Topics in Kernel Learning

Purushottam Kar*

Joint work with Harish Karnick* and Prateek Jain†

* Dept. of CSE, IIT, Kanpur

† Microsoft Research Labs India, Bengaluru

Alan Turing Centenary Year Celebrations, IIT Kanpur, October 9-10, 2012.

Learning problems

- Classification problems
  - An internet search yields 200 images of human faces. Someone sits and tags these as “male” or “female”. We end up with 100 male and 100 female faces. Can we use these to tag untagged face images on the internet as male/female ?
    - Assume that images of Justin Bieber and Cher have been discarded
  - A Gmail user tags 100 of his emails as spam. He tags another 100 emails in his inbox as useful. Can we now predict if a new incoming mail would be useful for the user or would he discard it as spam ?
    - Would the same predictions work for his friend ? His mom ?

Learning problems

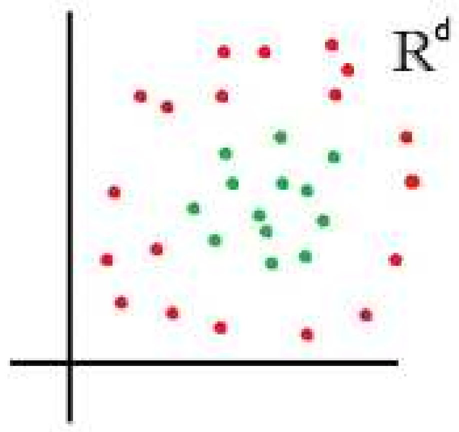
- Real-valued regression problems
  - We are given census data on the annual income and personal details (say average age) of family members for a 100 families in a locality along with their monthly expenditure. Can we use this data to predict the consumption patterns of other families for whom we know these details as well ?
  - An insurance company has the medical information about past customers such as heart rate, basal metabolic rate etc and their lifespan. Can it use this data to predict the expected lifespan of a new customer while issuing him a policy ?
- Clustering problems
- Structured learning problems

A mathematical abstraction

- Domain : $\mathfrak{X}$
  - Set of all 50 x 50 pixel images of faces
  - Set of all English language strings (emails)
- Label set : $\mathfrak{Y}$
  - $\{\pm 1\}$ in case of classification
  - $\mathbb{R}$ (or some subset thereof) in case of real valued regression
- True (unknown) pattern
  - $f^* : \mathfrak{X} \to \mathfrak{Y}$
  - We get to see the output of the true pattern (i.e. true labels) on a finite number of domain elements

The goal of supervised learning

- Given training data …
  - Lots of domain elements along with their true labels
  - $\left(x_1, f^*(x_1)\right), \ldots, \left(x_n, f^*(x_n)\right)$
- … learn a hypothesis pattern …
  - $h : \mathfrak{X} \to \mathfrak{Y}$
- … that is close to the true pattern …
  - $\Pr\left[h(x) \neq f^*(x)\right] \leq \epsilon$
- Some issues
  - How big should be my training set ?
  - How should I select my training set ?
  - Over what distribution are you calculating probabilities ?



Scope of this talk

- Most issues shall be resolved by simply locating a big enough carpet
- Shall mostly focus on one learning paradigm and look at some aspects of it
  - Kernel based learning algorithms
- For the more interested ... please talk to me !

