# Non-linear Models-II

CS771: Introduction to Machine Learning

Purushottam Kar

# Outline of today's discussion

- Recap Introduction to Kernels and Kernel SVMs
- Kernel Ridge Regression
- Kernel K-means
- Kernel k-NN … done two ways!
- Next lecture:
  - Kernel PCA
  - Accelerated learning with kernels
  - PML with kernels: Gaussian Processes

# Recap

# Kernels

- Used to learn non-linear models when linear models do badly
- Measures of similarity b/w two data points from a universe $\mathcal{X}$
$$K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
- Can be defined when $\mathcal{X}$ is a set of vectors, sets, strings, documents, graphs, trees, images, …
- Can be used to do classification, regression, clustering, dim-red using powerful non-linear models

# There are kernels and there are kernels



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

$$K(\mathbf{x}, \mathbf{y})$$



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$K(\mathbf{x}, \mathbf{y})$$



seriousseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$


OS Kernel

# There are kernels and there are kernels

Kernels of corn

$$\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)


OS Kernel

# There are kernels and there are kernels

Kernels of corn

$$\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)


OS Kernel



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$


OS Kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)

# There are kernels and there are kernels

Kernels of corn

$$\cancel{\{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)

Operating System

OS Kernel



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$\cancel{\{\mathbf{x} : A\mathbf{x} = 0\}}$$

~~Kernel of linear transformation~~

$$K(\mathbf{x}, \mathbf{y})$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)


OS Kernel



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# There are kernels and there are kernels

Kernels of corn

$$\cancel{\{\mathbf{x} : A\mathbf{x} = 0\}}$$

Kernel of linear transformation

$$K(\mathbf{x}, \mathbf{y})$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution kernels (masks)


OS Kernel



seriouseats.com, sharkhammershark.wordpress.com, wikipedia.com,

# The *nice* Mercer kernel

- Given any object set $\mathcal{X}$ (need not be vectors, may be set of images, video, strings, genome sequences), a similarity function
$$K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
is called a Mercer kernel if there exists a projection $\phi: \mathcal{X} \to \mathcal{H}$ such that for all $x, y \in \mathcal{X}$ we have $K(x, y) = \langle \phi(x), \phi(y) \rangle$
- $\phi$ often called feature map, embedding as well
- $\mathcal{H}$ can be $\mathbb{R}^D$ for some large/moderate $D$ or even inf. dim
- In general $\mathcal{H}$ has to be a *Hilbert space*, a real vector space (possibly infinite dimensional) which allows dot products
- $\mathcal{H}$ often called the Reproducing Kernel Hilbert Space (RKHS) for $K$

# Examples of Kernels

- Can define kernels whenever have a sound notion of similarity
- Sometimes they turn out to be nice (Mercer) as well!
- When $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ are vectors
  - Linear kernel $K_{\text{lin}}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$
  - Quadratic kernel $K_{\text{quad}}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2$
  - Polynomial kernel $K_{\text{poly}}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^p$, $c \geq 0, p \in \mathbb{N}$
  - Gaussian kernel $K_{\text{gauss}}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \cdot \|\mathbf{x} - \mathbf{y}\|_2^2)$
  - Laplacian kernel $K_{\text{lap}}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \cdot \|\mathbf{x} - \mathbf{y}\|_1)$

# Examples of Kernels contd …

- When $X, Y \subseteq \mathcal{U}$ are sets
  - Intersection kernel $K_{\text{int}}(X, Y) = |X \cap Y|$
  - Norm. Int. kernel $K_{\text{int-n}}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$ (notice that $K_{\text{int-n}} \in (0,1)$)
- When $\mathbf{x}, \mathbf{y}$ are strings/documents of words from a dictionary $\Sigma$
  - Let dictionary $\Sigma = \{w_1, w_2, \dots, w_d\}$ have $d$ words in it
  - Let $c_i(x)$ be the count of word $i$ in string $\mathbf{x}$
  - Intersection kernel $K_{\text{int}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} \min\{c_i(\mathbf{x}), c_i(\mathbf{y})\}$
  - Norm. Int. kernel $K_{\text{int-n}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} \frac{\min\{c_i(\mathbf{x}), c_i(\mathbf{y})\}}{\sqrt{c_i(\mathbf{x}) \cdot c_i(\mathbf{y})}}$ (define $\frac{0}{0} = 0$

# How to construct new Mercer kernels

- **Method 1**: operations on existing kernels
- If $K_1, K_2$ are existing Mercer kernels then
  - $K_3 = c_1 \cdot K_1 + c_2 \cdot K_2$ is also a Mercer kernel if $c_1, c_2 \geq 0$
$$K_3(\mathbf{x}, \mathbf{y}) = c_1 \cdot K_1(\mathbf{x}, \mathbf{y}) + c_2 \cdot K_2(\mathbf{x}, \mathbf{y})$$
  - $K_4 = K_1 \cdot K_2$ is also a nice kernel
$$K_4(\mathbf{x}, \mathbf{y}) = K_1(\mathbf{x}, \mathbf{y}) \cdot K_2(\mathbf{x}, \mathbf{y})$$
- **Method 2**: find a new feature construction for the data
- Find a way to represent data $x$ as a vector $\phi_{\text{new}}(x) \in \mathbb{R}^d$
$$K_{\text{new}}(x, y) = \langle \phi_{\text{new}}(x), \phi_{\text{new}}(y) \rangle$$
- **Method 3**: mix and match
- Take new data representation $\phi_{\text{new}}(x) \in \mathbb{R}^d$ and an old kernel $K_{\text{old}}$
$$K_{\text{newer}}(x, y) = K_{\text{old}}\left(\phi_{\text{new}}(x), \phi_{\text{new}}(y)\right)$$

# Exercises

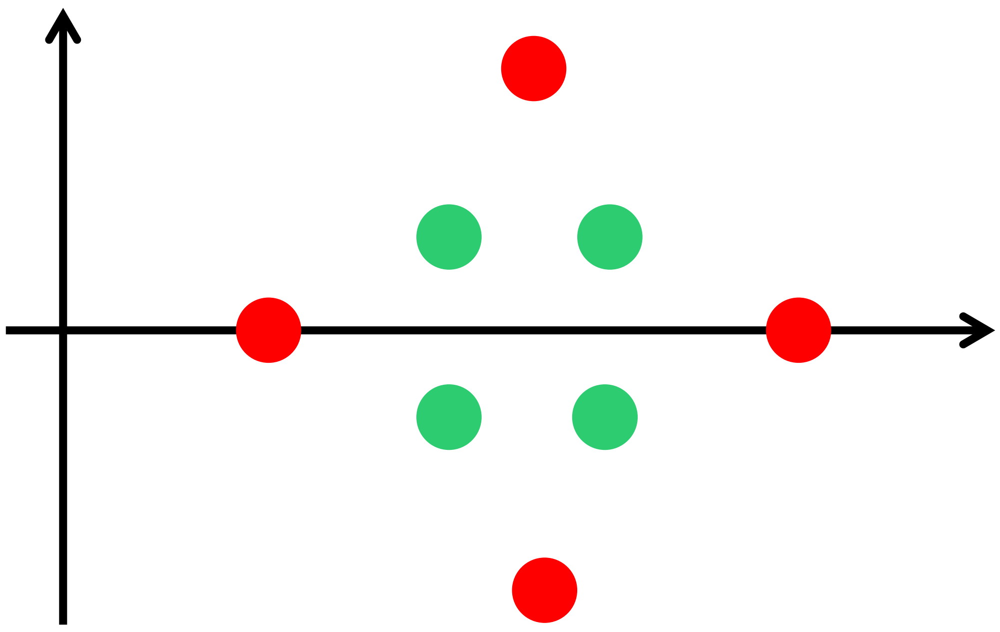
- Show that the kernels on vectors (poly, Gaussian) are indeed measures of similarity (assume all vectors are unit $L_2$ norm)
- Show that the Intersection kernel on sets is Mercer
- Show that the normalized Intersection kernel on sets is Mercer
- Show that if $K_1, K_2$ are Mercer kernels and then so are
  - $K_3 = c_1 \cdot K_1 + c_2 \cdot K_2$
  - $K_4 = K_1 \cdot K_2$
- Show that if $K_5$ is Mercer and $x_1, x_2, \ldots, x_n \in \mathcal{X}$ are data points and $G \in \mathbb{R}^{n \times n}$ is a "Gram matrix" such that $G_{ij} = K_5(x_i, x_j)$, then $G$ is PSD i.e. for all vectors $\mathbf{c} \in \mathbb{R}^n$, we have $\mathbf{c}^\top G\mathbf{c} \geq 0$
- For above questions, assume $K_1, K_2, K_5$ have feature maps $\phi_i: \mathcal{X} \to \mathbb{R}^{d_i}$ for $i = 1, 2, 5$ that map to finite dim. spaces

