c a weak representation?
  - If triangle inequality holds, yes (I think).
  - Otherwise, ??? Not proved!
- Example. f_C

$P(Mo;A,D) = +1 \neq Q(f_c(Mo);f_c(A),f_c(D))$



$C \subset S$, subset to use for constructing weak representations.

$Q(w; u,v) = \text{sgn}(|w - v| - |w - u|)$.

1-pivot weak representations $f : S \to \mathbf{R}$

For $c \in C$, define $f_c : S \to \mathbf{R}$ by

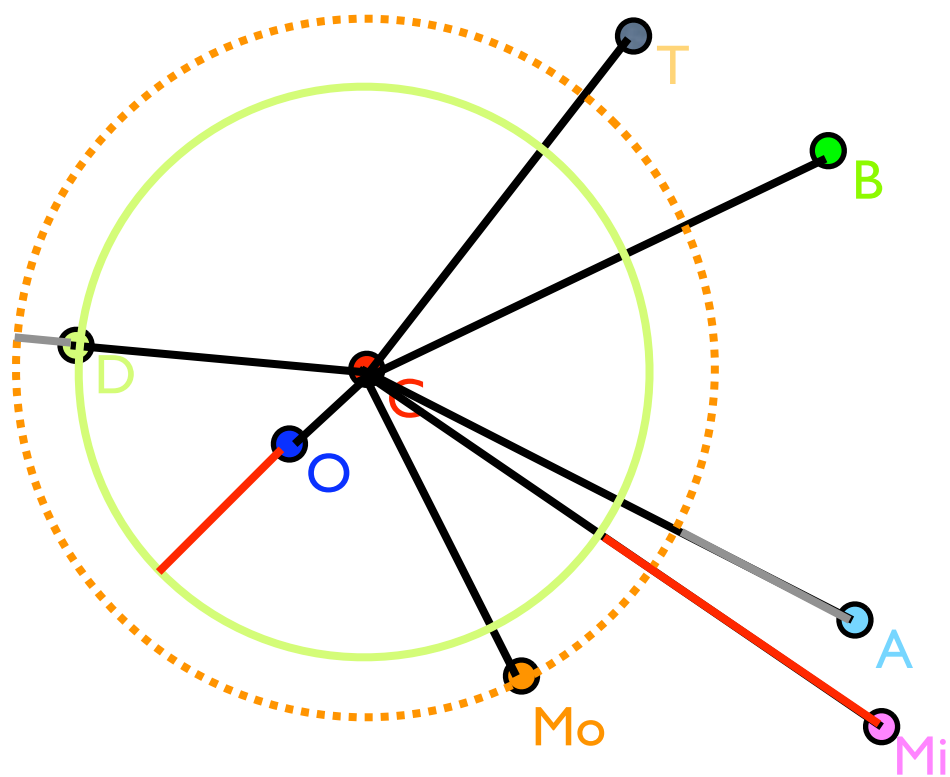
$$f_c(s) = D(c,s).$$

- For all triples of form $(c; s,t)$,
  $P(c; s,t) = Q(f_c(c); f_c(s), f_c(t))$.
- Is f_c a weak representation?
  - If triangle inequality holds, yes (I think).
  - Otherwise, ??? Not proved!
- Example. f_C

$P(Mo;A,D) = +1 \neq Q(f_c(Mo);f_c(A),f_c(D))$

$P(D;O,Mi) = +1 = Q(f_c(D);f_c(O),f_c(Mi))$

North American Cities



$C \subset S$, subset to use for constructing weak representations.

$Q(w; u,v) = \text{sgn}(|w - v| - |w - u|)$.

