# Thoughts on Learning and Clustering

Avrim Blum
Carnegie Mellon University

(Portions of this talk are based on work joint with Nina Balcan and Santosh Vempala [BBV04] [BB06] [BBVnn])

[Cornell colloquium talk, Jan 2007]

## A talk in 3 parts…

- Part 1: A quick, biased intro to machine learning and where we are today.
- Part 2: A new theoretical perspective on kernel functions and what makes them useful for learning.
- Part 3: Applications to understanding clustering.

Will get to what they are

## Part 1: A quick intro to machine learning

## Machine learning can be used to...

- recognize speech, handwriting, faces,
- identify patterns in data,
- play games,
- categorize documents, ...

Machine learning theory:

- Understand learning as a computational process.
- Prove guarantees for algorithms.
- Understand what types of guarantees we might hope to achieve.

## A typical 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 produce good prediction rule (a "hypothesis") *h(x)* for future data.

## The concept learning setting

E.g.,

| | sales | sex | Mr. | bad spelling | known-sender | spam? |
|---|---|---|---|---|---|---|
| Msg1 | Y | N | Y | Y | N | Y |
| Msg2 | N | N | N | Y | Y | N |
| Msg3 | N | Y | N | N | N | Y |
| Msg4 | Y | N | N | N | Y | N |
| Msg5 | N | N | Y | N | Y | N |
| Msg6 | Y | N | N | Y | N | Y |
| Msg7 | N | N | Y | N | N | N |
| Msg8 | N | Y | N | Y | N | Y |

Given data, some reasonable rules might be:

- Predict SPAM if unknown AND (sex OR sales)
- Predict SPAM if sales + sex – known > 0.
- ...

## Big questions

(A)How might we automatically generate rules that do well on observed data?
[algorithm design]

(B)What kind of confidence do we have that they will do well in the future?
[confidence bound / sample complexity]

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

## Natural framework (PAC)

- We are given sample $S = \{(x,\ell)\}$.
  - Assume x's chosen at random from some probability distribution D over instance space.
  - View labels $\ell$ as being produced by some (unknown) target function f.
- Alg does optimization over S to produce some hypothesis (prediction rule) h.
- Goal is for h to do well on new examples also from D. I.e., $\Pr_{x\sim D}[h(x)\neq f(x)] < \varepsilon$.

## Basic confidence/sample-complexity argument

- Suppose I have some set of rules H (the *hypothesis class*) that seem worth considering. E.g., OR-functions over n binary features.
- Consider a bad h (error > ε). Chance it is consistent with S is at most $(1-\varepsilon)^{|S|}$.

So, Pr[any bad $h \in H$ is consistent] < $|H|(1-\varepsilon)^{|S|}$ ,
< 0.01 for $|S| > (1/\varepsilon)[\ln(|H|) + \ln(100)]$.

- $2^n$ OR-functions, so in this case $\ln(|H|) < n$.
- So, roughly, if $|S| > 10n$, whp any OR-function consistent with S will have true error < 10% over D. So, we can be confident in output of algorithm that finds consistent h.

## Nice interpretation in terms of Occam's razor

William of Occam (~1320 AD):

"entities should not be multiplied unnecessarily" (in Latin)

Which we interpret as: "in general, prefer simpler explanations".

Why? Is this a good policy?

## Occam's razor (contd)

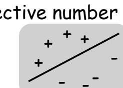
A computer-science-ish way of looking at it:

$|S| > (1/\varepsilon)[\ln(|H|) + \ln(100)]$

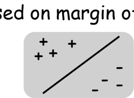
- Say "simple" = "short description".
- At most $2^b$ explanations can be < b bits long.
- So, if $|S| > 10b$, then can be confident in explanations of < b bits… because there are not too many of them, so it's unlikely a bad simple explanation will fool you just by chance.

## Extensions

- What about classes like "all linear separators"?
  - Replace log(|H|) with "effective number of degrees of freedom". (VC dimension)



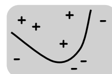
- What if dimension (# features) is very large?
  - Can instead give bounds based on margin of separation.



- What if have access to cheap unlabeled data?
  - Can use to adjust your description language. Reduce amount of labeled data needed.

