# Machine Learning

## Learning with kernels

Lluís A. Belanche
belanche@cs.upc.edu



Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)
Universitat Politècnica de Catalunya

# Learning with kernels (I): The SVM

## Linear regression revisited

- **Problem**: We wish to find a function $y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ which best models a data set $D = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\} \subset \mathbb{R}^d \times \mathbb{R}$

- Then we minimize the regularized (aka penalized) empirical error:

$$E^\lambda_{\text{emp}}(y) = \sum_{n=1}^{N} \Big(t_n - y(\boldsymbol{x}_n)\Big)^2 + \lambda \sum_{i=1}^{d} w_i^2 = \|\boldsymbol{t} - X\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

  The parameter $\lambda > 0$ defines a trade-off between the fit to the data and the complexity of the vector $\boldsymbol{w}$

# Learning with kernels (I): The SVM

## Linear regression revisited

Setting $\frac{\partial E^{\lambda}_{\text{emp}}(y)}{\partial \boldsymbol{w}} = 0$, we obtain the (regularized) normal equations:

$$-2X^{\mathsf{T}}(\boldsymbol{t} - X\boldsymbol{w}) + 2\lambda \boldsymbol{w} = 0$$

with solution

$$\widehat{\boldsymbol{w}} = (X^{\mathsf{T}}X + \lambda I_d)^{-1} X^{\mathsf{T}}\boldsymbol{t}$$

and therefore

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^{\top}\boldsymbol{x}$$

# Learning with kernels (I): The SVM

## Linear regression revisited

It turns out that the regularized solution can be written as:

$$\widehat{\boldsymbol{w}} = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n, \qquad \widehat{\boldsymbol{w}} = \begin{pmatrix} \widehat{w}_1 \\ \widehat{w}_2 \\ \vdots \\ \widehat{w}_d \end{pmatrix}$$

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\boldsymbol{x}_n{}^\top \boldsymbol{x}), \qquad \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix}$$

The new vector of parameters is given by $\boldsymbol{\alpha} = (XX^\top + \lambda I_N)^{-1}\boldsymbol{t}$

# Learning with kernels (I): The SVM

## Linear regression revisited

So we have the **primal** and the **dual** forms for $y(\boldsymbol{x})$:

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \boldsymbol{x} \quad \text{and} \quad y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\boldsymbol{x}_n{}^\top \boldsymbol{x})$$

The dual form is usually more convenient when $d \gg N$:

- the primal requires the computation & inversion of $X^\mathsf{T} X + \lambda I_d$, requiring $O(Nd^2 + d^3)$ operations
- the dual requires the computation & inversion of $XX^\mathsf{T} + \lambda I_N$, requiring $O(dN^2 + N^3)$ operations

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

How can we achieve non-linear regression?

A **feature map** is a function $\phi : \mathbb{R}^d \to \mathbb{R}^M$:

$$\phi(\boldsymbol{x}) = \Big(\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_M(\boldsymbol{x})\Big)^\mathsf{T}$$

- $\phi(\boldsymbol{x})$ is called the **feature vector**
- $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}$ is the **feature space** (FS), and typically $M \gg d$.

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

- Define $\Phi_{N\times M}$ the matrix of the $\phi(\boldsymbol{x}_n)$ as

$$\phi_{nm} = \phi_m(\boldsymbol{x}_n), n = 1, \ldots, N, m = 1, \ldots, M.$$

- Suppose we perform ridge regression on the $\Phi$ matrix

- The new regression function has the **primal** representation:

$$y(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \phi(\boldsymbol{x})$$

Note the primal now (explicitly) operates in feature space

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

Given a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^M$, we define its associated **kernel function** $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u})^\top \phi(\boldsymbol{v}), \qquad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$$

- The feature space where $k$ *implicitly* operates is $\mathbb{R}^M$
- For some feature maps, computing $k(\boldsymbol{u}, \boldsymbol{v})$ is independent of $M$

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

Since $\widehat{\boldsymbol{w}} = \sum_{n=1}^{N} \alpha_n \phi(\boldsymbol{x}_n)$, the new regression function has the **dual** representation:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n (\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x})) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