# How do kernels help learn non-linear models?

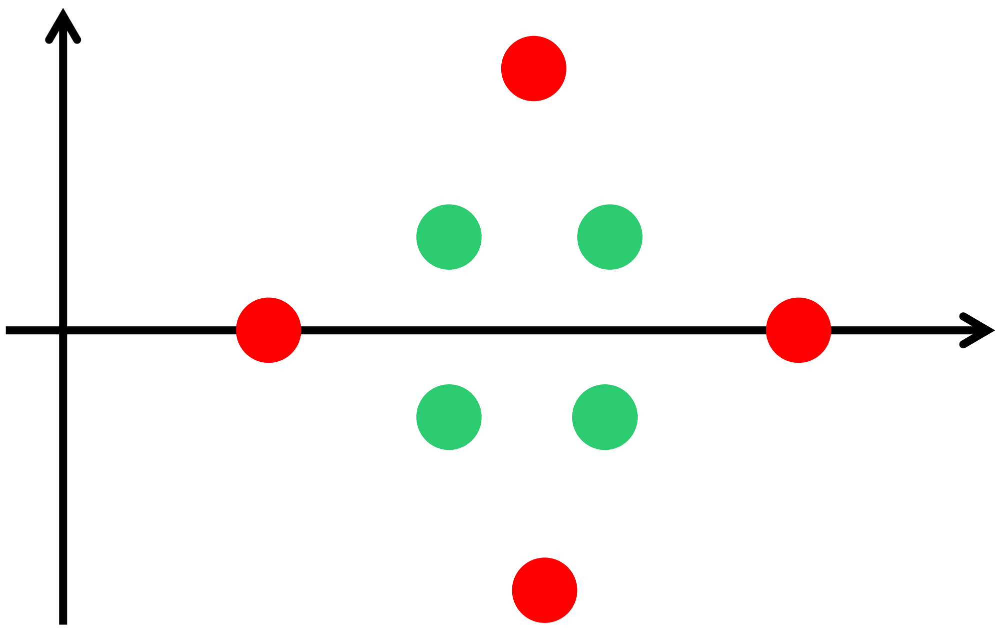
- Consider the quadratic kernel $K_{\text{quad}}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$
- The feature map for $K_{\text{quad}}$ is $\phi(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \phi_2(\mathbf{x})] \in \mathbb{R}^{d^2+d+1}$
- A linear function over $\phi(\mathbf{x})$ is $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle$ for some $\mathbf{w} \in \mathbb{R}^{d^2+d+1}$ corresponds to a quadratic function over $\mathbf{x} \in \mathbb{R}^d$
- Can utilize a linear learning algorithm in $\mathbb{R}^{d^2+d+1}$ to learn a quadratic function in $\mathbb{R}^d$
- If we learn the best linear model in $\mathbb{R}^{d^2+d+1}$, we would automatically learn the best quadratic model in $\mathbb{R}^d$
- We already know algorithms to learn the best linear model
- Just need to avoid explicitly mapping data to $\mathbb{R}^{d^2+d+1}$

# A toy example: non-linear classification

# A toy example: non-linear classification

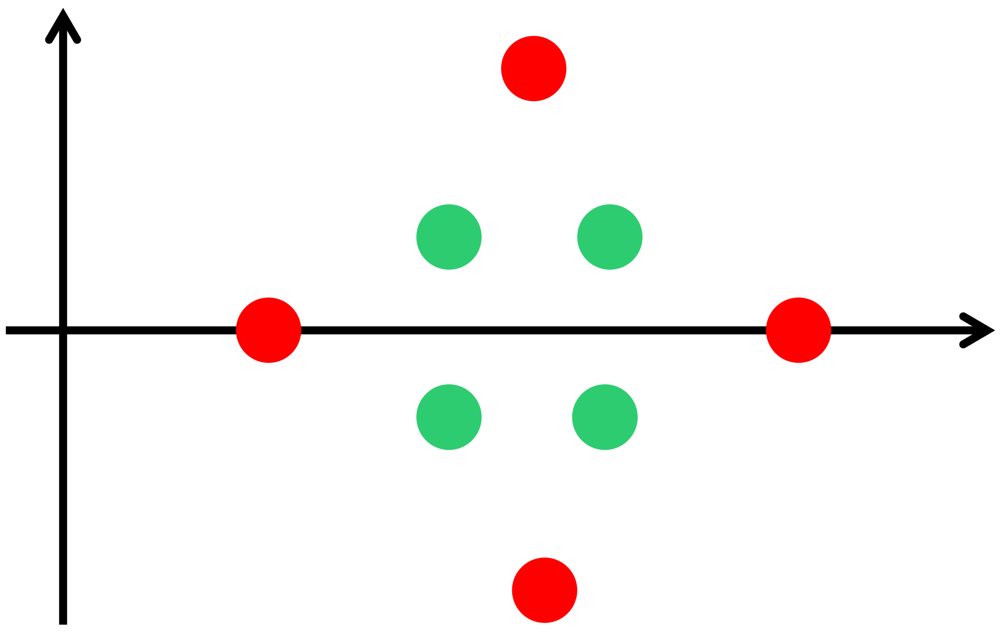


# A toy example: non-linear classification



$$\mathbb{R}^2 \ni (x, y) \mapsto \phi(x, y) = [x, x^2, y^2] \in \mathbb{R}^3$$

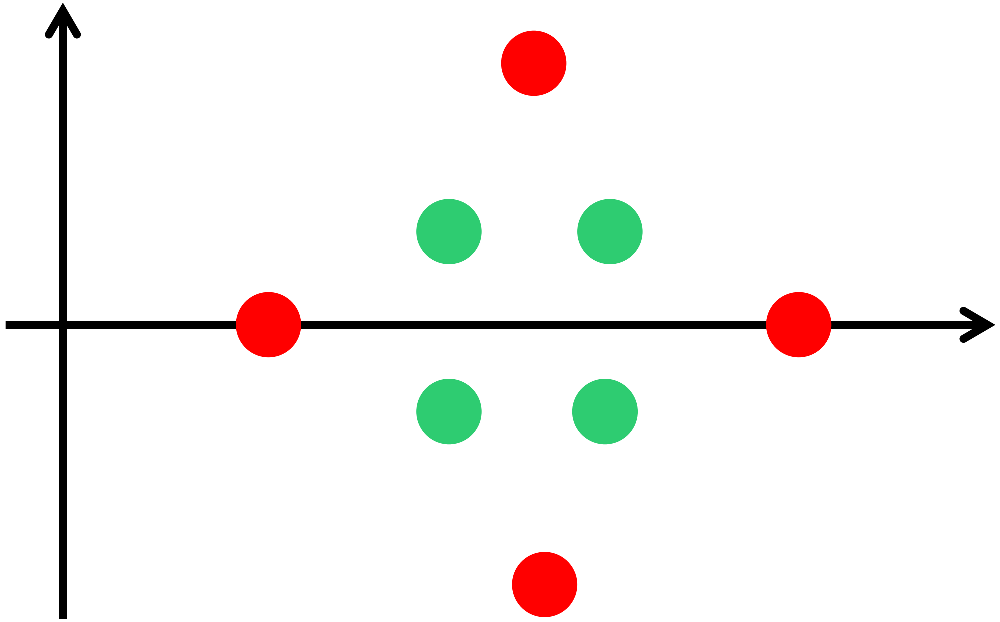
# A toy example: non-linear classification