$f_c : S \to \mathbf{R} : f_c(s) = D(c,s)$, for $c \in C$, 1-pivot weak representations

2-Pivot Weak Representations: 3 Views

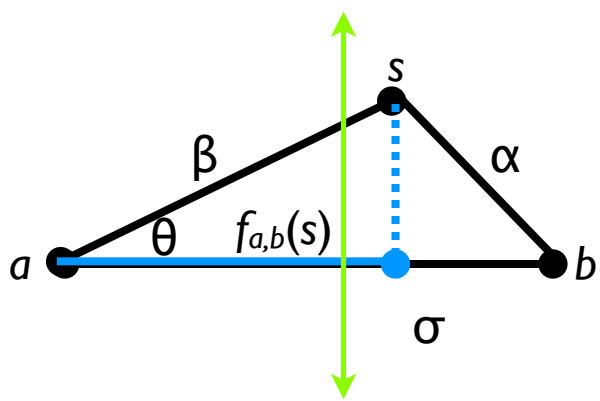
Viewpoint 1. Euclidean / geometric motivation. For $a, b \in C$, define

$$f_{a,b}(s) = \frac{D(s,a)^2 + D(a,b)^2 - D(s,b)^2}{2\,D(a,b)}$$

= orthogonal projection of s onto line ab

Remark. This formula occurs in FastMap.

In geometric case, $Q(f_{a,b}(s); f_{a,b}(a), f_{a,b}(b)) = \pm 1$ as s is closer to a or to b, i.e., on which side of the bisecting hyperplane s lies.



Law of Cosines: $\beta^2 + \sigma^2 - \alpha^2 = 2\beta\sigma \cos(\theta)$.
Rearrange, and find $f_{a,b}(s) =$ —

BoostMap: Algorithm: 2 pivots 3 ways

$C \subset S$, subset to use for constructing weak representations.

$Q(w; u,v) = \mathrm{sgn}(|w - v| - |w - u|)$.

$f_c : S \to \mathbf{R} : f_c(s) = D(c,s)$, for $c \in C$, 1-pivot weak representations

$f_{a,b} : S \to \mathbf{R} : f_{a,b}(s) = (D(s,a)^2 + D(a,b)^2 - D(s,b)^2) / 2\, D(a,b)$, for $a,b \in C$, 2-pivot weak representations (view 1)

Viewpoint 2. Motivated by rewriting $f_{a,b}$.

Say f is a weak representation and

$$g(s) = \alpha f(s) + \beta$$

for some real numbers α, β, $\alpha \neq 0$. Then

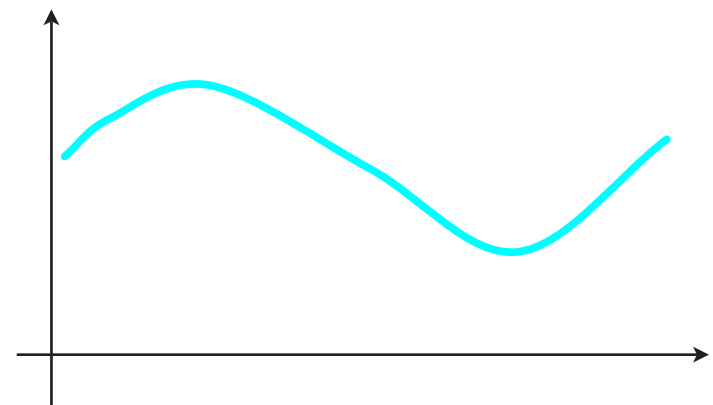
$$Q(g(r); g(s), g(t)) = Q(f(r); f(s), f(t)) .$$

In a linear combination of f's or g's, β is irrelevant, and α can be absorbed into the coefficients. So replace $f_{a,b}(s)$ by

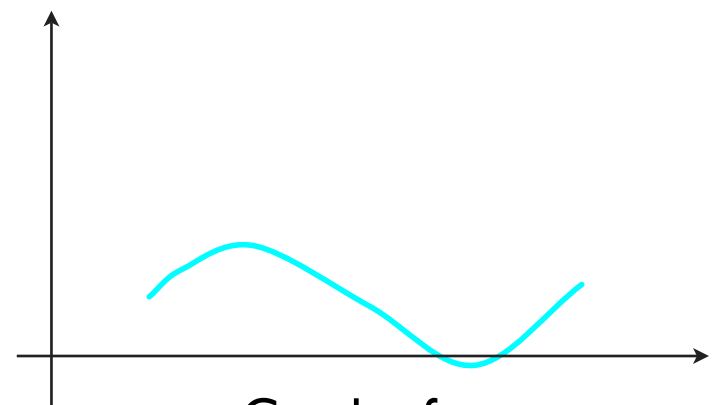
$$g_{a,b}(s) = D(s,b)^2 - D(s,a)^2 ,$$

whose $\pm$ sign differentiates "closer to a" from "closer to b."