Kernel learning : the basics

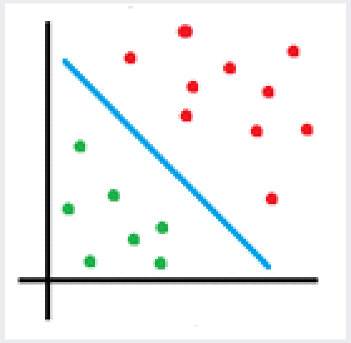
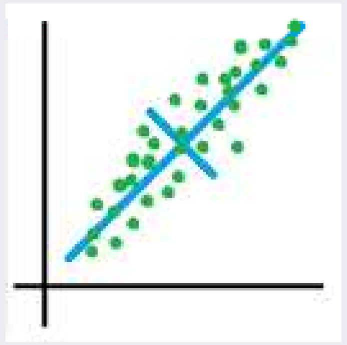
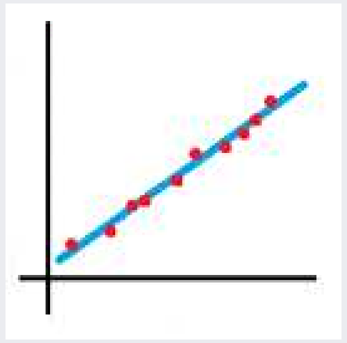
- Assume that the unknown pattern is "smooth“
  - Close-by points have similar labels
  - Label for a new point can be reconstructed using labels of training points close to the new point
- How to quantify "closeness" ?
  - Let $K : \mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$ be a measure of similarity (kernel)
  - E.g. on Euclidean spaces, the dot product is a kernel
  - Kernel methods use hypotheses of the form

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) f^*(x_i)$$

  - Different kernels will yield different hypotheses

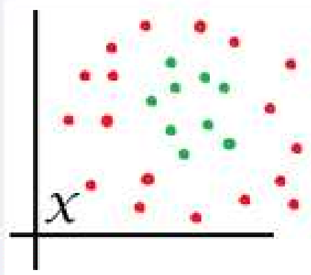
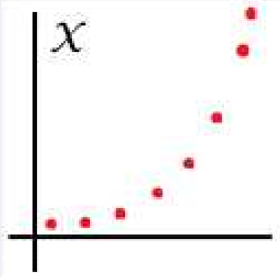
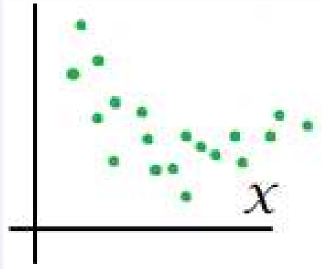
Kernel learning : a simple example

- Assume that $\mathfrak{X} \subset \mathbb{R}^2$ and $K(x_1, x_2) = \langle x_1, x_2 \rangle$
- The hypotheses look like $h(x) = \sum \alpha_i f^*(x_i) \langle x, x_i \rangle = \langle x, w \rangle$
  - where $w = \sum \alpha_i f^*(x_i) x_i \in \mathbb{R}^2$
- Finding a hypothesis involves finding (good) coefficients α_i
  - Solve a mathematical (optimization) program (classfn, regressn)
  - Perform spectral decomposition (PCA)

Problem	Classification	Regression	PCA
A thousand words			

Kernel learning : non-linear kernels

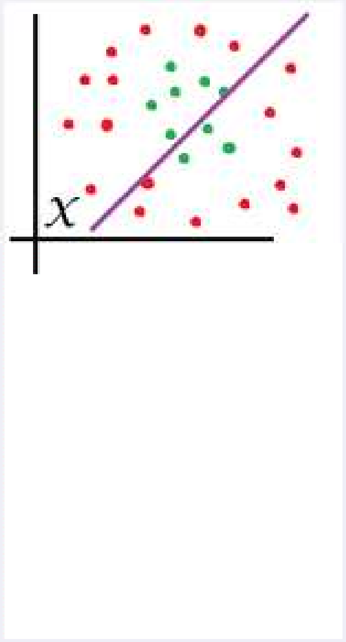
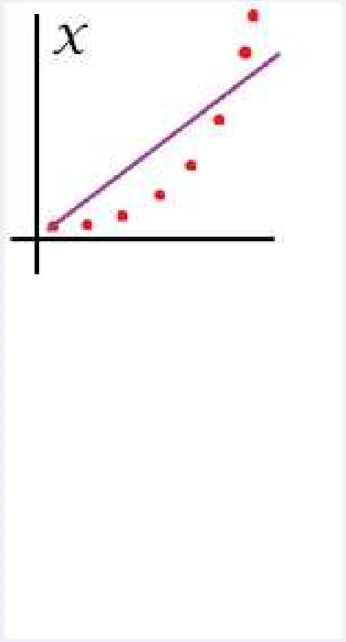
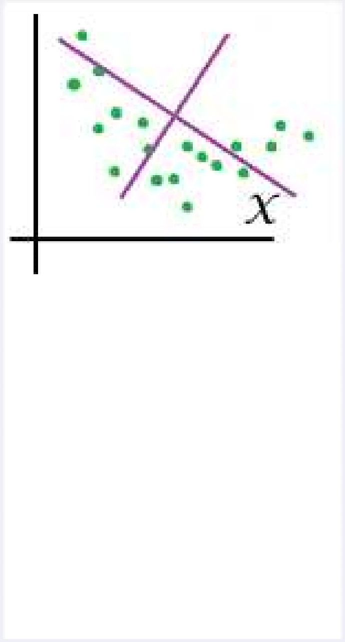
- Linear kernel is insufficient for most practical uses
  - use more complex non-linear kernels e.g. $K(x_1, x_2) = \langle x_1, x_2 \rangle^2$
  - the job of these kernels is to (implicitly) linearize the problem in some abstract space*