A linear function on this 3-D vector looks like
$\langle \mathbf{w}, \phi(x,y) \rangle$
$\mathbf{w}_1 \cdot x + \mathbf{w}_2 \cdot x^2 + \mathbf{w}_3 \cdot y^2$

$$\mathbb{R}^2 \ni (x,y) \mapsto \phi(x,y) = [x, x^2, y^2] \in \mathbb{R}^3$$

# A toy example: non-l … on



Hmm … so a linear function in the 3D space can learn the decision boundary

$$-2x + x^2 + y^2$$
$$= (x-1)^2 + y^2 - 1$$

A linear function on this 3-D vector looks like

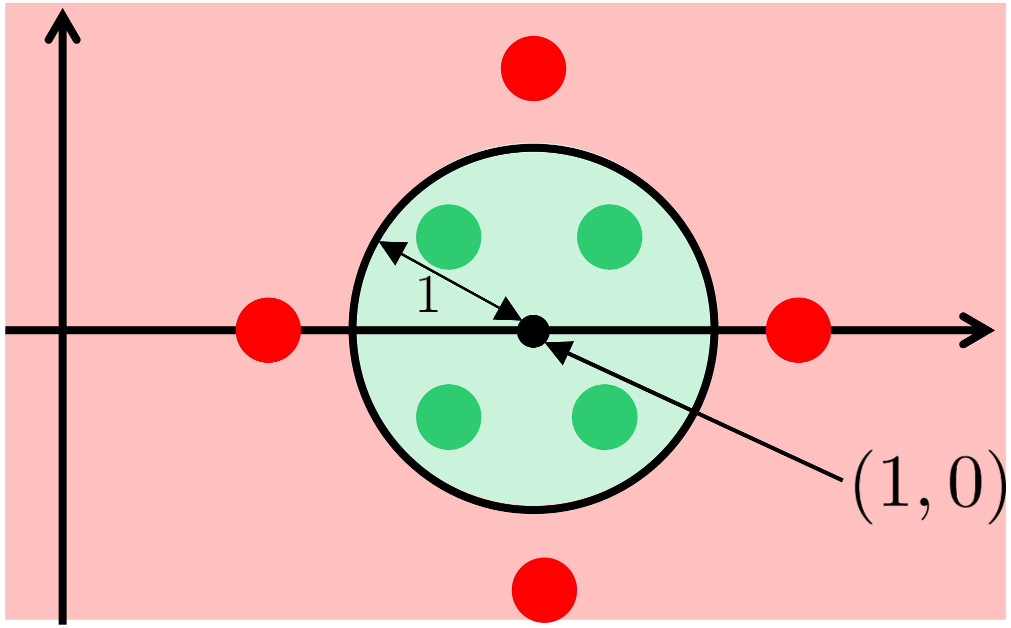
$$\langle \mathbf{w}, \phi(x,y) \rangle$$
$$\mathbf{w}_1 \cdot x + \mathbf{w}_2 \cdot x^2 + \mathbf{w}_3 \cdot y^2$$

$$\mathbb{R}^2 \ni (x,y) \mapsto \phi(x,y) = [x, x^2, y^2] \in \mathbb{R}^3$$

# A toy example: non-l[...]on

Hmm … so a linear function in the 3D space can learn the decision boundary

$$-2x + x^2 + y^2$$
$$= (x-1)^2 + y^2 - 1$$




A linear function on this 3-D vector looks like

$$\langle \mathbf{w}, \phi(x, y) \rangle$$
$$\mathbf{w}_1 \cdot x + \mathbf{w}_2 \cdot x^2 + \mathbf{w}_3 \cdot y^2$$

$$\mathbb{R}^2 \ni (x, y) \mapsto \phi(x, y) = [x, x^2, y^2] \in \mathbb{R}^3$$

# Kernel SVMs

# Executing an SVM in original space $\mathbb{R}^d$

**Primal Formulation**

- $\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\cdot\|\mathbf{w}\|_2^2$
  s.t. $1 - y^i\langle\mathbf{w},\mathbf{x}^i\rangle \leq 0$
- Have seen GD/SGD methods to solve this ☺

**Dual Formulation**

- $\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^n \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i,j}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y^i y^j\langle\mathbf{x}^i,\mathbf{x}^j\rangle$
  s.t. $\alpha_i \geq 0$
- Have seen SCD methods to solve this ☺
- Can recover $\mathbf{w} = \sum_{i=1}^n \boldsymbol{\alpha}_i y^i \boldsymbol{x}^i$

But my problem does horribly with linear classifiers. I really need non-linearity

Hmm … okay. Choose a nice feature map/projection $\phi(\cdot)\colon \mathbb{R}^d \to \mathcal{H}$ and solve the SVM in the space of $\mathcal{H}$

# Executing an SVM in RKHS $\mathcal{H}$

## Primal Formulation

- $\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2} \cdot \|\mathbf{w}\|_2^2$
  s.t. $1 - y^i \langle \mathbf{w}, \phi(\mathbf{x}^i) \rangle \leq 0$
- But I chose $\mathcal{H}$ to be really large dimensional
- A single step of GD/SGD will take enormous time to run ☹

# Executing an SVM in RKHS $\mathcal{H}$

## Primal Formulation

- $\min_{\mathbf{w}\in\mathcal{H}} \frac{1}{2}\cdot\|\mathbf{w}\|_2^2$
  s.t. $1 - y^i\langle\mathbf{w}, \phi(\mathbf{x}^i)\rangle \leq 0$
- But I chose $\mathcal{H}$ to be really large dimensional
- A single step of GD/SGD will take enormous time to run ☹

## Dual Formulation

- $\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^n \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i,j}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y^i y^j \langle\phi(\mathbf{x}^i), \phi(\mathbf{x}^j)\rangle$
  s.t. $\alpha_i \geq 0$
- Wow … this still has only $n$ variables
- But what about $\langle\phi(\mathbf{x}^i), \phi(\mathbf{x}^j)\rangle$?
- If only I had a way to compute this fast 😐
- Kernels !!!
- Choose a kernel $K$ and take $\phi = \phi_K$ to be its feature map
- We will get $\langle\phi(\mathbf{x}^i), \phi(\mathbf{x}^j)\rangle = K(\boldsymbol{x}^i, \boldsymbol{x}^j)$

# Executing an SVM in RKHS $\mathcal{H}$

## Primal Formulation

- $\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2} \cdot \|\mathbf{w}\|_2^2$
  s.t. $1 - y^i \langle \mathbf{w}, \phi(\mathbf{x}^i) \rangle \leq 0$
- But I chose $\mathcal{H}$ to be really large dimensional
- A single step of GD/SGD will take enormous time to run ☹
- Can now choose $\mathcal{H}$ to be infinite-dim as well ☺
- How do I recover $\mathbf{w}$ from $\boldsymbol{\alpha}$? Uh oh …

## Dual Formulation

- $\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^n \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i,j} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y^i y^j \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle$
  s.t. $\alpha_i \geq 0$
- Wow … this still has only $n$ variables
- But what about $\langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle$?
- If only I had a way to compute this fast 😐
- Kernels !!!
- Choose a kernel $K$ and take $\phi = \phi_K$ to be its feature map
- We will get $\langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle = K(\boldsymbol{x}^i, \boldsymbol{x}^j)$

# Using an SVM learnt in RKHS $\mathcal{H}$

- **Exercise**: take the Gaussian kernel and show how a single step of SCD on the dual can be executed in $\mathcal{O}(nd)$ time
- Note: $d$ is not the dimensionality of $\mathcal{H}$ which may be infinite dim
- So we can argue that SCD is really still taking $\mathcal{O}(n)$ time ☺
- But once we get a dual solution $\boldsymbol{\alpha}$ how do we get a classifier?
- Getting $\mathbf{w} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \phi(\boldsymbol{x}^i)$ pointless since $\mathbf{w}$ is infinite/large dim.
- Instead, exploit the fact the we will only need to calculate
  $$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$
  to classify a test point $\mathbf{x}$
- $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \left\langle \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \phi(\boldsymbol{x}^i), \phi(\mathbf{x}) \right\rangle = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \left\langle \phi(\boldsymbol{x}^i), \phi(\mathbf{x}) \right\rangle$
  $= \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i K(\mathbf{x}^i, \mathbf{x})$

# Using an SVM learnt in RKHS $\mathcal{H}$

Outliers, redundant points should have $\boldsymbol{\alpha}_i \approx 0$

$\boldsymbol{\alpha}_i$ indicates how useful the data point $(\mathbf{x}^i, y^i)$ is while voting

If $\mathbf{x}^i$ "similar" to $\mathbf{x}$ i.e. $K(\mathbf{x}^i, \mathbf{x})$ is large, $y^i$ influences predicted label more – nice!!