The new vector of parameters is given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda I_N)^{-1} \boldsymbol{t}, \qquad \text{where } \mathbf{K} = \Big(k(\boldsymbol{x}_n, \boldsymbol{x}_m)\Big)$$

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

Many (classical and new) learning algorithms can be "kernelized":

- The Support Vector Machine (SVM) and the Relevance Vector Machine (RVM)
- Fisher Discriminant Analysis (KFDA), Principal Components Analysis (KPCA), Canonical Correlation Analysis (KCCA), ...
- Kernel (regularized) linear regression
- Kernel k-means, kernel kNN
- (less known or very recent): PLS, Parzen Windows, logistic regression, statistical tests, ...

# Learning with kernels (I): The SVM

## Key aspects of kernel methods

Kernel-based methods consist of two ingredients:

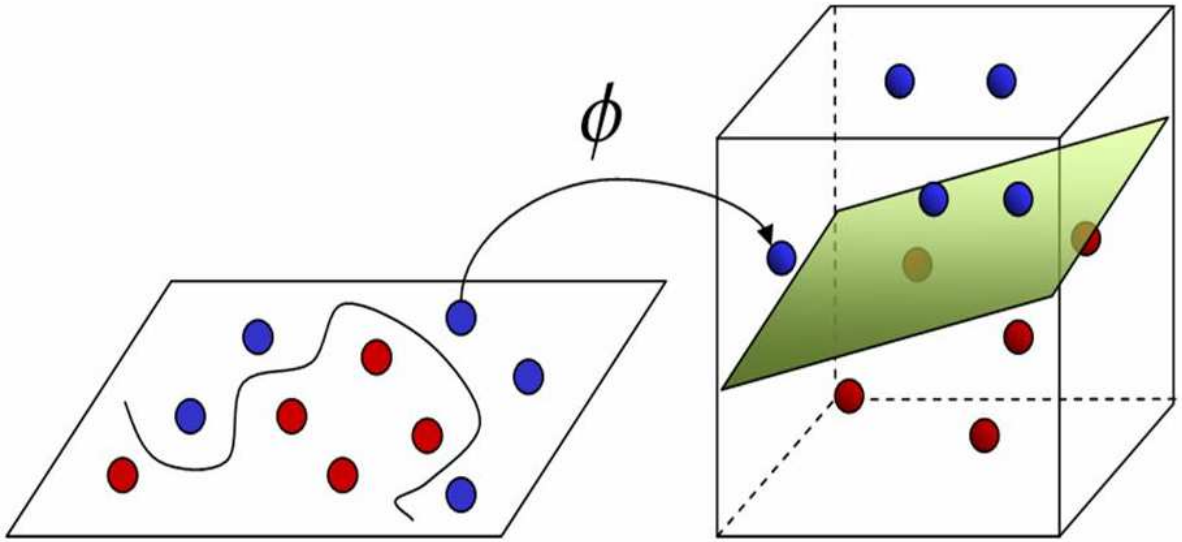
1. The kernel function (this is non-trivial)

2. The algorithm taking kernels as input

   - Data items are embedded into a vector space (feature space FS)
   - Linear relations are sought among the elements of the FS
   - The coordinates of these images are not needed: only their pairwise inner products
   - These inner products can sometimes be computed efficiently and implicitly in the input space (kernel function)
   - The solution vector is expressed as a linear combination of the kernel centered at the data

# Learning with kernels (II): Kernel functions

A kernel function implicitly defines a map $\phi : \mathcal{X} \to \mathcal{H}$ from an input space of objects $\mathcal{X}$ into some Hilbert space $\mathcal{H}$ (called the *feature space*). The "kernel trick" consists in performing the mapping and the inner product simultaneously by defining its associated kernel function:




$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle_{\mathcal{H}}, \ \boldsymbol{u}, \boldsymbol{v} \in \mathcal{X},$$

# Learning with kernels (II): Kernel functions

## The Kernel Trick

- Suppose we take $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle^d$ (a simple choice).
- What is the underlying mapping $\phi$ here?

