New Horizons in Machine Learning

Avrim Blum CMU

This is mostly a survey, but portions near the end are joint work with Nina Balcan and Santosh Vempala

[Workshop on New Horizons in Computing, Kyoto 2005]

What is Machine Learning?

- Design of programs that adapt from experience, identify patterns in data.
- Used to:
  - recognize speech, faces, images
  - steer a car,
  - play games,
  - categorize documents, info retrieval, ...
- Goals of ML theory: develop models, analyze algorithmic and statistical issues involved.

Plan for this talk

- Discuss some of current challenges and "hot topics".
- Focus on topic of "kernel methods", and connections to random projection, embeddings.

- Start with a quick orientation…

The concept learning setting

- Imagine you want a computer program to help you decide which email messages are spam and which are important.
- Might represent each message by n features. (e.g., return address, keywords, spelling, etc.)
- Take sample S of data, labeled according to whether they were/weren't spam.
- Goal of algorithm is to use data seen so far to produce good prediction rule (a "hypothesis") $h(x)$ for future data.

The concept learning setting

E.g.,

money	pills	Mr.	bad spelling	known-sender	spam?
Y	N	Y	Y	N	Y
N	N	N	Y	Y	N
N	Y	N	N	N	Y
Y	N	N	N	Y	N
N	N	Y	N	Y	N
Y	N	N	Y	N	Y
N	N	Y	N	N	N
N	Y	N	Y	N	Y

example (highlighted row) — *label* (spam? column)

Given data, some reasonable rules might be:

- Predict SPAM if unknown AND (money OR pills)
- Predict SPAM if money + pills – known > 0.
- ...

Big questions

(A) How to optimize?

- How might we automatically generate rules like this that do well on observed data? [Algorithm design]

(B) What to optimize?

- Our real goal is to do well on new data.
- What kind of confidence do we have that rules that do well on sample will do well in the future?
  - Statistics
  - Sample complexity
  - SRM

for a given learning alg, how much data do we need...

To be a little more formal…

PAC model setup:

- Alg is given sample $S = \{(x, l)\}$ drawn from some distribution D over examples x, labeled by some target function f.
- Alg does optimization over S to produce some hypothesis $h \in H$. [e.g., H = linear separators]
- Goal is for h to be close to f over D.
  - $\Pr_{x \in D}(h(x) \neq f(x)) \leq \varepsilon$.
- Allow failure with small prob δ (to allow for chance that S is not representative).

The issue of sample-complexity

- We want to do well on D, but all we have is S.
  - Are we in trouble?
  - How big does S have to be so that low error on S $\Rightarrow$ low error on D?
- Luckily, simple sample-complexity bounds:
  - If $|S| \geq (1/\varepsilon)[\log|H| + \log 1/\delta]$,

    [think of log|H| as the number of bits to write down h]

    then whp $(1-\delta)$, all $h \in H$ that agree with S have true error $\leq \varepsilon$.
  - In fact, with extra factor of $O(1/\varepsilon)$, enough so whp all have |true error - empirical error| $\leq \varepsilon$.

The issue of sample-complexity

- We want to do well on D, but all we have is S.
  - Are we in trouble?
  - How big does S have to be so that low error on S $\Rightarrow$ low error on D?

- Implication:
  - If we view cost of examples as comparable to cost of computation, then don't have to worry about data cost since just ~ $1/\varepsilon$ per bit output.
  - But, in practice, costs often wildly different, so sample-complexity issues are crucial.

Some current hot topics in ML

- More precise confidence bounds, as a function of observable quantities.
  - Replace log |H| with log(# ways of splitting S using functions in H).
  - Bounds based on margins: how well-separated the data is.
  - Bounds based on other observable properties of S and relation of S to H; other complexity measures...

Some current hot topics in ML

- More precise confidence bounds, as a function of observable quantities.
- Kernel methods.
  - Allow to implicitly map data into higher-dimensional space, without paying for it if algorithm can be "kernelized".
  - Get back to this in a few minutes…
  - Point is: if, say, data not linearly separable in original space, it could be in new space.

Some current hot topics in ML

- More precise confidence bounds, as a function of observable quantities.
- Kernel methods.
- Semi-supervised learning.
  - Using labeled and unlabeled data together (often unlabeled data is much more plentiful).
  - Useful if have beliefs about not just form of target but also its relationship to underlying distribution.
  - Co-training, graph-based methods, transductive SVM,…

Some current hot topics in ML

- More precise confidence bounds, as a function of observable quantities.
- Kernel methods.
- Semi-supervised learning.
- Online learning / adaptive game playing.
  - Classic strategies with excellent regret bounds (from Hannan in 1950s to weighted-majority in 80s-90s).
  - New work on strategies that can efficiently handle large implicit choice spaces. [KV][Z]…
  - Connections to game-theoretic equilibria.