- Note: $d$ is not the dime…
- So we can argue that SC… is really still taking…
- But once we get a c… we get a classifier?
- Getting $\mathbf{w} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \phi(\ldots$ …ointless since $\mathbf{w}$ is infinite/large dim.
- Instead, exploit the f… the we w… late

  $\langle \mathbf{w}, \ldots$

  to classify a test point $\mathbf{x}$

Note that this looks like soft weighted k-NN

Need to perform $\tilde{n}$ kernel evaluations to predict on a test point

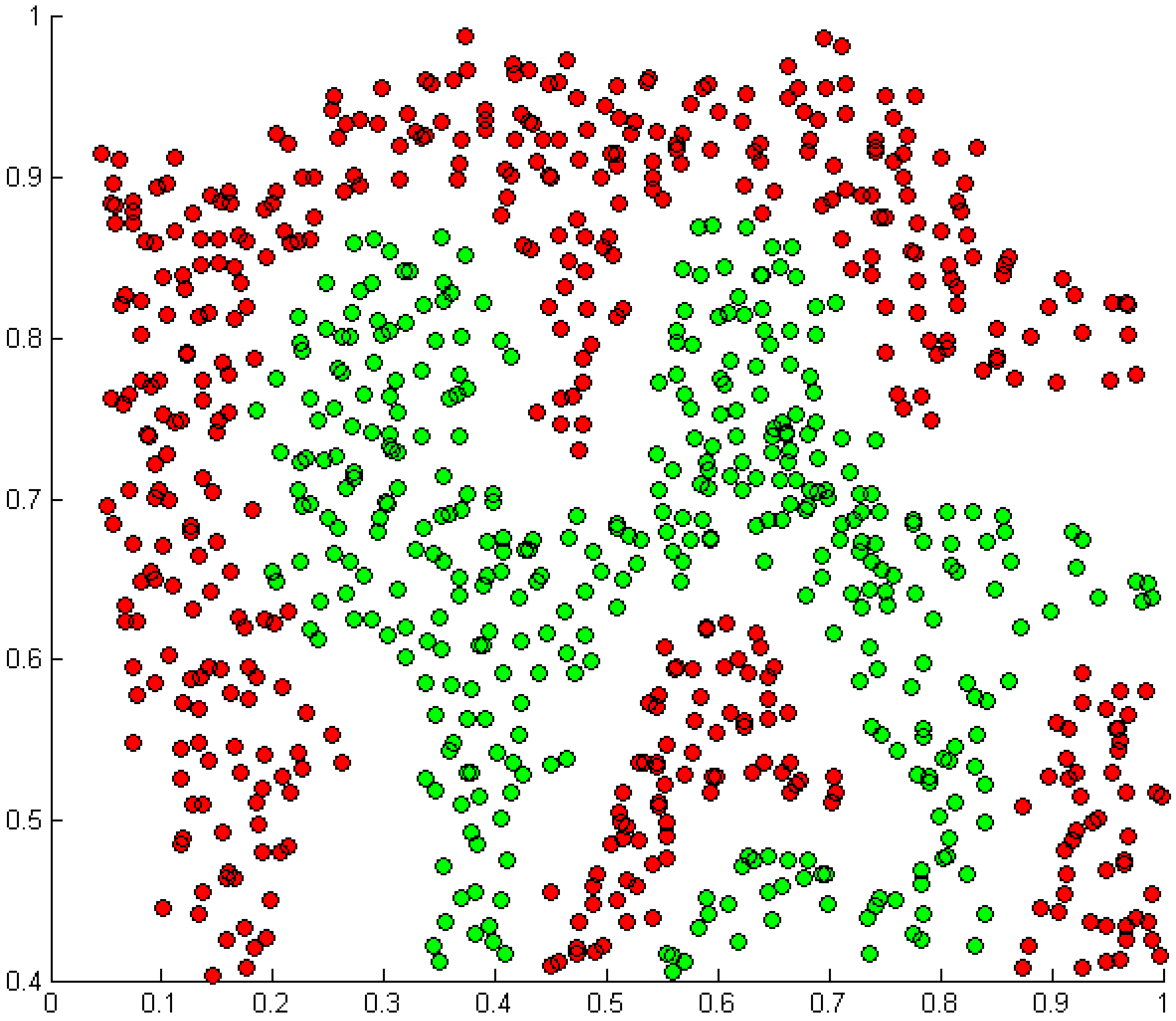
- $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \langle \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \phi(\mathbf{x}^i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \langle \phi(\ldots$

  $= \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i K(\mathbf{x}^i, \mathbf{x})$

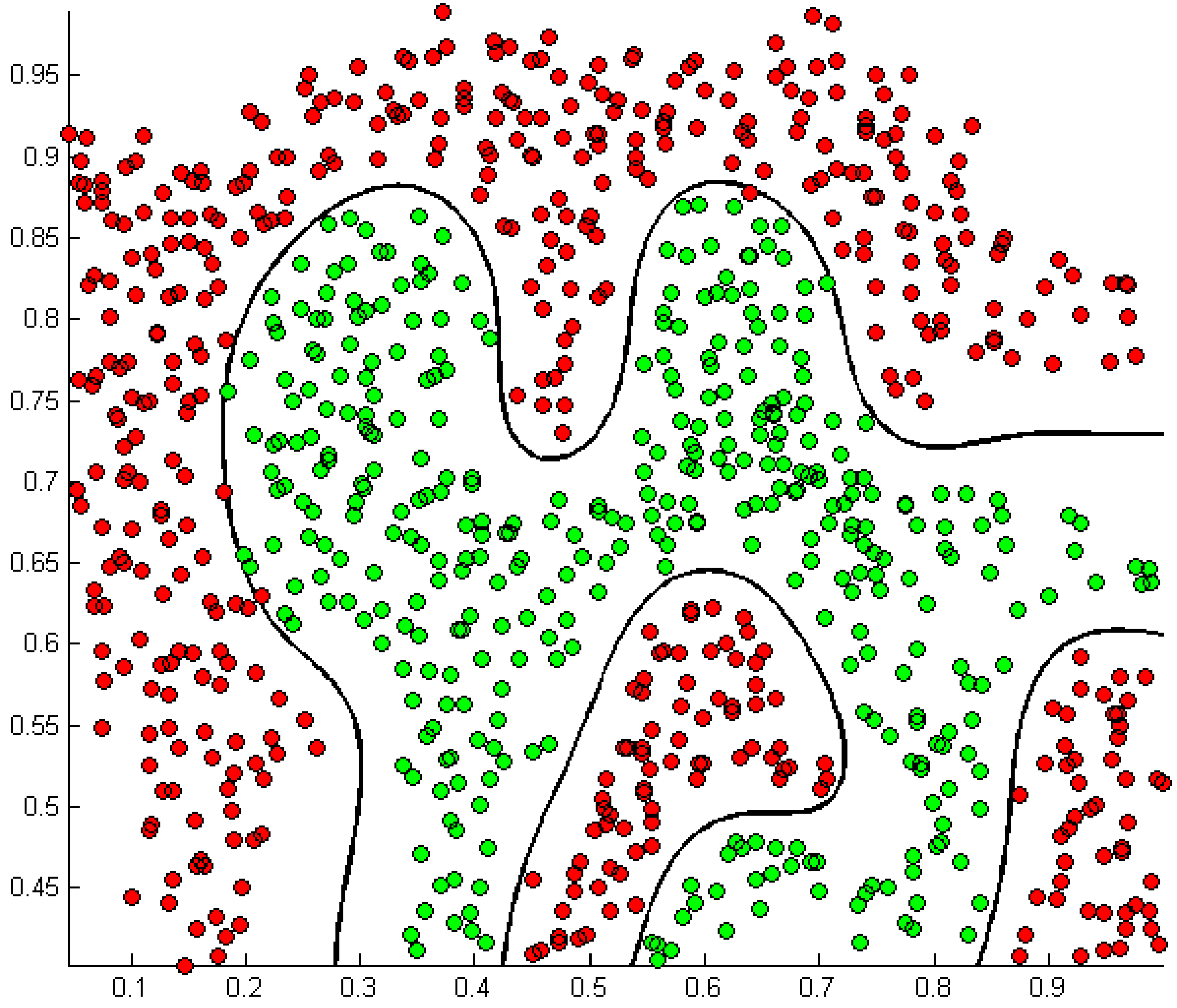
$\tilde{n}$ is the number of support vectors which have $\boldsymbol{\alpha}_i \neq 0$