Problem	Classification	Regression	PCA
A thousand words			

*Reproducing Kernel Hilbert Space

Kernel learning : non-linear kernels

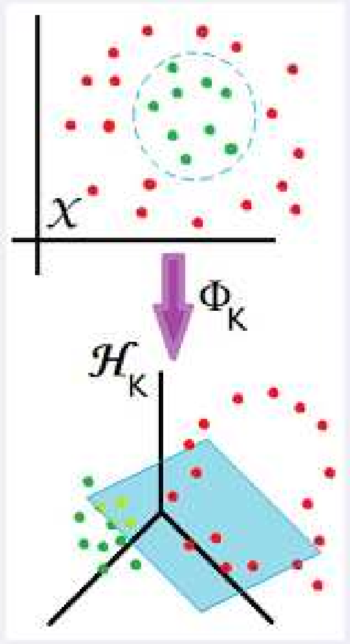
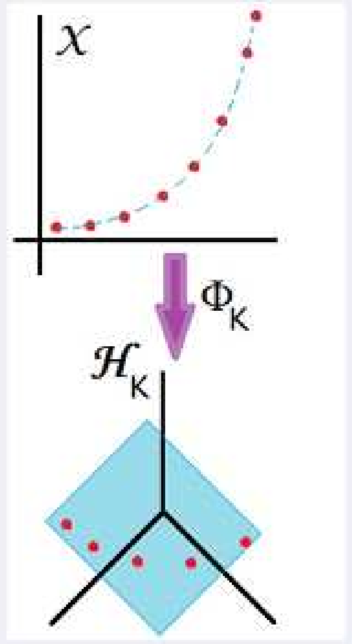
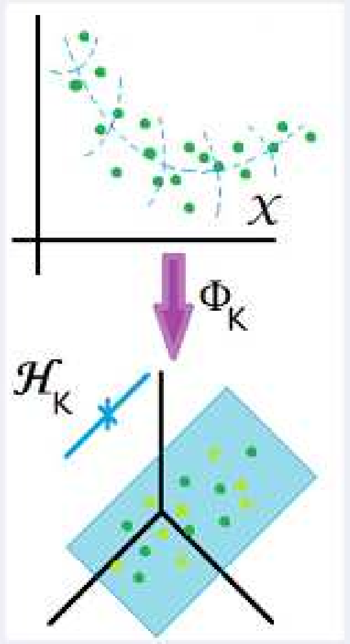
- Linear kernel is insufficient for most practical uses
  - use more complex non-linear kernels e.g. $K(x_1, x_2) = \langle x_1, x_2 \rangle^2$
  - the job of these kernels is to (implicitly) linearize the problem in some abstract space*