  $\Longrightarrow$ Answer: this choice of kernel corresponds to a map $\phi$ leading into the space spanned by all products of exactly $d$ dimensions of $\mathbb{R}^n$.
- Let us take, for instance, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$, and take $d = 2$:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle^2 = \left[ \left\langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \right\rangle \right]^2$$

$$= (u_1 v_1 + u_2 v_2)^2 = (u_1 v_1)^2 + 2 u_1 v_1 u_2 v_2 + (u_2 v_2)^2$$

$$= u_1^2 v_1^2 + (\sqrt{2} u_1 u_2)(\sqrt{2} v_1 v_2) + u_2^2 v_2^2$$

$$= \left\langle \begin{pmatrix} u_1^2 \\ \sqrt{2} u_1 u_2 \\ u_2^2 \end{pmatrix}, \begin{pmatrix} v_1^2 \\ \sqrt{2} v_1 v_2 \\ v_2^2 \end{pmatrix} \right\rangle = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle$$

- Therefore, $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ with $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T$

# Kernel-Based Learning

## Key aspects of kernel methods

- A feature map is of the general form $\phi : \mathcal{X} \to \mathcal{H}$. The associated kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle_{\mathcal{H}}, \ \boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}$

- $\mathcal{X}$ can be any space, $\mathcal{H}$ is any **Hilbert space**:

  - An abstract complete vector space possessing the structure of an inner product

  - Examples would be $\mathbb{R}^M$ or the $l_2$ space of square-summable sequences

---

In our previous discussion, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{H} = \mathbb{R}^M$

# Learning with kernels (II): Kernel functions

## Characterization of Kernels

**Definition**: A symmetric function $k$ is called **positive semi-definite** in $\mathcal{X}$ if:

for every $N \in \mathbb{N}$, and every choice $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N \in \mathcal{X}$,

the matrix $\mathbf{K} = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is **positive semi-definite**.

---

**Theorem**: $k$ admits the existence of a map $\phi : \mathcal{X} \to \mathcal{H}$ s.t.

$\mathcal{H}$ is a Hilbert space and $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle_{\mathcal{H}}$

if and only if $k$ is a positive semi-definite symmetric function in $\mathcal{X}$.

# Learning with kernels (II): Kernel functions

## On positive semi-definiteness

There are many equivalent characterizations of the psd (*positive semi-definite*) property for real symmetric matrices. Here are some:

1. $A_{d\times d}$ is psd if and only if all of its eigenvalues are non-negative.

2. $A_{d\times d}$ is psd if and only if the determinants of all of its leading principal minors are non-negative.

3. $A_{d\times d}$ is psd if and only if there is a psd matrix $B$ such that $BB^T = A$ (this matrix B is unique and called the square root of $A$).

4. $A_{d\times d}$ is psd if and only if, $\forall \boldsymbol{c} \in \mathbb{R}^d,\ \boldsymbol{c}^T A \boldsymbol{c} \geq 0$.

# Learning with kernels (II): Kernel functions

## General linear kernel

If $A_{d\times d}$ is a psd matrix, then the function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T A \boldsymbol{v}$ is a kernel.

*Proof.* Since A is psd we can write it in the form $A = BB^T$. For every $N \in \mathbb{N}$, and every choice $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$, we form the matrix $K = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$. Then for every $\boldsymbol{c} \in \mathbb{R}^N$:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j k_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \boldsymbol{x}_i^T A \boldsymbol{x}_j = \sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j (B^T \boldsymbol{x}_i)^T (B^T \boldsymbol{x}_j)$$

$$= \left\| \sum_{i=1}^{N} c_i (B^T \boldsymbol{x}_i) \right\|^2 \geq 0. \qquad \text{Note that } \phi(\boldsymbol{x}) = B^T \boldsymbol{x}$$