# One last thing…

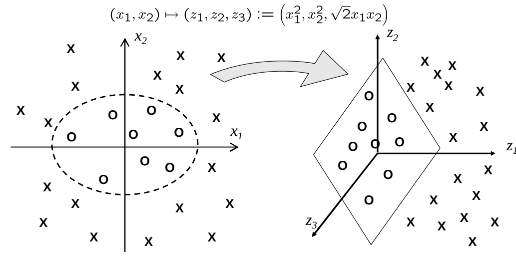
- We have a lot of great algorithms for learning linear separators (perceptron, SVM, …). But, a lot of time, data is not linearly separable.
  - "Old" answer: use a multi-layer neural network.
  - "New" answer: use a kernel function!
- Many algorithms only interact with the data via dot-products.
  - So, let's just re-define dot-product.
  - E.g., $K(x,y) = (1 + x \cdot y)^d$.
    - $K(x,y) = \phi(x) \cdot \phi(y)$, where $\phi()$ is implicit mapping into an $n^d$-dimensional space.
  - Algorithm acts as if data is in "$\phi$-space". Allows it to produce non-linear curve in original space.
  - Don't have to pay for high dimension if data is linearly separable there by a large margin.



# One last thing…

- E.g., for the case of n=2, d=2, the kernel $K(x,y) = (1 + x \cdot y)^d$ corresponds to the mapping:

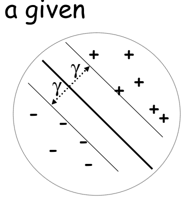
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$




Part 2: About those kernel functions...

# Kernel fns have become very popular

- Useful in practice for dealing with many different kinds of data.
  - Images , strings, ...
- Nice theory in terms of margins about what makes a given kernel good for a given learning problem.
  - If data is separable by large margin $\gamma$ in $\phi$-space, then need sample size only $\tilde{O}(1/\gamma^2)$ to get confidence in generalization.

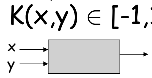



Assume $|\phi(x)| \leq 1$.

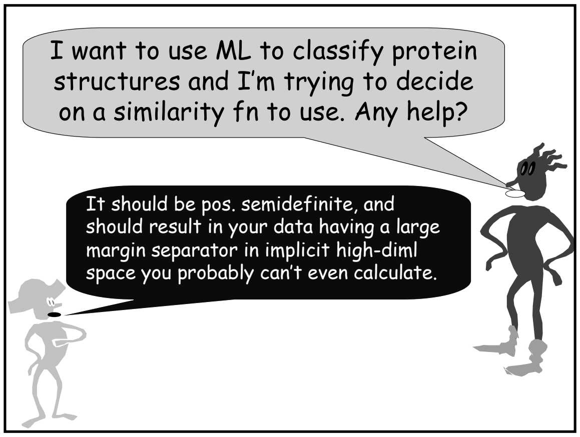
# Kernel fns have become very popular

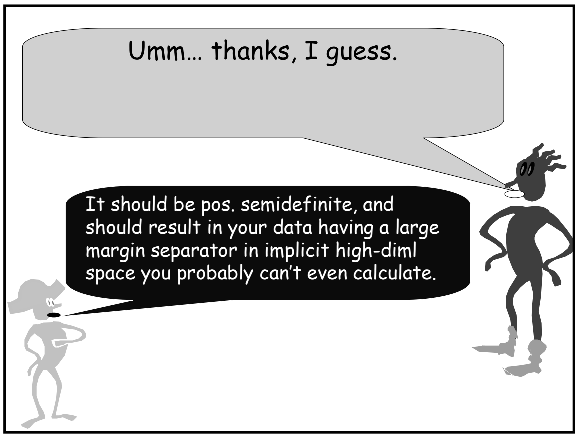
…but there's something a little funny:

- On the one hand, operationally a kernel is just a similarity function: $K(x,y) \in [-1,1]$, with some extra reqts.




- And in practice, people think of a good kernel as a good measure of similarity.
- But Theory talks about margins in implicit high-dimensional $\phi$-space. $K(x,y) = \phi(x) \cdot \phi(y)$.

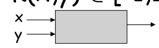





## Kernel fns have become very popular

…but there's something a little funny:

- On the one hand, operationally a kernel is just a similarity function: $K(x,y) \in [-1,1]$, with some extra reqts.




- But Theory talks about margins in implicit high-dimensional $\phi$-space. $K(x,y) = \phi(x)\cdot\phi(y)$.
  - Not great for intuition (do I expect this kernel or that one to work better for my kind of data)
  - Has a something-for-nothing feel to it. "All the power of the high-dim'l implicit space without having to pay for it". More prosaic explanation?

## Goal: notion of "good similarity function" for a learning problem that…

1. Talks in terms of more intuitive properties (no implicit high-diml spaces, no requirement of positive-semidefiniteness, etc)
2. If K satisfies these properties for our given problem, then has implications to learning
3. Is broad: includes usual notion of "good kernel" (one that induces a large margin separator in $\phi$-space).