Some current hot topics in ML

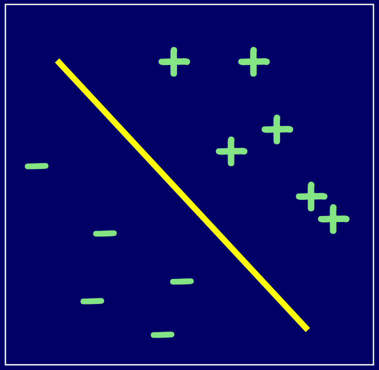
- More precise confidence bounds, as a function of observable quantities.
- Kernel methods.
- Semi-supervised learning.
- Online learning / adaptive game playing.

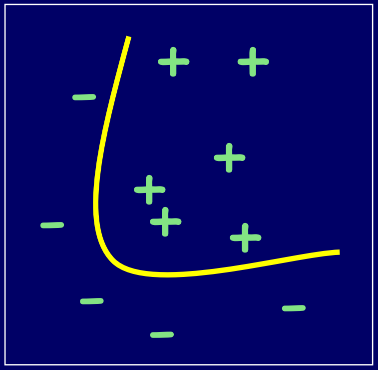
Could give full talk on any one of these.

Focus on #2, with connection to random projection and metric embeddings…

Kernel Methods

- One of the most natural approaches to learning is to try to learn a linear separator.



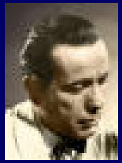
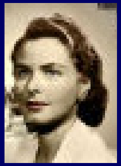


- But what if the data is not linearly separable? Yet you still want to use the same algorithm.
- One idea: Kernel functions.

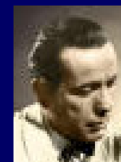
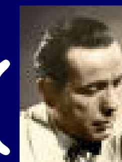
Kernel Methods

- A Kernel Function $K(x,y)$ is a function on pairs of examples, such that for some *implicit* function $\Phi(x)$ into a possibly high-dimensional space, $K(x,y) = \Phi(x) \cdot \Phi(y)$.
- E.g., $K(x,y) = (1 + x \cdot y)^m$.
  - If $x \in R^n$, then $\Phi(x) \in R^{n^m}$.
  - K is easy to compute, even though you can't even efficiently write down $\Phi(x)$.
- The point: many linear-separator algorithms can be *kernelized* – made to use K and act *as if* their input was the $\Phi(x)$'s.
  - E.g., Perceptron, SVM.

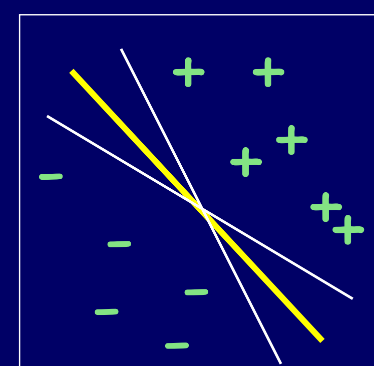
Typical application for Kernels

- Given a set of images:  , represented as pixels, want to distinguish men from women.
- But pixels not a great representation for image classification.
- Use a Kernel K(, ) = Φ()$\cdot\Phi$(), Φ is implicit, high-dimensional mapping. Choose K appropriate for type of data.

What about sample-complexity?

- Use a Kernel K(, ) = $\Phi($  $)\cdot\Phi($  $)$, Φ is implicit, high-dimensional mapping.
- What about # of samples needed?
  - Don't have to pay for dimensionality of Φ-space if data is separable by a large margin γ.
  - E.g., Perceptron, SVM need sample size only $\tilde{O}(1/\gamma^2)$.

$$|w\cdot\Phi(x)|/|\Phi(x)| \geq \gamma, \quad |w|=1$$



Φ–space

So, with that background…

Question

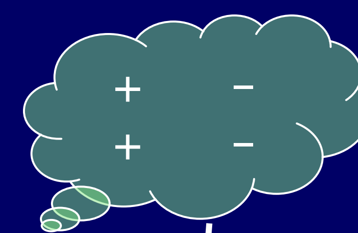
- Are kernels really allowing you to magically use power of implicit high-dimensional Φ-space without paying for it?
- What's going on?
- Claim: [BBV] Given a kernel [as a black-box program K(x,y)] and access to typical inputs [samples from D],
  - Can run K and reverse-engineer an explicit (small) set of features, such that if K is good [∃ large-margin separator in Φ-space for f,D], then this is a good feature set [∃ almost-as-good separator in this explicit space].