Problem	Classification	Regression	PCA
A thousand words			

*Reproducing Kernel Hilbert Space

Kernel learning : non-linear kernels

- Linear kernel is insufficient for most practical uses
  - use more complex non-linear kernels e.g. $K(x_1, x_2) = \langle x_1, x_2 \rangle^2$
  - the job of these kernels is to (implicitly) linearize the problem in some abstract space*

Problem	Classification	Regression	PCA
Two thousand words			

*Reproducing Kernel Hilbert Space

Kernel learning : some issues

- Programs are convex iff the kernel is *Positive Semi-definite*
  - Are non PSD similarity functions inherently unusable ?
  - Earth mover's distance, intersection similarity
- Even if the kernel is PSD, the resulting hypothesis function cannot be evaluated quickly
  - Recall the form of the hypothesis

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) f^*(x_i)$$

  - Must calculate the value $K(x, x_i)$ for every i such that $\alpha_i \neq 0$
  - Very slow for real time applications !

Learning with non-positive definite kernels

- Different kernels give rise to different hypotheses
  - There are "good" kernels and there are "not-so-good" kernels
- One can define notions of "goodness" formally
  - Classification : positives close to positives than negatives
  - A "good" kernel, whether PSD or not, can be used to (efficiently) learn a good hypothesis !
- Several results
  - Classification : [Balcan-Blum ICML06], [Wang *et al* ICML07], [Srebro COLT07], [K.-Jain NIPS11]
  - Other supervised learning problems [K.-Jain NIPS12]
  - Sparse learning to get fast hypotheses [K.-Jain NIPS12]

Accelerated kernel learning

- Consider a hypothesis $h(x) = \sum \alpha_i K(x, x_i) f^*(x_i)$
  - Assume that m of the α_i are non-zero
  - Getting the label for a new point takes $\Omega(m)$ time
- If the kernel is a bilinear function, we can do better
  - Example : $K(x_1, x_2) = \langle x_1, x_2 \rangle$

$$h(x) = K\left(x, \sum \alpha_i f^*(x_i)\right)$$

- Since $\sum \alpha_i f^*(x_i) x_i$ can be pre-computed, we can get the label in $O(1)$ time !
- Real-life problems require complex non-linear kernels for good performance ☹

Feature maps for fast kernel learning

- Can we approximate non-linear kernels by "linear" ones ?
  - Yes, for any kernel there **exist** "feature maps" $Z : \mathfrak{X} \to \mathbb{R}^D$