# Learning with kernels (II): Kernel functions

## Summary of kernel properties

If $k, k'$ are kernels on $\mathbb{R}^d$, $k''$ is a kernel on $\mathbb{R}^N$, $a, b \geq 0$, and $\phi : \mathbb{R}^d \to \mathbb{R}^N$, then the following functions are kernels on $\mathbb{R}^d$:

1. $k(\boldsymbol{u}, \boldsymbol{v}) + k'(\boldsymbol{u}, \boldsymbol{v})$

2. $ak(\boldsymbol{u}, \boldsymbol{v}) + b$

3. $k(\boldsymbol{u}, \boldsymbol{v})k'(\boldsymbol{u}, \boldsymbol{v})$

4. $k''(\phi(\boldsymbol{u}), \phi(\boldsymbol{v}))$

# Learning with kernels (II): Kernel functions

## Normalization

If $k$ is a kernel, then so is:

$$k_n(\boldsymbol{u}, \boldsymbol{v}) := \frac{k(\boldsymbol{u}, \boldsymbol{v})}{\sqrt{k(\boldsymbol{u}, \boldsymbol{u})}\sqrt{k(\boldsymbol{v}, \boldsymbol{v})}}$$

Moreover,

$$\begin{aligned} |k_n(\boldsymbol{u}, \boldsymbol{v})| &\leq 1 \\ k_n(\boldsymbol{u}, \boldsymbol{u}) &= 1 \end{aligned}$$

# Learning with kernels (II): Kernel functions

## Polynomial combinations

1. If $k$ is a kernel and $p$ is a polynomial of degree $m$ with positive coefficients, then the function

$$k_p(\boldsymbol{u}, \boldsymbol{v}) = p(k(\boldsymbol{u}, \boldsymbol{v}))$$

   is also a kernel.

2. The special case where $k$ is linear and $p(z) = (az + 1)^m$ leads to the so-called **polynomial kernel**

# Learning with kernels (II): Kernel functions

## Polynomial combinations

Consider the kernel family:

$$\Big\{k_i(\boldsymbol{u},\boldsymbol{v}) = \alpha_i\Big(\langle \boldsymbol{u},\boldsymbol{v}\rangle + a_i\Big)^{\beta_i} \;/\; \beta_i \in \mathbb{N}, \alpha_i > 0, a_i \geq 0\Big\}$$

For any $q > 0 \in \mathbb{N}$,

$$\sum_{i=0}^{q} k_i(\boldsymbol{u},\boldsymbol{v})$$

is a kernel.

# Learning with kernels (II): Kernel functions

## Polynomial combinations

Consider the particular case $a_i = 0, \beta_i = i$ and $\alpha_i = \frac{\alpha^i}{i!}$, for some real $\alpha > 0$, and take the limit $q \to \infty$.

The obtained series is convergent for all $\alpha$ and the resulting kernel is:

$$\sum_{i=0}^{\infty} \frac{\alpha^i}{i!}(\langle \boldsymbol{u}, \boldsymbol{v} \rangle)^i = e^{\alpha \langle \boldsymbol{u}, \boldsymbol{v} \rangle}$$

Assume that $u, v \in \mathbb{R}$; then $\exp(\alpha u v) = \langle \phi(u), \phi(v) \rangle$ with $\phi(z) = \left( \sqrt{\frac{\alpha^i}{i!}} z^i \right)_{i=0}^{\infty}$, and therefore we have designed a feature space of infinite dimension!

# Learning with kernels (II): Kernel functions

## Translation invariant and radial kernels

We say that a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is:

**Translation invariant** if it has the form $k(\boldsymbol{u}, \boldsymbol{v}) = T(\boldsymbol{u} - \boldsymbol{v})$, where $T : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function.

**Radial** if it has the form $k(\boldsymbol{u}, \boldsymbol{v}) = t(\|\boldsymbol{u} - \boldsymbol{v}\|)$, where $t : [0, \infty) \to [0, \infty)$ is a differentiable function.

Radial kernels fulfill $k(\boldsymbol{u}, \boldsymbol{u}) = t(0)$.