contd

- Claim: [BBV] Given a kernel [as a black-box program K(x,y)] & access to typical inputs [samples from D]
  - Can run K and reverse-engineer an explicit (small) set of features, such that if K is good [∃ large-margin separator in Φ-space], then this is a good feature set [∃ almost-as-good separator in this explicit space].
- Eg, sample $z^1,\ldots,z^d$ from D. Given x, define $x_i=K(x,z^i)$.
- Implications:
  - Practical: alternative to kernelizing the algorithm.
  - Conceptual: View choosing a kernel like choosing a (distrib dependent) set of features, rather than "magic power of implicit high dimensional space". [though argument needs existence of Φ functions]

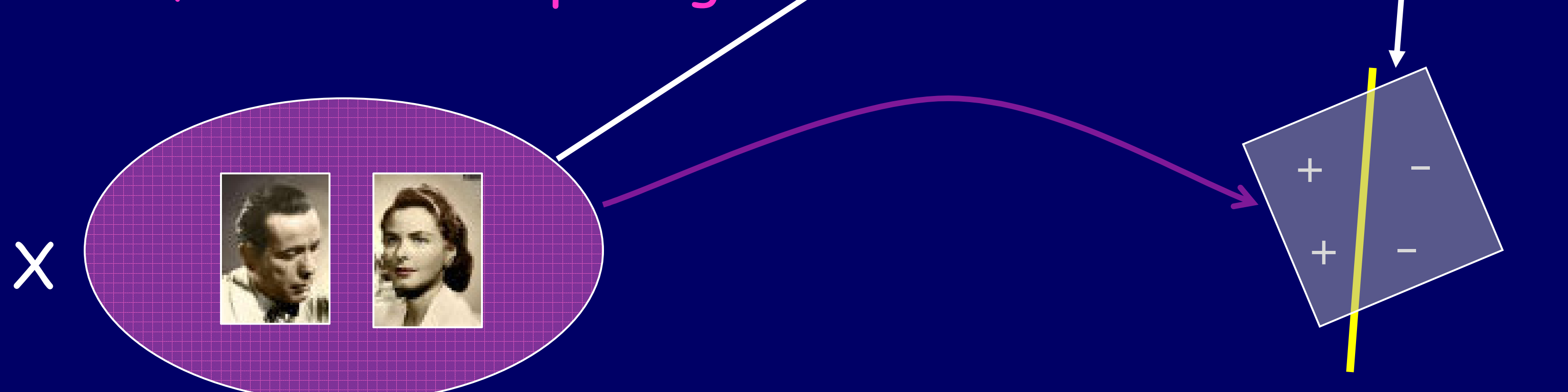
Why is this a plausible goal in principle?

- JL lemma: If data separable with margin γ in Φ-space, then with prob $1-\delta$, a *random* linear projection down to space of dimension $d = O((1/\gamma^2)\log[1/(\delta\varepsilon)])$ will have a linear separator of error $< \varepsilon$.



- If vectors are $r^1, r^2, \ldots, r^d$, then can view coords as features $x_i = \Phi(x) \cdot r^i$.

- Problem: uses Φ. Can we do directly, using K as black-box, without computing Φ?



3 methods (from simplest to best)

1. Draw d examples $z^1,\ldots,z^d$ from D. Use:

   $F(x) = (K(x,z^1), \ldots, K(x,z^d))$. [So, "$x_i$" = $K(x,z^i)$]

   For $d = (8/\varepsilon)[1/\gamma^2 + \ln 1/\delta]$, if separable with margin γ in Φ-space, then whp this will be separable with error ε. (but this method doesn't preserve margin).

2. Same d, but a little more complicated. Separable with error ε at margin $\gamma/2$.

3. Combine (2) with further projection as in JL lemma. Get d with log dependence on $1/\varepsilon$, rather than linear. So, can set $\varepsilon \ll 1/d$.

All these methods need access to D, unlike JL. Can this be removed? We show **NO** for generic K, but may be possible for natural K.

Actually, the argument is pretty easy...

(though we did try a lot of things first that didn't work...)