## Defn satisfying (1) and (2):

- Say have a learning problem P (distribution D over examples labeled by unknown target f).
- Sim fn $K:(x,y)\rightarrow[-1,1]$ is $(\varepsilon,\gamma)$-good for P if at least a $1-\varepsilon$ fraction of examples x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

- E.g., suppose positives have $K(x,y) \geq 0.2$, negatives have $K(x,y) \geq 0.2$, but for a pos and a neg, $K(x,y)$ are uniform random in $[-1,1]$.
- Note: whp such a K is not a "legal" kernel.

## Defn satisfying (1) and (2):

- Say have a learning problem P (distribution D over examples labeled by unknown target f).
- Sim fn $K:(x,y)\rightarrow[-1,1]$ is $(\varepsilon,\gamma)$-good for P if at least a $1-\varepsilon$ fraction of examples x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

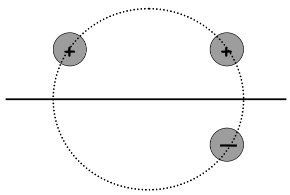
How can we use it?

## How to use it

At least a $1-\varepsilon$ prob mass of x satisfy:
$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

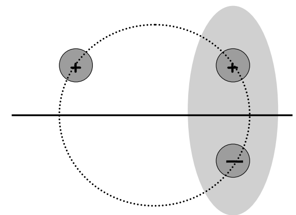
- Draw $S^+$ of $O((1/\gamma^2)\ln 1/\delta^2)$ positive examples.
- Draw $S^-$ of $O((1/\gamma^2)\ln 1/\delta^2)$ negative examples.
- Classify x based on which gives better score.
  - Hoeffding: for any given "good x", prob of error over draw of $S^+,S^-$ at most $\delta^2$.
  - So, at most $\delta$ chance our draw is bad on more than $\delta$ fraction of "good x".
- With prob $\geq 1-\delta$, error rate $\leq \varepsilon + \delta$.

## But not broad enough




- $K(x,y)=x \cdot y$ has good separator but doesn't satisfy defn. (half of positives are more similar to negs that to typical pos)

## But not broad enough




- Idea: would work if we didn't pick y's from top-left.
- Broaden to say: OK if $\exists$ large region R s.t. most x are on average more similar to $y \in R$ of same label than to $y \in R$ of other label. (even if don't know R in advance)

## Broader defn…

- Say $K:(x,y) \rightarrow [-1,1]$ is an $(\varepsilon,\gamma)$-good similarity function for P if exists a weighting function $w(y) \in [0,1]$ s.t. at least $1-\varepsilon$ frac. of x satisfy:

$$E_{y\sim D}[w(y)K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[w(y)K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

- Can still use for learning:
  - Draw $S^+ = \{y_1,\ldots,y_n\}$, $S^- = \{z_1,\ldots,z_n\}$. $n=\tilde{O}(1/\gamma^2)$
  - Use to "triangulate" data:
    $F(x) = [K(x,y_1), \ldots,K(x,y_n), K(x,z_1),\ldots,K(x,z_n)]$.
  - Whp, exists good separator in this space: $w = [w(y_1),\ldots,w(y_n),-w(z_1),\ldots,-w(z_n)]$

## Broader defn…

- Say $K:(x,y) \rightarrow [-1,1]$ is an $(\varepsilon,\gamma)$-good similarity function for P if exists a weighting function $w(y) \in [0,1]$ s.t. at least $1-\varepsilon$ frac. of x satisfy:

$$E_{y\sim D}[w(y)K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[w(y)K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

  - So, take new set of examples, project to this space, and run your favorite linear separator learning algorithm.*
  - Whp, exists good separator in this space: $w = [w(y_1),\ldots,w(y_n),-w(z_1),\ldots,-w(z_n)]$

*Technically bounds are better if adjust definition to penalize examples more that fail the inequality badly…

## And furthermore

- An $(\varepsilon,\gamma)$-good kernel [at least $1-\varepsilon$ fraction of x have margin $\geq \gamma$] is an $(\varepsilon',\gamma')$-good sim fn under this definition.
- But our current proofs suffer a penalty: $\varepsilon' = \varepsilon + \varepsilon_{extra}$, $\gamma' = \gamma^3\varepsilon_{extra}$.

Nati Srebro has improved to $\gamma^2$, which is tight, + extended to hinge-loss.

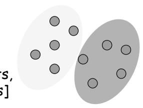
## Implications

- Statements about what makes a similarity fn useful for learning that don't require reference to implicit spaces.
- Includes usual notion of "good kernels" modulo the loss in some parameters.
  - Theory also holds for similarity fns that aren't necessarily positive-semidefinite (or even symmetric).
- May help with intuition when designing similarity fns for a given application.