N.B. $g_{a,b}(s)$ is a weak representation if and only if $f_{a,b}(s)$ is a weak representation.



Graph of f



Graph of g

BoostMap: Algorithm: 2 pivots 3 ways

$C \subset S$, selected to use for constructing weak representations.
For $a, b, c \in C$, $f_c(s) = D(c,s)$ defines a set of 1-pivot weak representations,
$f_{a,b}(s) = (D(s,a)^2 + D(a,b)^2 - D(s,b)^2) / 2\,D(a,b)$, for viewpoint 1, and
$g_{a,b}(s) = D(s,b)^2 - D(s,a)^2$, for viewpoint 2, define a set of 2-pivot weak representations
$Q(w; u,v) = \text{sgn}(\,|w - v| - |w - u|\,)$ in $\mathbf{R}$.

Viewpoint 3. Motivated by viewpoint 2 and linear algebra.
Since the presumed weak representations are to be combined into some linear combination to make a strong representation, why not just let

$$h_c(s) = D(c,s)^2 \;?$$

Then the hoped-for strong classifier will be a simple expression in $D(c,s)$ and $D(b,s)^2$ as c, b run over C. Mathematically exactly the same linear combinations are achievable.

- $h_c(s)$ is a weak representation if and only if $f_c(s)$ is.
- How will this affect performance, for process described below?

AdaBoost

AdaBoost expects as input:

A set T of labeled training data;

A set W of weak classifiers

AdaBoost then performs greedy search with updating weights for best linear combination $H = \sum \alpha_i f_i$ of functions in W.

Initial weight on each training object is $|T|^{-1}$, i.e., uniform distribution.

REPEAT: Given H_j at round j,

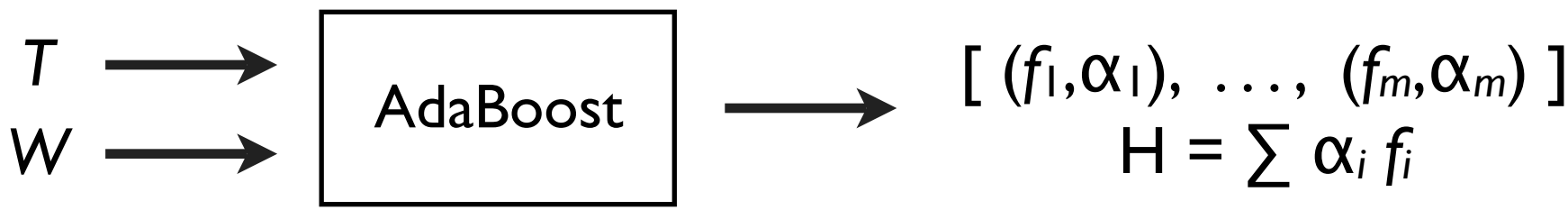
Find unused $f \in W$ with least weighted error.

Find (unique, positive) α so that $H_j + \alpha f$ induces least weighted error.

Update weights: more weight goes to misclassified samples.

UNTIL: no (f, α) leads to any improvement.

H is a real-valued function whose sign classifies, for a binary problem.



The AdaBoost Algorithm

Input: Set T of labeled training data $(\mathbf{t}, y)$, $y = \pm 1$ (for simplicity)
Set W of weak classifiers f .

Initialize: Weights $\omega_0(\mathbf{t}) = |T|^{-1}$ for all $\mathbf{t} \in T$.
Hypothesis $H_0 = 0$. Set of f occurring in H_0 is $B_0 = \varnothing$.