# Example

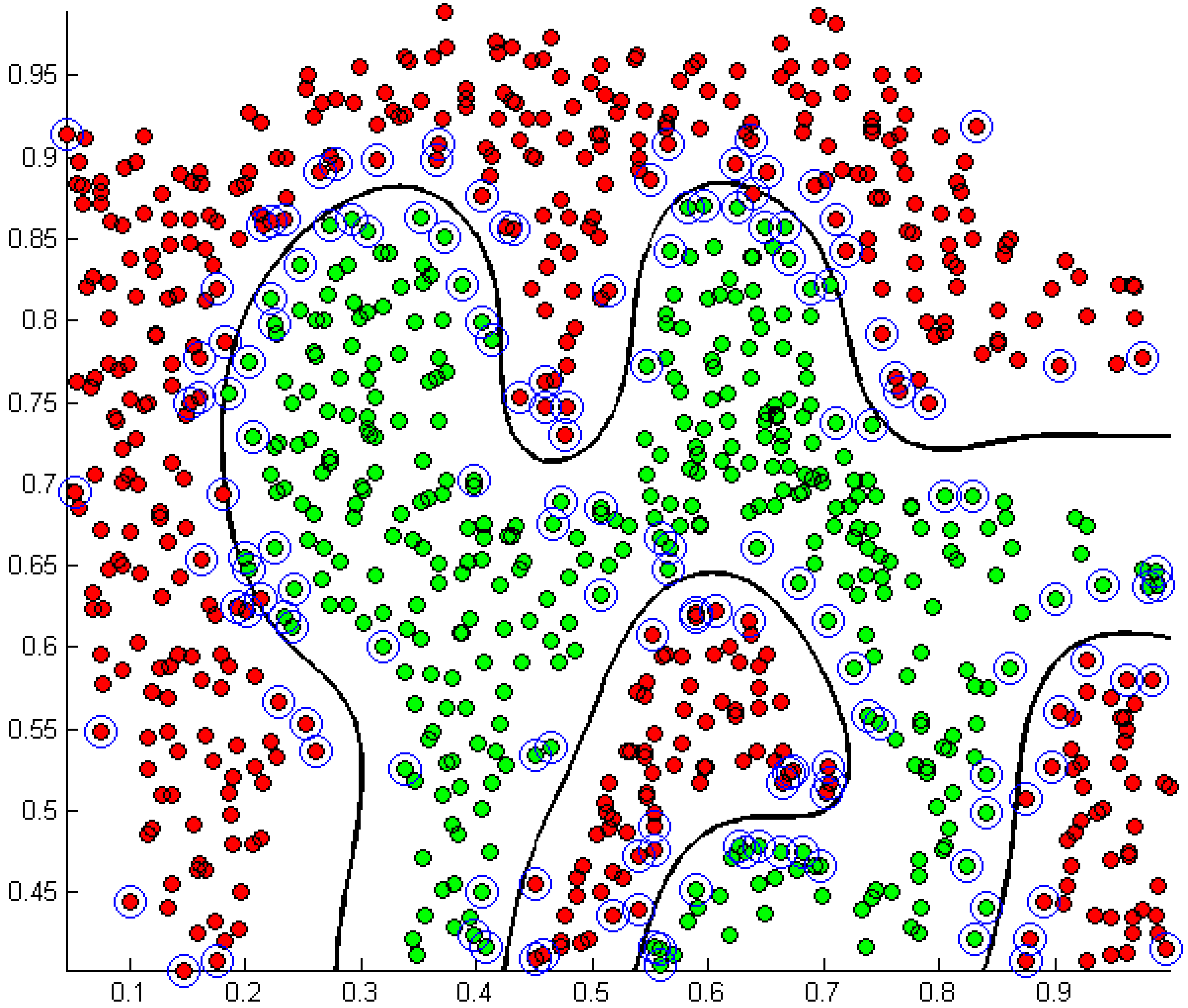
# Example



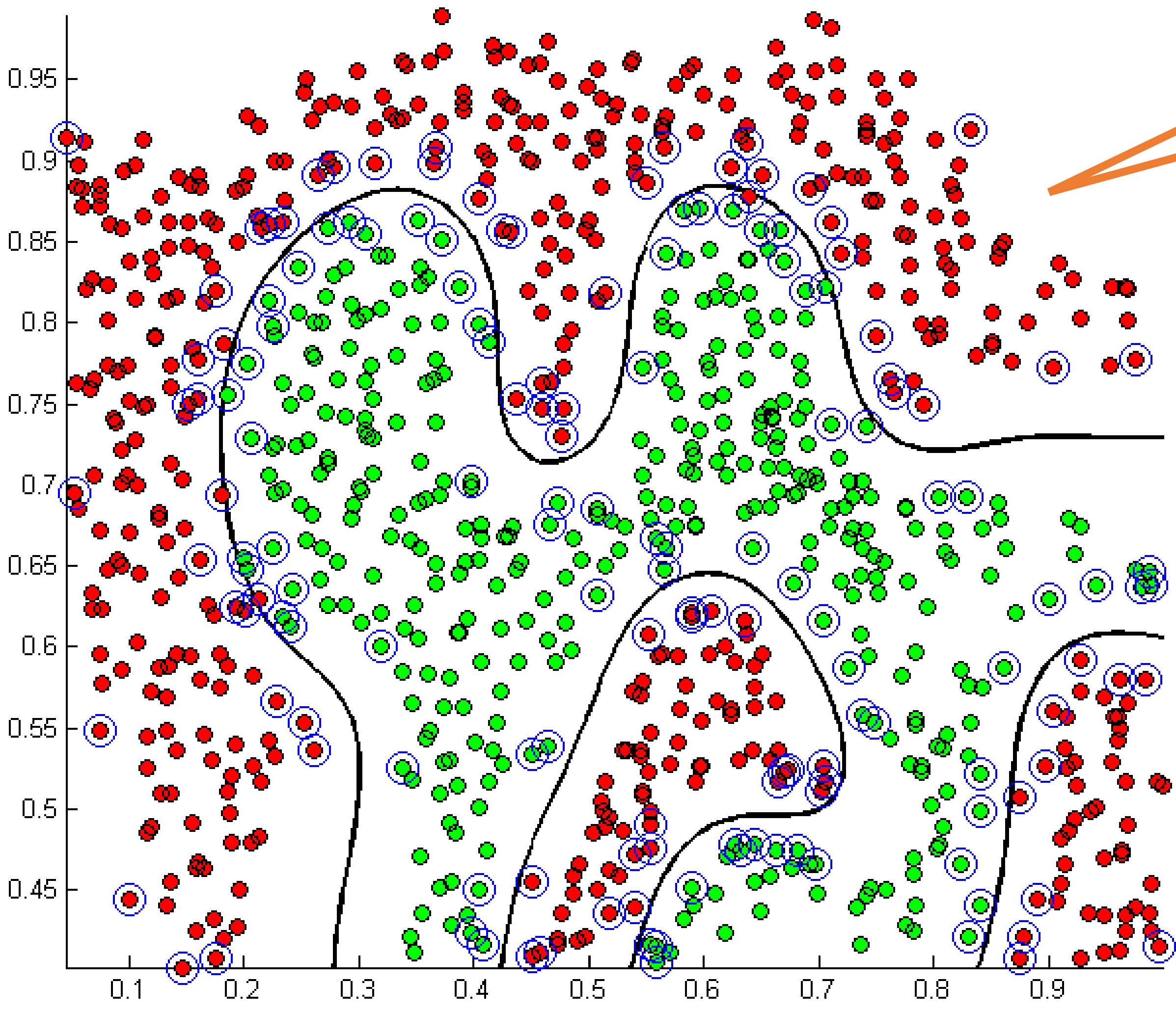
# Example



# Example

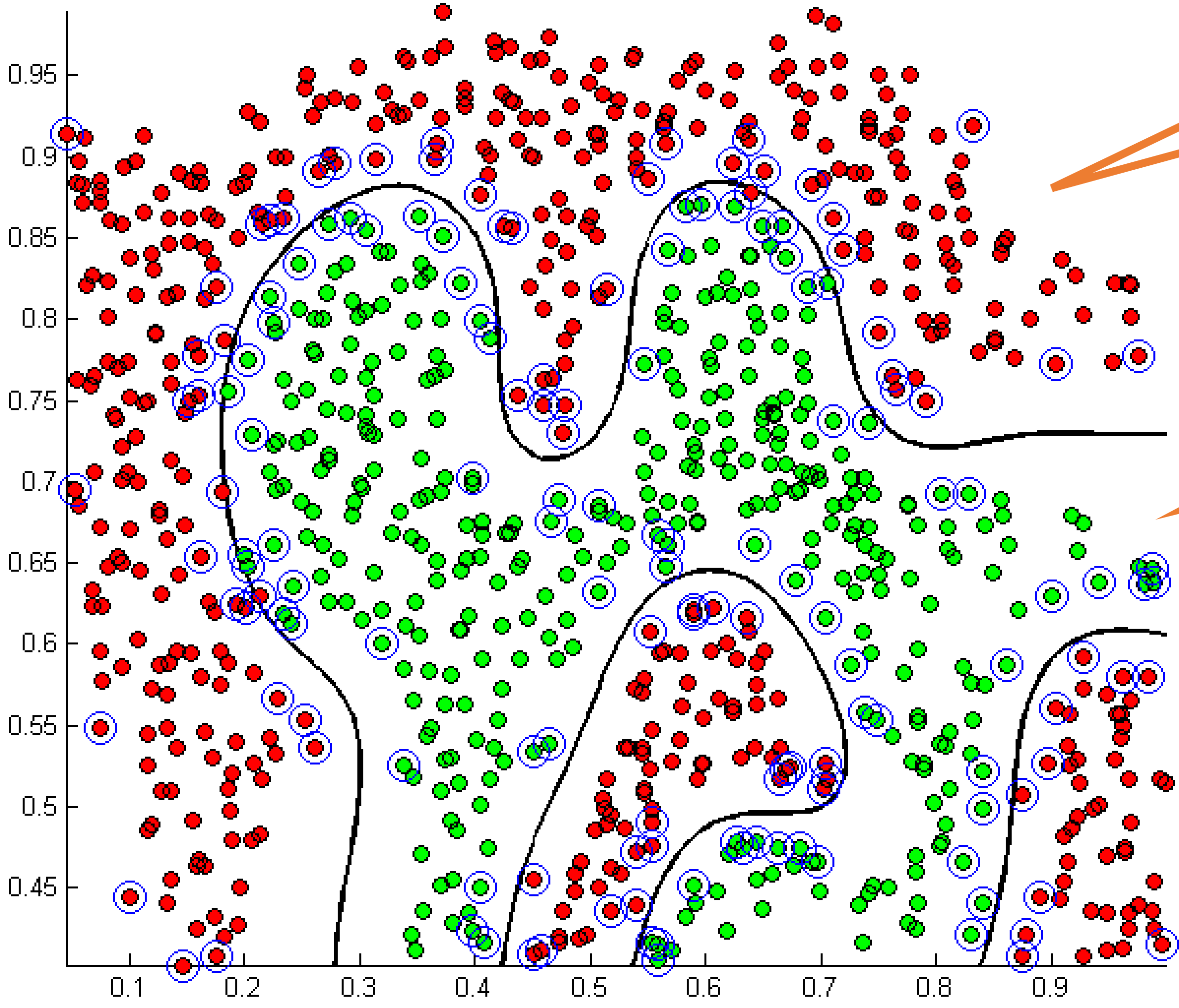


# Example




Gaussian kernel with $\gamma = 100$, 850 points, 150 SVs

# Example




Gaussian kernel with $\gamma = 100$, 850 points, 150 SVs

Most SVs close to the decision boundary

# Kernel Ridge Regression

# Ridge Regression in original space $\mathbb{R}^d$

- Data given to us is $\{\mathbf{x}^i, y^i\}_{i=1,2,\ldots,n}$ where $x^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$
- Let $X = [x^1, x^2, \ldots, x^n] \in \mathbb{R}^{d \times n}$ and $\mathbf{y} = [y^1, y^2, \ldots, y^n]^\top \in \mathbb{R}^n$
- $\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$
- Closed form solution

$$\hat{\mathbf{w}} = (XX^\top + \lambda \cdot I_d)^{-1} X\mathbf{y}$$

- Can find it by inverting a $d \times d$ matrix or by using S/GD

# Ridge Regression in RKHS $\mathcal{H}$

- Data given to us is $\{\phi(\mathbf{x}^i), y^i\}_{i=1,2,\ldots,n}$ where $\phi(\mathbf{x}^i) \in \mathcal{H}$ and $y^i \in \mathbb{R}$
- Since $\mathcal{H}$ can be infinite (or at least high) dimensional, and we have $\mathbf{w} \in \mathcal{H}$, the closed form solution is useless ☹
- Hmm … if only there were a dual solution for Ridge Regression 😐
- But no constraints in $\min\limits_{\mathbf{w}\in\mathcal{H}} \sum_{i=1}^n \left(y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i)\rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$
- Let us introduce some ☺