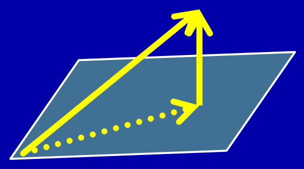
Key fact

Claim: If $\exists$ perfect w of margin γ in ϕ-space, then if draw $z^1,\ldots,z^d \in D$ for $d \geq (8/\varepsilon)[1/\gamma^2 + \ln 1/\delta]$, whp $(1-\delta)$ exists w' in $\mathrm{span}(\Phi(z^1),\ldots,\Phi(z^d))$ of error $\leq \varepsilon$ at margin $\gamma/2$.

Proof: Let S = examples drawn so far. Assume $|w|=1$, $|\Phi(z)|=1\ \forall z$.

- $w_{in} = \mathrm{proj}(w,\mathrm{span}(S))$, $w_{out} = w - w_{in}$.
- Say w_{out} is **large** if $\Pr_z\left(|w_{out}\cdot\Phi(z)| \geq \gamma/2\right) \geq \varepsilon$; else **small**.
- If small, then done: $w' = w_{in}$.
- Else, next z has at least ε prob of improving S.

$$|w_{out}|^2 \leftarrow |w_{out}|^2 - (\gamma/2)^2$$



- Can happen at most $4/\gamma^2$ times. □

So....

If draw $z^1,\ldots,z^d \in D$ for $d = (8/\varepsilon)[1/\gamma^2 + \ln 1/\delta]$, then whp exists w' in $\text{span}(\Phi(z^1),\ldots,\Phi(z^d))$ of error $\leq \varepsilon$ at margin $\gamma/2$.

- So, for some $w' = \alpha_1\Phi(z^1) + \ldots + \alpha_d\Phi(z^d)$,

$$\Pr_{(x,l) \in P}[\text{sign}(w' \cdot \Phi(x)) \neq l] \leq \varepsilon.$$

- But notice that $w' \cdot \Phi(x) = \alpha_1 K(x,z^1) + \ldots + \alpha_d K(x,z^d)$.

$\Rightarrow$ vector $(\alpha_1,\ldots\alpha_d)$ is an ε-good separator in the feature space: $x_i = K(x,z^i)$.

- But margin not preserved because length of target, examples not preserved.

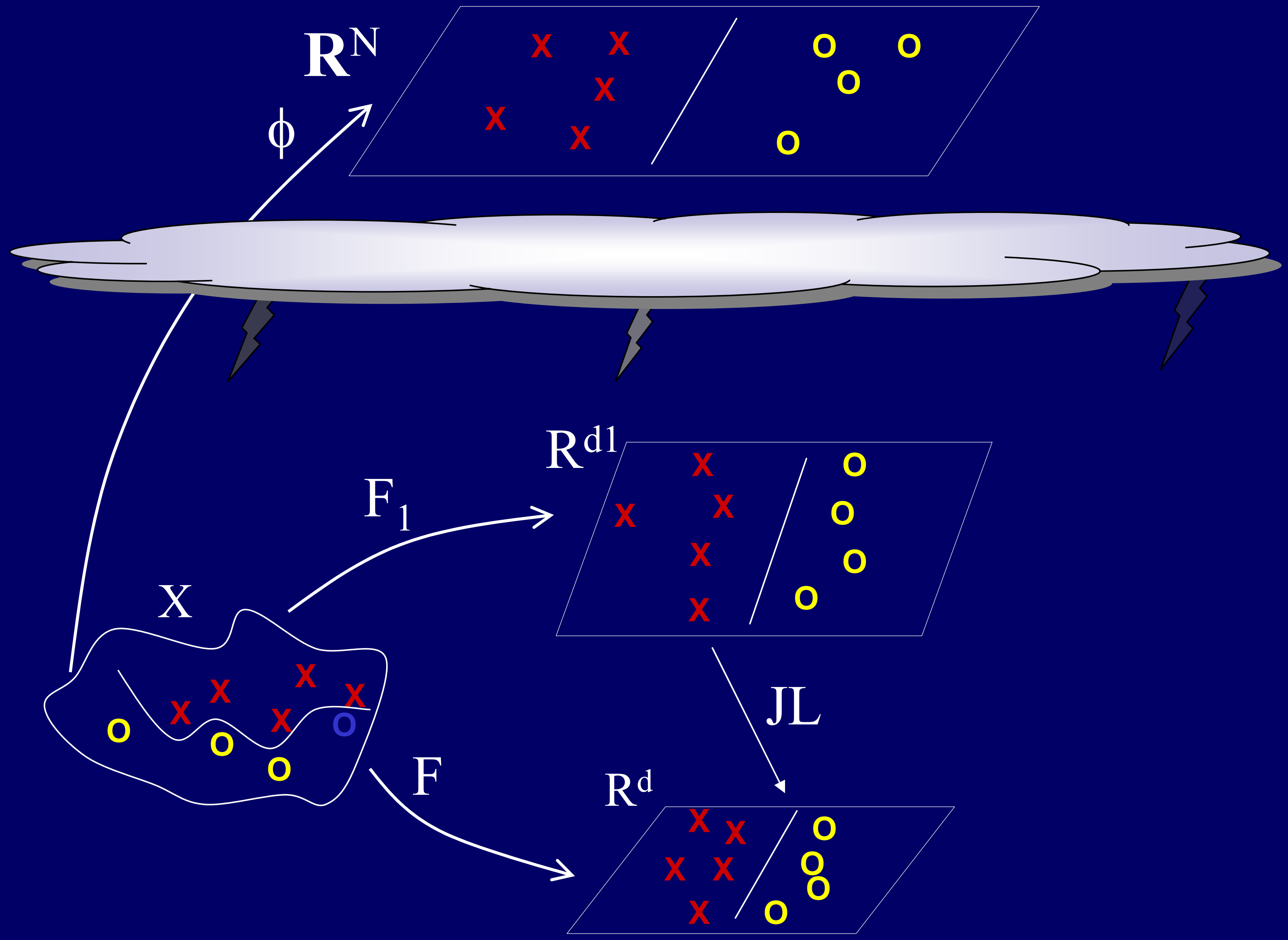
What if we want to preserve margin? (mapping 2)