Do: At training round j ,

1. Determine whether current hypothesis H_j can be improved by modifying one or more coefficients α.
2. If no α have changed, let f_j be the $f \in W - B_{j-1}$ with least *training error* $\epsilon_j = \sum_{\mathbf{t} \in T} \omega_{j-1}(\mathbf{t})\ (\ f(\mathbf{t})\ y(\mathbf{t})\ )$, and $\alpha_j = 0.5 \ln((1+\epsilon_j) / (1-\epsilon_j))$;
3. Unless $\epsilon_j = 0$ or $\epsilon_j \geq 0.4999$, in which case return.
4. Update weights: $\omega_j(\mathbf{t}) = \omega_{j-1}(\mathbf{t}) \exp(-\alpha_j f(\mathbf{t})\ y(\mathbf{t})) / Z_j$, where Z_j is the normalization factor required to make $\sum_{\mathbf{t} \in T} \omega_j(\mathbf{t}) = 1$.

Output: Sequence $[\ (f_1,\alpha_1),\ \ldots,\ (f_m,\alpha_m)\ ]$ and hypothesis $H = \sum \alpha_j f_j$.

The BoostMap Procedure

AdaBoost expects: (i) Labeled training data; (ii) a set of weak classifiers

Training Set T: From data set S randomly choose triples (r,s,t) and label with proximity order $P(r; s,t) = \text{sgn}(D(r,t) - D(r,s))$. D is distance function for S.

Weak classifiers W: Choose subset $C \subset S$, for all $c \in C$, let $f_c(s) = D(c,s)$.
For $O(|C|)$ pairs (a,b), let $f_{a,b}(s) = [D(s,b)^2 + D(a,b)^2 - D(s,a)^2] / 2D(a,b)$;
OR $g_{a,b}(s) = D(s,b)^2 - D(s,a)^2$; OR for all $c \in C$, let $h_c(s) = D(c,s)^2$.

AdaBoost outputs: $[(f_1,\alpha_1), \ldots, (f_m,\alpha_m)]$ and $H = \sum \alpha_i f_i$.

To carry out similarity search with probably approximately correct results:
Embed S in $\mathbf{R}^k$ by $F(s) = (f_1(s), \ldots, f_k(s))$. ($k$ is number of terms in H)
Use weighted L_1 norm $||u||_H = \sum \alpha_i |u_i|$ on $\mathbf{R}^k$, and induced distance $d_H(\mathbf{u},\mathbf{v}) = \sum \alpha_i |v_i - u_i|$

BoostMap: Algorithm: Complexity

Data set S, with Distance function D.

Training Set T: Triples (r,s,t) labeled by $P(r; s,t) = \text{sgn}(D(r,t) - D(r,s))$.

Weak classifiers W: For $a,b,c \in C \subset S$,

$f_c(s) = D(c,s)$; $h_c(s) = D(c,s)^2$.

$f_{a,b}(s) = [D(s,b)^2 + D(a,b)^2 - D(s,a)^2] / 2D(a,b)$; $g_{a,b}(s) = D(s,b)^2 - D(s,a)^2$.

AdaBoost outputs: $[(f_1,\alpha_1), \ldots, (f_m,\alpha_m)]$ and $H = \sum \alpha_j f_j$.

Similarity search (probably approximately correct results):

$F(s) = (f_1(s), \ldots, f_k(s))$ embeds S in $\mathbf{R}^k$.

$d_H(\mathbf{u},\mathbf{v}) = \sum \alpha_i |v_i - u_i|$ distance on $\mathbf{R}^k$.

Complexity:

Construction: $O(r\,|T|\,|C|\,\delta)$, where

r is the number of rounds of training

δ is the time to evaluate $D(s,t)$.

Query: $O(k\,\delta)$.

Space: $\Omega(|S|)$, may vary with choice of access structure.

Outline

Main Ideas and Definitions

The Algorithm

Examples & Remarks

Experiments

The paper offers little guidance on how many (or which) hypotheses to use. But there is extensive literature on AdaBoost and Boosting

This experiment computer generated 26 hand signs each in 4128 3D orientations, for 107,328 data

- 200,000 triples for training
- |C| = 1000
- plus 1000 2-pivot weak rept'ns
- max(k) set to 256

"Chamfer distance" is distance function (see below).