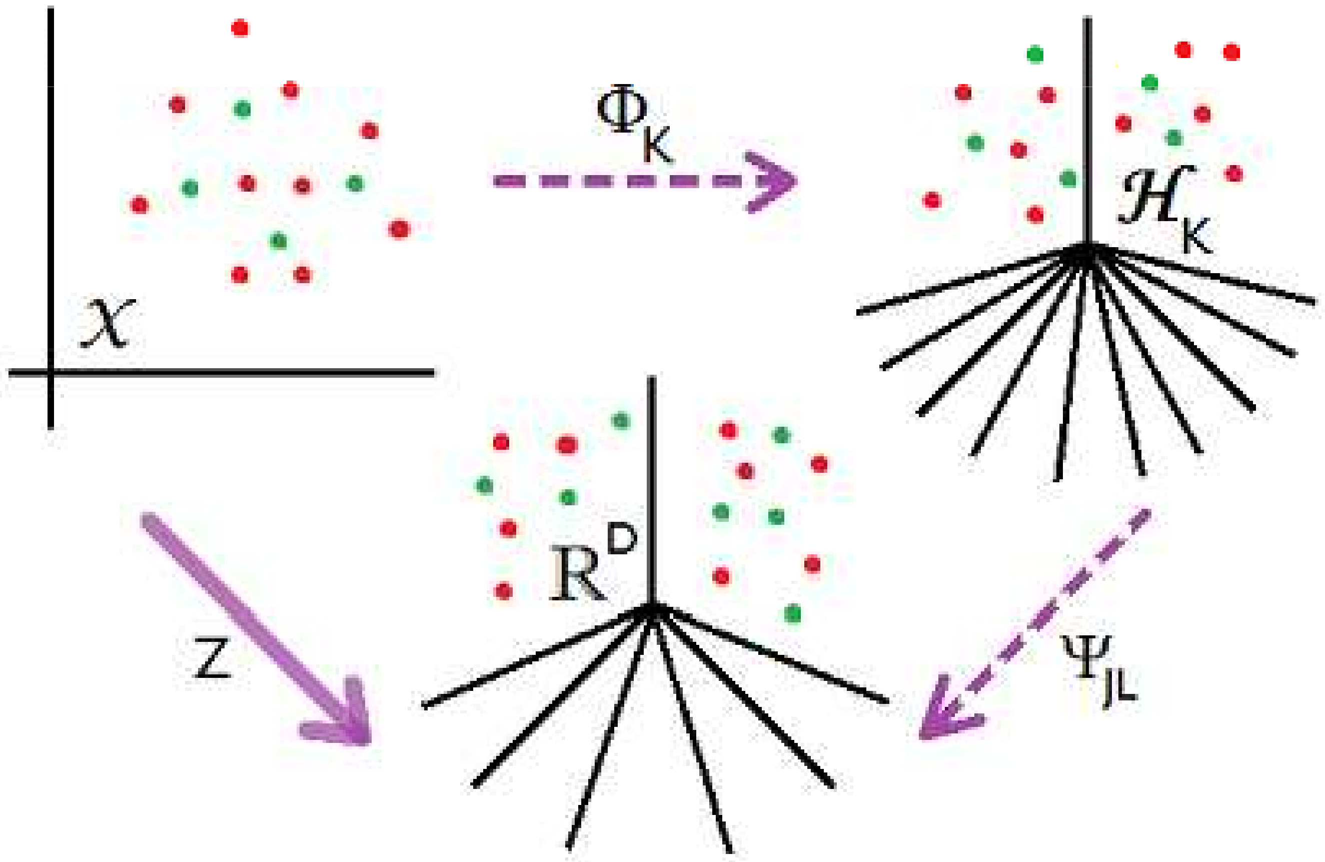
    $$K(x_1, x_2) \approx \langle Z(x_1), Z(x_2) \rangle$$

  - c.f. the Johnson-Lindenstrauss Lemma
- This allows for $O(1)$ hypothesis evaluation

  $$h(x) \approx \sum \alpha_i \langle Z(x), Z(x_i) \rangle f^*(x_i) = \langle Z(x), \sum \alpha_i f^*(x_i) Z(x_i) \rangle$$