- Problem with last mapping is $\Phi(z)$'s might be highly correlated. So, dot-product mapping doesn't preserve margin.

- Instead, given a new x, want to do an orthogonal projection of $\Phi(x)$ into that span. (preserves dot-product, decreases $|\Phi(x)|$, so only increases margin).
  - Run $K(z^i,z^j)$ for all $i,j=1,\ldots,d$. Get matrix M.
  - Decompose $M = U^T U$.
  - (Mapping #2) = (mapping #1)U^{-1}. □

Use this to improve dimension

- Current mapping gives $d = (8/\varepsilon)[1/\gamma^2 + \ln 1/\delta]$.
- Johnson-Lindenstrauss gives $d = O((1/\gamma^2) \log 1/(\delta\varepsilon))$. Nice because can have $d \ll 1/\varepsilon$. [So can set ε small enough so that whp a sample of size O(d) is perfectly separable]
- Can we achieve that efficiently?
- Answer: just combine the two...
  - Run Mapping #2, then do random projection down from that. (using fact that mapping #2 had a margin)
  - Gives us desired dimension (# features), though sample-complexity remains as in mapping #2.

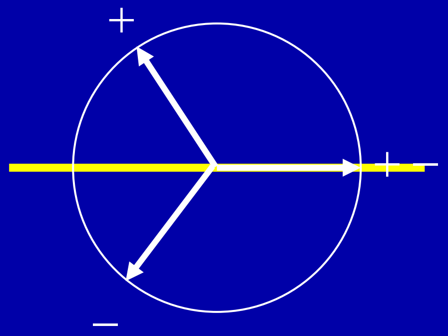


R^N
ϕ
R^{d1}
F_1
X
JL
F
R^d

Lower bound (on necessity of access to D)

For **arbitrary** black-box kernel K, can't hope to convert to small feature space without access to D.

- Consider $X=\{0,1\}^n$, random $X' \subset X$ of size $2^{n/2}$, D = uniform over X'.
- c = arbitrary function (so learning is hopeless).
- But we have this magic kernel $K(x,y) = \Phi(x)\cdot\Phi(y)$
  - $\Phi(x) = (1,0)$ if $x \notin X'$.
  - $\Phi(x) = (-\frac{1}{2}, \sqrt{3}/2)$ if $x \in X'$, $c(x)$=pos.
  - $\Phi(x) = (-\frac{1}{2}, -\sqrt{3}/2)$ if $x \in X'$, $c(x)$=neg.
- P is separable with margin $\sqrt{3}/2$ in Φ-space.
- But, without access to D, all attempts at running $K(x,y)$ will give answer of 1.



Open Problems

- For specific natural kernels, like "polynomial" kernel $K(x,y) = (1 + x\cdot y)^m$, is there an efficient analog to JL, without needing access to D?
  - Or, can one at least reduce the sample-complexity ? (use fewer accesses to D)
  - This would increase practicality of this approach
- Can one extend results (e.g., mapping #1: $x \rightarrow [K(x,z^1), \ldots, K(x,z^d)]$) to more general similarity functions K?
  - Not exactly clear what theorem statement would look like.