Training took 2 day at 1.2GHz

Accuracy measured with 703 queries

- Median rank of exact nearest neighbor vs. number of dimensions
- Median highest ranking correct match vs. number of dimensions

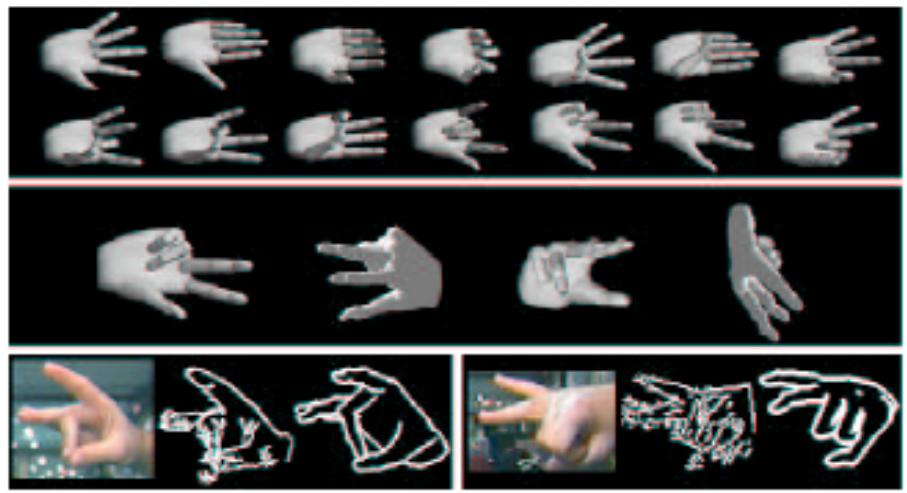
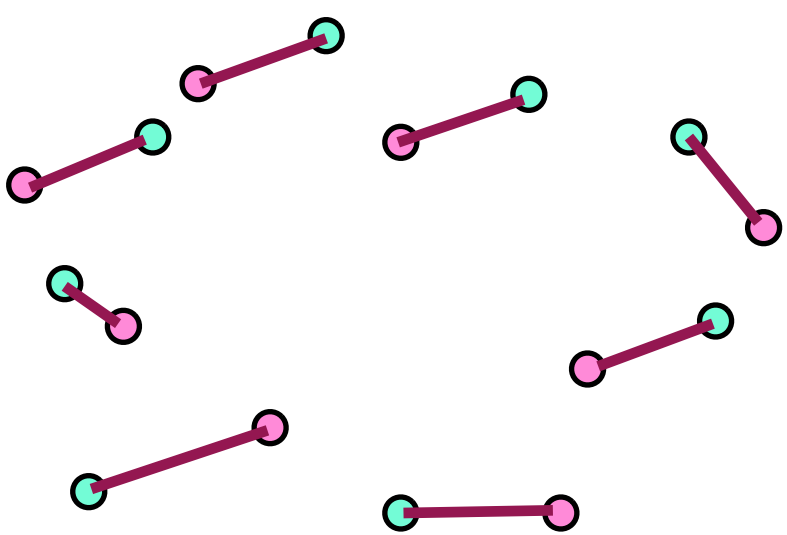


Figure 3: Top: 14 of the 26 hand shapes used to generate the hand database. Middle: four of the 4128 3D orientations of a hand shape. Bottom: for two test images we see, from left to right: the original hand image, the extracted edge image that was used as a query, and a correct match (noise-free computer-generated edge image) retrieved from the database.

BoostMap: Examples & Remarks

Chamfer Distance. Given two planar configurations of *n* points,

- Normalize to equal enclosed area
- Superimpose centroids
- Sum distances between corresponding points
- Take minimum over all relative rotations.