- Rewrite the RR problem in $\mathcal{H}$ as

$$\min_{\substack{\mathbf{w}\in\mathcal{H}\\ \mathbf{r}\in\mathbb{R}^n}} \quad \lambda \cdot \|\mathbf{w}\|_{\mathcal{H}}^2 + \|\mathbf{r}\|_2^2$$

$$\text{s.t.} \quad \mathbf{r}_i = y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i)\rangle$$

# Ridge Regression in RKHS $\mathcal{H}$

- Data given to us is $\{\phi(\mathbf{x}^i), y^i\}_{i=1,2,\ldots,n}$ where $\phi(\mathbf{x}^i) \in \mathcal{H}$ and $y^i \in \mathbb{R}$
- Since $\mathcal{H}$ can be infinite (or at least high) dimensional, and we have $\mathbf{w} \in \mathcal{H}$, the closed form solution is useless ☹
- Hmm … if only there were a dual solution for
- But no constraints in $\min_{\mathbf{w} \in \mathcal{H}} \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i) \rangle\right)$
- Let us introduce some ☺
- Rewrite the RR problem in $\mathcal{H}$ as

$$\min_{\substack{\mathbf{w} \in \mathcal{H} \\ \mathbf{r} \in \mathbb{R}^n}} \quad \lambda \cdot \|\mathbf{w}\|_{\mathcal{H}}^2 + \|\mathbf{r}\|_2^2$$

$$\text{s.t.} \qquad \mathbf{r}_i = y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i) \rangle$$

Can think of $\|\mathbf{w}\|_{\mathcal{H}}^2$ as $\|\mathbf{w}\|_2^2$ for now

The constraint is a dummy one but very useful in deriving dual

# Ridge Regression in RKHS $\mathcal{H}$

$$\min_{\substack{\mathbf{w}\in\mathcal{H}\\ \mathbf{r}\in\mathbb{R}^n}} \quad \lambda\cdot\|\mathbf{w}\|_{\mathcal{H}}^2 + \|\mathbf{r}\|_2^2$$

$$\text{s.t.} \qquad \mathbf{r}_i = y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i)\rangle$$