# Part 3

## Can we use this angle to help think about clustering?

## Can we use this angle to help think about clustering?

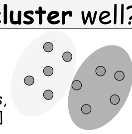


Consider the following setting:

- Given data set S of n objects. [documents, web pages]
- There is some (unknown) "ground truth" clustering. Each x has true label $\ell(x)$ in {1,…,t}. [topic]
- Goal: produce hypothesis h of low error up to isomorphism of label names.

Like learning from unlabeled data only.

## What conditions on a similarity function would be enough to allow one to **cluster** well?

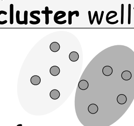


Consider the following setting:

- Given data set S of n objects. [documents, web pages]
- There is some (unknown) "ground truth" clustering. Each x has true label $\ell(x)$ in {1,…,t}. [topic]
- Goal: produce hypothesis h of low error up to isomorphism of label names.

Like learning from unlabeled data only.

## What conditions on a similarity function would be enough to allow one to **cluster** well?



Contrast with more standard approach to clustering analysis:

- Given as input a graph or embedding of points into $R^d$. View as "ground truth".
- Analyze abilities of algorithms to achieve different optimization criteria.
- Argue about which criterion produces better-looking results.
- Here, we flip this around.

## What conditions on a similarity function would be enough to allow one to **cluster** well?

<u>Here is a condition that trivially works:</u>

Suppose K has property that:

- $K(x,y) > 0$ for all x,y such that $\ell(x) = \ell(y)$.
- $K(x,y) < 0$ for all x,y such that $\ell(x) \neq \ell(y)$.

If we have such a K, then clustering is easy.

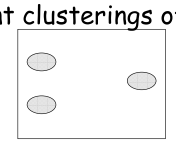
Now, let's try to make this condition a little weaker….

## What conditions on a similarity function would be enough to allow one to **cluster** well?

Suppose K has property that exists c:

- $K(x,y) > c$ for all x,y such that $\ell(x) = \ell(y)$.
- $K(x,y) < c$ for all x,y such that $\ell(x) \neq \ell(y)$.

Problem: the same K can satisfy for two very different clusterings of the same data!



Unlike learning, you can't even test your hypotheses!

## Let's change our goals a bit…

OK to output a small number of clusterings such that at least one has low error.

– Like list-decoding

Now previous case is fine: exists c such that

- $K(x,y) > c$ for all x,y such that $\ell(x) = \ell(y)$.
- $K(x,y) < c$ for all x,y such that $\ell(x) \neq \ell(y)$.

At most n clusterings consistent. Can produce using Kruskal-like algorithm.

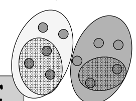
Condition is still a lot to ask though.
Can we weaken it?

## What if K is good for learning?

- Like in earlier part of talk….
- If # clusters t is small, $\gamma$ large, can do:
  - Pick $O(t/\gamma^2 \log t/\delta)$ random points.
  - Guess how they cluster.
  - Run learning alg to cluster remaining points.
  - Output all $t^{O(t/\gamma^2 \ldots)}$ different clusterings produced
- OK, maybe that's going overboard.
- Can we do better?

## What if you want to do better?

- Suppose our similarity function satisfies the stronger condition:
- Ground truth is "stable" in that



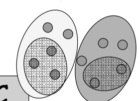
For all clusters C, C', for all A⊂C, A'⊂C': A and A' are not both more attracted to each other than to their own clusters.

K(x,y) is attraction between x and y

- Then, can construct a tree (hierarchical clustering) such that the correct clustering is some pruning of this tree.

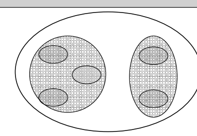
## What if you want to do better?

- Suppose our similarity function satisfies the stronger condition:
- Ground truth is "stable" in that



For all clusters C, C', for all A⊂C, A'⊂C': A and A' are not both more attracted to each other than to their own clusters.

K(x,y) is attraction between x and y



## Main point

- Exploring the question: what are minimal conditions on a similarity function that allow it to be useful for clustering?
  a. Allows algorithm to pick out right answer.
  b. Small number of candidate clusterings.
  c. Output a tree (hierarchical clustering) such that right answer is some pruning of it.
- Cases (b) or (c) can then allow for right answer to be identified with a little bit of additional feedback.

## Conclusions

- Theoretical approach to question: what are minimal conditions that allow a similarity to be useful for learning/clustering.
- For learning, formal way of analyzing kernels as similarity functions.
  - Doesn't require reference to implicit spaces or PSD properties.
- For clustering, "reverses" the usual view.
- Lot more to be done, esp in terms of other properties and objectives for clustering.
  - Perhaps objectives motivated by other forms of feedback.