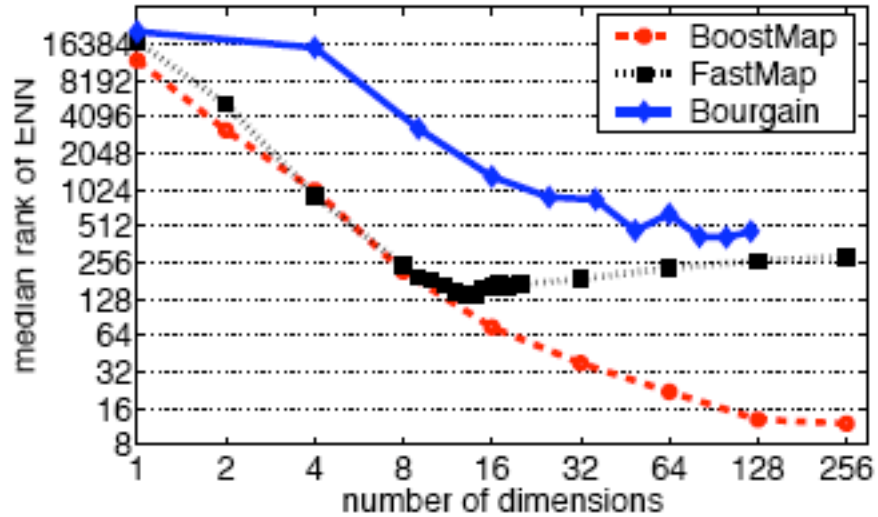


Figure 5: Median rank of exact nearest neighbor (ENN), versus number of dimensions, in approximate similarity rankings obtained using three different methods, for 703 queries to the hand database.

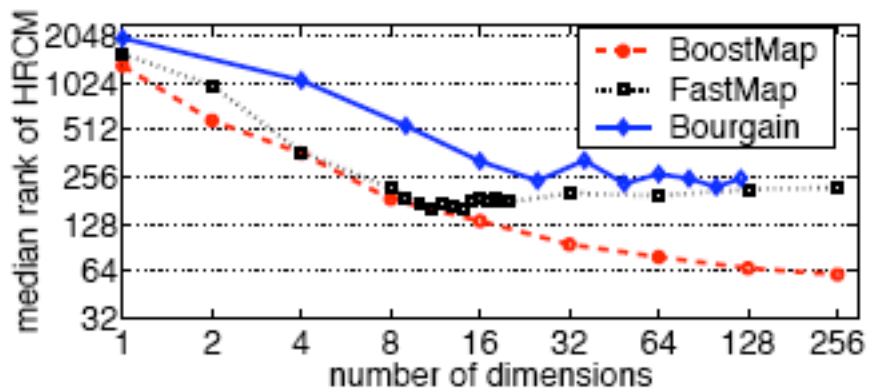


Figure 6: Median rank of highest ranking correct match (HRCM), versus number of dimensions, in approximate similarity rankings obtained using three different methods, for 703 queries to the hand database. For comparison, the median HRCM rank for the exact distance was 21.

BoostMap: Examples & Remarks

Another experiment was based on American Sign Language video, and an even more complicated distance measure based in part on optical flow.

Finally, for both these same two db's the paper used BoostMap together with "filter and refine." In particular, it sought to jointly optimize the dimension k and the number of matches to catch in the filter to find the correct first nearest neighbor for 95% or 100% of queries. (I found this not very convincing of anything.)

Remarks. Consider trade-offs between:

- Construction vs. query time.
- Embedding dimension (query time) vs. accuracy
- Different kinds of data sets, for which is BoostMap vs other options a good choice?
  - "Continuous" data with clusters?
  - Lengthy Boolean vectors, Hamming distance?

With images, for example, note decoupling of original embedding based on pixel intensities and classification based on (human) perception.

References

Athitsos, Alon, Sclaroff & Kollios, BoostMap: A method for efficient Approximate Similarity Rankings, CVPR, vol. II, 2004, pp. 268–275.

Schapire & Singer, Improving boosting algorithems using confidence-rated predictions, *Machine Learning* **37**, 1999, 297–336.

Faloutsos & Lin, *FastMap*: A fast algorithm for indexing, data-mining and visualization of traditionsl and multimedia datasets, SIGMOD, 1995, 163–174.

Hjaltason & Samet, Properties of embedding methods for similarity searching in metric spaces, *IEEE Trans. Pattern Analysis & Machine Intelligence* **25**, 2003, pp. 530–549.

G. Rätsch, RBF and Regularized AdaBoost software package (for Matlab), http://users.rsise.anu.edu.au/~raetsch/Software.html