Exercise!
Assume $\mathcal{H} = \mathbb{R}^D$ and $\|\mathbf{w}\|_{\mathcal{H}}^2 = \|\mathbf{w}\|_2^2$

- The Lagrangian becomes

$$\mathcal{L}(\mathbf{w},\mathbf{r},\boldsymbol{\alpha}) = \lambda\cdot\|\mathbf{w}\|_{\mathcal{H}}^2 + \|\mathbf{r}\|_2^2 + \sum_{i=1}^n \boldsymbol{\alpha}_i\left(y^i - \langle \mathbf{w}, \phi(\mathbf{x}^i)\rangle - \mathbf{r}_i\right)$$

- The dual (after simplification) becomes

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{4\lambda}\cdot\sum_{i,j}^n \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j\langle\phi(\mathbf{x}^i),\phi(\mathbf{x}^j)\rangle + \frac{1}{4}\cdot\|\boldsymbol{\alpha}\|_2^2 - \boldsymbol{\alpha}^\top\mathbf{y}$$

- … with a beautiful way to recover the primal from the dual

$$\mathbf{w} = \frac{1}{2\lambda}\sum_i^n \boldsymbol{\alpha}_i\cdot\phi(\mathbf{x}^i) \text{ and } \mathbf{r} = \frac{\boldsymbol{\alpha}}{2}$$

# Ridge Regression in RKHS $\mathcal{H}$

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{4\lambda} \cdot \sum_{i,j}^{n} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle + \frac{1}{4} \cdot \|\boldsymbol{\alpha}\|_2^2 - \boldsymbol{\alpha}^\top \mathbf{y}$$

- Let $G \in \mathbb{R}^{n\times n}$ be the Gram matrix such that
$$G_{ij} = K(\mathbf{x}^i, \mathbf{x}^j) = \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle$$
- Can rewrite the above as
$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \boldsymbol{\alpha}^\top (G + \lambda \cdot I_n) \boldsymbol{\alpha} - 4\lambda \cdot \boldsymbol{\alpha}^\top \mathbf{y}$$
- Using first order optimality we get
$$\boldsymbol{\alpha} = 2\lambda \cdot (G + \lambda \cdot I_n)^{-1} \mathbf{y}$$
- So the dual requires inverting an $n \times n$ matrix
- Yet again closed form solution but no agony of high dimensions ☺

# Using Regressor learnt in RKHS $\mathcal{H}$

- We can rewrite the solution as
$$\boldsymbol{\beta} = (G + \lambda \cdot I_n)^{-1}\mathbf{y}$$
- Note that we now have $\mathbf{w} = \sum_i^n \boldsymbol{\beta}_i \cdot \phi(\mathbf{x}^i)$
- Cannot store this or use this to predict directly
- Use kernel trick again to predict on a test point $\mathbf{x}$
$$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \left\langle \sum_{i=1}^{n} \boldsymbol{\beta}_i \phi(\boldsymbol{x}^i), \phi(\mathbf{x}) \right\rangle = \sum_{i=1}^{n} \boldsymbol{\beta}_i \langle \phi(\boldsymbol{x}^i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} \boldsymbol{\beta}_i K(\mathbf{x}^i, \mathbf{x})$$
- $\mathcal{O}(n^3)$ time to train, $\mathcal{O}(nd)$ storage, $\mathcal{O}(nd)$ time for prediction
- Costlier than linear ridge regression but benefit of non-linearity

# A few Thoughts

- Note: dual holds even if no kernel is used/linear kernel used
- RR using the primal solution required inverting a $d \times d$ matrix
- RR using the dual solution required inverting an $n \times n$ matrix
- Aha … so we can use the dual solution to solve even the linear RR problem more cheaply if $d > n$
- Linear RR is just kernel RR with the linear kernel
- Advantage in linear RR is that we can easily use $\mathbf{w} = \sum_i^n \boldsymbol{\beta}_i \cdot \phi(\mathbf{x}^i)$
- **Challenge**: can you show how logistic regression can be done in an RKHS corresponding to a kernel $K$?
  Hint: find the dual problem for Logistic Regression

# Kernel K-means

# Hard Assignment K-means in $\mathbb{R}^d$

## K-MEANS/LLOYD'S ALGORITHM

1. Initialize means $\{\boldsymbol{\mu}^{k,0}\}_{k=1,\ldots K} \in \mathbb{R}^d$
2. For $i \in [n]$, update $z^{i,t}$ using $\boldsymbol{\mu}^{k,t}$
    1. Let $z^{i,t} = \arg\min_k \left\|\mathbf{x}^i - \boldsymbol{\mu}^{k,t}\right\|_2^2$
3. Update $\boldsymbol{\mu}^{k,t+1} = \frac{1}{n_k^t} \sum_{i:z^{i,t}=k} \mathbf{x}^i$, where $n_k^t = |\{i: z^{i,t} = k\}|$
4. Repeat until convergence

- Recall that this corresponds to learning a mixture of $K$ Gaussians with unknown means and identity (i.e. known) covariances
- The "soft" version of this algorithm is nothing but the EM algo ☺

# Hard Assignment K-means in RKHS $\mathcal{H}$

- Need to (re)define the two operations in k-means
  - Assign each data point to the closest cluster center
  - Recalculate the cluster centers
- Cluster centers are infinite dimensional ☹
- How do we define distances in an infinite dimensional RKHS?

# Calculating Distances implicitly in RKHS $\mathcal{H}$

- Calculating distances directly as $\|\mathbf{x}-\mathbf{y}\|_2^2$ will not work in $\mathcal{H}$
- Why? Since the vectors may be infinite dimensional
- Need to rewrite in terms of dot/inner products
- Let us rewrite

$$\|\mathbf{x}-\mathbf{y}\|_2^2 = \|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - 2\cdot\langle\mathbf{x},\mathbf{y}\rangle$$
$$= \langle\mathbf{x},\mathbf{x}\rangle + \langle\mathbf{y},\mathbf{y}\rangle - 2\cdot\langle\mathbf{x},\mathbf{y}\rangle$$

- We define the following distance function over the RKHS $\mathcal{H}$

$$\|\phi(x)-\phi(y)\|_{\mathcal{H}}^2 = \langle\phi(x),\phi(x)\rangle + \langle\phi(y),\phi(y)\rangle - 2\cdot\langle\phi(x),\phi(y)\rangle$$
$$= K(x,x) + K(y,y) - 2\cdot K(x,y)$$

- Corresponds to notion of Euclidean distance in Hilbert spaces

# Representing Cluster Centers implicitly

- Note that k-means always maintains cluster centers of the form

$$\boldsymbol{\mu}^{k,t+1} = \frac{1}{n_k^t} \sum_{i:z^{i,t}=k} \mathbf{x}^i$$

- This means cluster centers can always be written as linear combinations of the actual data points ☺
- Nice, so I can write

$$\boldsymbol{\mu}^{k,t+1} = \sum_{i=1}^{n} \alpha_i^{k,t} \mathbf{x}^i$$

- … where

$$\alpha_i^{k,t} = \begin{cases} 0; \text{if } z^{i,t} \neq k \\ \dfrac{1}{n_k^t}; \text{if } z^{i,t} = k \end{cases}, \text{where } n_k^t = \left|\{i : z^{i,t} = k\}\right|$$

# Representing Cluster Centers in RKHS $\mathcal{H}$

- Instead of maintaining centers explicity, maintain the coefficients needed to construct them.
- Store the following values

$$\alpha_i^{k,t} = \begin{cases} 0; \text{if } z^{i,t} \neq k \\ \frac{1}{n_k^t}; \text{if } z^{i,t} = k \end{cases}, \text{where } n_k^t = \left|\{i : z^{i,t} = k\}\right|$$