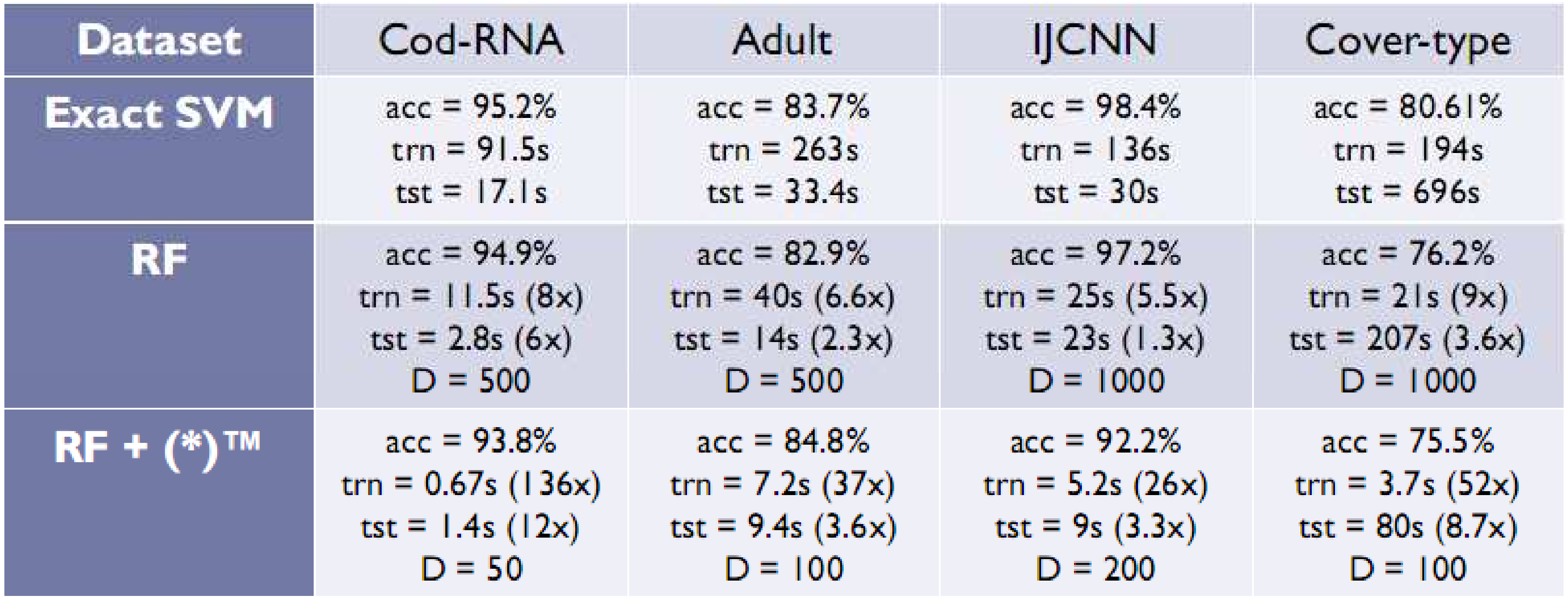
- Algorithmitizing these results requires more work
  - Require "Structure Theorems" to construct such feature maps
    - Translation Invariant [Rahimi-Recht NIPS07]
    - Homogeneous [Vedaldi-Zisserman CVPR10]
    - Dot Product [K.-Karnick AISTATS12]

Feature maps in pictures



Feature maps in action !

- Accelerated training and testing routines
  - Speed ups frequently in order(s) of magnitude
  - Comparable/increased frequencies



Dataset	Cod-RNA	Adult	IJCNN	Cover-type

Exact SVM	acc = 95.2%
	trn = 91.5s
	tst = 17.1s
		acc = 83.7%
		trn = 263s
		tst = 33.4s
			acc = 98.4%
			trn = 136s
			tst = 30s
				acc = 80.61%
				trn = 194s
				tst = 696s

RF	acc = 94.9%
	trn = 11.5s (8x)
	tst = 2.8s (6x)
	D = 500
		acc = 82.9%
		trn = 40s (6.6x)
		tst = 14s (2.3x)
		D = 500
			acc = 97.2%
			trn = 25s (5.5x)
			tst = 23s (1.3x)
			D = 1000
				acc = 76.2%
				trn = 21s (9x)
				tst = 207s (3.6x)
				D = 1000

RF + (*)™	acc = 93.8%
	trn = 0.67s (136x)
	tst = 1.4s (12x)
	D = 50
		acc = 84.8%
		trn = 7.2s (37x)
		tst = 9.4s (3.6x)
		D = 100
			acc = 92.2%
			trn = 5.2s (26x)
			tst = 9s (3.3x)
			D = 200
				acc = 75.5%
				trn = 3.7s (52x)
				tst = 80s (8.7x)
				D = 100

Some parting thoughts

- Do these results come for free ? No !
- The algorithmic formulations for learning with non-positive kernels usually lead to dense and slow hypothesis
  - Use diversity inducing heuristics [K.-Jain NIPS11] to speed things in case of classification - no theoretical guarantees
  - Much better scenario for regression where sparse learning techniques offer theoretical guarantees as well as empirical speedups [K.-Jain NIPS12]
- Most feature map constructions are random
  - Variance reduction is an issue
  - Since feature maps provide an approximation, they introduce additional bias
  - Slight dips in accuracy of the learned hypotheses