- so I can write if need be

$$\boldsymbol{\mu}^{k,t+1} = \sum_{i=1}^{n} \alpha_i^{k,t} \cdot \phi(x^i)$$

# Finding the closest center in RKHS $\mathcal{H}$

- Let the cluster center $\boldsymbol{\mu}^{k,t+1}$ be represented as

$$\boldsymbol{\mu}^{k,t+1} = \sum_{j=1}^{n} \alpha_j^{k,t} \cdot \phi(x^j)$$

- Then we have for any data point $x^i$

$$\left\|\phi(x^i) - \boldsymbol{\mu}^{k,t+1}\right\|_{\mathcal{H}}^2 = \langle \phi(x^i), \phi(x^i) \rangle + \langle \boldsymbol{\mu}^{k,t+1}, \boldsymbol{\mu}^{k,t+1} \rangle - 2 \cdot \langle \phi(x^i), \boldsymbol{\mu}^{k,t+1} \rangle$$

$$= K(x^i, x^i) + \sum_{j,l}^{n} \alpha_j^{k,t} \alpha_l^{k,t} \langle \phi(x^j), \phi(x^l) \rangle - 2 \cdot \sum_{j=1}^{n} \alpha_j^{k,t} \cdot \langle \phi(x^i), \phi(x^j) \rangle$$

$$= K(x^i, x^i) + \sum_{j,l}^{n} \alpha_j^{k,t} \alpha_l^{k,t} K(x^j, x^l) - 2 \cdot \sum_{j=1}^{n} \alpha_j^{k,t} K(x^i, x^j)$$

# Hard Assignment Kernel K-means in RKHS $\mathcal{H}$

# Hard Assignment Kernel K-means in RKHS $\mathcal{H}$

## KKM ALGORITHM

1. Given: Data $\{x^i\}_{i=1,2,\ldots,n} \in \mathcal{X}$, Kernel $K$ (with $\|\cdot\|_{\mathcal{H}}$)
2. Calculate the Gram matrix $G = [G_{ij}]$, $G_{ij} = K(x^i, x^j)$
3. Initialize cluster center coefficients $\{\boldsymbol{\alpha}^{k,0}\}_{k=1,\ldots K} \in \mathbb{R}^n$
4. For $i \in [n]$, update $z^{i,t}$
   1. Let $z^{i,t} = \arg\min_k \left(\boldsymbol{\alpha}^{k,t}\right)^\top G\boldsymbol{\alpha}^{k,t} - 2\langle G_i, \boldsymbol{\alpha}^{k,t}\rangle$
5. Update $\boldsymbol{\alpha}_i^{k,t+1} = \begin{cases} 0; \text{if } z^{i,t} \neq k \\ \frac{1}{n_k^t}; \text{if } z^{i,t} = k \end{cases}$, where $n_k^t = \left|\{i: z^{i,t} = k\}\right|$
6. Repeat until convergence

# Hard Assignment Kernel K-means in RKHS $\mathcal{H}$

## KKM ALGORITHM

$G_i$ is the i-th column of the matrix $G$

1. Given: Data $\{x^i\}_{i=1,2,\ldots,n} \in \mathcal{X}$, Kernel $K$ (with $\|\cdot$
2. Calculate the Gram matrix $G = [G_{ij}]$, $G_{ij} = K(x^i, x^j)$
3. Initialize cluster center coefficients $\{\boldsymbol{\alpha}^{k,0}\}_{k=1,\ldots,K} \in \mathbb{R}^n$
4. For $i \in [n]$, update $z^{i,t}$
    1. Let $z^{i,t} = \arg\min_k \left(\boldsymbol{\alpha}^{k,t}\right)^\top G\boldsymbol{\alpha}^{k,t} - 2\langle G_i, \boldsymbol{\alpha}^{k,t}\rangle$
5. Update $\boldsymbol{\alpha}_i^{k,t+1} = \begin{cases} 0; \text{if } z^{i,t} \neq k \\ \frac{1}{n_k^t}; \text{if } z^{i,t} = k \end{cases}$, where $n_k^t = |\{i : z^{i,t} = k\}|$
6. Repeat until convergence

# Exercises

- Develop the "soft" version of this algo – the KEM algo ☺
- Find out how can we perform the k-means++ initialization in an RKHS corresponding to a kernel $K$!
- Develop a variant of the perceptron algorithm that can work in an RKHS corresponding to a kernel $K$
  Hint: find a dual representation of the model and updates for that representation
- Develop a variant for mixed ridge regression in an RKHS corresponding to a kernel $K$

# Kernel K-NN

# Kernel k-Nearest Neighbors version 1

- Given data from some universe $\mathcal{X}$
- Given a kernel $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

**Kk-NN1**

1. Given: training set $\{x^i, y^i\}_{i=1,\dots,n}$, test point $x$
2. Find $k$ training points $i_1, i_2, \dots, i_k$ with largest value of $K(x^i, x)$
3. Predict test label $\hat{y}$ using $\{y^{i_1}, y^{i_2}, \dots, y^{i_k}\}$

# Kernel k-Nearest Neighbors version 1

- Given data from some universe $\mathcal{X}$
- Given a kernel $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

Can instead choose all points with similarity greater than a threshold (recall r-NN) to get Kr-NN ☺

See Lecture 3

**Kk-NN1**

1. Given: training set $\{x^i, y^i\}_{i=1,\ldots,n}$, test point $x$
2. Find $k$ training points $i_1, i_2, \ldots, i_k$ with largest value of $K(x^i, x)$
3. Predict test label $\hat{y}$ using $\{y^{i_1}, y^{i_2}, \ldots, y^{i_k}\}$

Can do (multi/binary) classification, regression, multi-label

Aggregate using average, weighted average, etc

# Kernel k-Nearest Neighbors version 2

- Given data from some universe $\mathcal{X}$
- Given a kernel $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with associated norm $\|\cdot\|_{\mathcal{H}}$
- We have $\left\|\phi\left(x^i\right) - \phi(x)\right\|_{\mathcal{H}}^2 = K\left(x^i, x^i\right) + K(x, x) - 2K\left(x^i, x\right)$

**Kk-NN2**

1. Given: training set $\left\{x^i, y^i\right\}_{i=1,\ldots,n}$, test point $x$
2. Find $k$ training points $i_1, i_2, \ldots, i_k$ with smallest value of $\left\|\phi\left(x^i\right) - \phi(x)\right\|_{\mathcal{H}}$
3. Predict test label $\hat{y}$ using $\left\{y^{i_1}, y^{i_2}, \ldots, y^{i_k}\right\}$

# Kernel k-Nearest Neighbors version 2

- Given data from some universe $\mathcal{X}$
- Given a kernel $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with associated norm $\|\cdot\|_{\mathcal{H}}$
- We have $\left\|\phi(x^i) - \phi(x)\right\|_{\mathcal{H}}^2 = K(x^i, x^i) + K(x, x) - 2K(x^i, x)$

See Lecture 3

Can instead choose all points with $\|\cdot\|_{\mathcal{H}}$ smaller than a threshold (recall r-NN) to get Kr-NN ☺

**Kk-NN2**

1. Given: training set $\{x^i, y^i\}_{i=1,\ldots,n}$, test point $x$
2. Find $k$ training points $i_1, i_2, \ldots, i_k$ with smallest value of $\left\|\phi(x^i) - \phi(x)\right\|_{\mathcal{H}}$
3. Predict test label $\hat{y}$ using $\{y^{i_1}, y^{i_2}, \ldots, y^{i_k}\}$

Aggregate using average, weighted average, etc

Can do (multi/binary) classification, regression, multi-label

# Kk-NN1 and Kk-NN2

- Should they not give the same results?
- **Exercise**: construct a situation where the two algorithms may give different results
- Show that there can exist data points $x, y, z \in \mathcal{X}$ and a kernel $K$ s.t.
$$K(x,y) > K(x,z), \text{but}$$
$$\|\phi(x) - \phi(y)\|_{\mathcal{H}} > \|\phi(x) - \phi(z)\|_{\mathcal{H}}$$
Hint: use the linear kernel over (non-unit) vectors in $\mathbb{R}^d$
- **Exercise**: find kernels where Kk-NN1 and Kk-NN2 always give the same results
- **Exercise**: find kernels where k-NN and Kk-NN1/2 are the same!
- There exists work on fast NN search in RKHS (look for *kernel LSH*)

# Please give your Feedback

http://tinyurl.com/ml17-18afb