# Learning with kernels (II): Kernel functions

## The Gaussian kernel

Consider the function $t(z) = \exp(-\gamma z^2), \gamma > 0$. The resulting radial kernel is known as the **Gaussian RBF kernel**:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2)$$

Note that some people call it the RBF kernel *par excellence*!

You can also find it as:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{2\sigma^2}\right)$$

# Learning with kernels (II): Kernel functions

## Popular choices for the Kernel

**Polynomial kernels** (relation to GLDs)

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\langle \boldsymbol{u}, \boldsymbol{v} \rangle + 1)^d, \ d \in \mathbb{N}$$

**Gaussian RBF kernels** (relation to RBFNNs)

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2\right) \ \gamma > 0 \in \mathbb{R}$$

**Laplacian RBF kernels** (relation to ???)

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|\right) \ \gamma > 0 \in \mathbb{R}$$

**Sigmoidal kernels** (relation to MLPs)

$$k(\boldsymbol{u}, \boldsymbol{v}) = g(\alpha \langle \boldsymbol{u}, \boldsymbol{v} \rangle + \beta)$$

with $g$ a sigmoidal (e.g., logistic, tanh, ...) and particular choices for $\alpha, \beta$.

# Learning with kernels (II): Kernel functions

## Euclidean space $\mathbb{R}^d$, but not only

- Kernels on sets/bitstrings
- Graph kernels
- Generative kernels (on probability distributions)
- Convolution kernels (on combinatorial structures)
- Tree kernels
- String kernels (text)

... and many others (functional data, categorical data, ...)

# Learning with kernels (II): Kernel functions

## A kernel for set comparison

Given two sets $A, B$, consider

$$k(A, B) = \sum_{a \in A} \sum_{b \in B} k_{\text{base}}(a, b)$$

If $k_{\text{base}}$ is the overlap kernel:

$$k(a, b) = \begin{cases} 1 & \text{if } a = b; \\ 0 & \text{otherwise.} \end{cases}$$

we get $k(A, B) = |A \cap B|$. Remarkably, $k(A, B) = \dfrac{|A \cap B|}{|A \cup B|}$ is also a kernel.

# Learning with kernels (II): Kernel functions

## Kernels for/from graphs (I)

- Consider a graph $G = (V, E)$, where the set of vertices (nodes) $V$ are the data points and $E$ is the set of edges. Call $N = |V|$, the number of nodes
- The idea is to compute a (base) matrix $B_{N \times N}$ whose entries are the weights of the edges and consider $B^2 = BB$ ($B$ need not be symmetric)
- Typical use: **connectivity matrix** of $G$: the $(i, j)$ element of $B^2$ is the number of paths of length exactly 2 between $i$ and $j$

---

Examples:

1. protein-protein interactions
2. people-to-people interactions

---

In 2, the $(i, j)$ element of $B^2$ is the number of common friends between data points $i$ and $j$ (it can be thought of as a measure of their similarity)

# Learning with kernels (II): Kernel functions

## Kernels for/from graphs (II)

Notes:

1. The entries of $B$ may be real-valued numbers (e.g., symmetric bounded similarities)

2. Higher powers of $B$ measure higher-order similarities

3. Only the even powers are guaranteed to be PSD

Consider, for a given $\lambda \in (0, 1)$:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \lambda^k B^k = \exp(\lambda B)$$

If $B$ is symmetric, then $B = U \Lambda U^T$ is its spectral decomposition, so $B^2 = (U \Lambda U^T)(U \Lambda U^T) = U \Lambda^2 U^T$. In general, we have $B^k = U \Lambda^k U^T$ and therefore:

$$K = \exp(\lambda B) = U \exp(\lambda \Lambda) U^T$$

is an example of a **diffusion** kernel (the name comes from the *heat equation* in physics)

# Learning with kernels (II): Kernel functions

## Afterthoughts

1. Importance of designing kernels that do not constitute explicit inner products between objects, and therefore fully exploit the kernel trick.

2. Possibility of learning the kernel function (or the kernel matrix) from the training data.

3. Need more research for handling special situations –like missing, imprecise or not-applicable (NA) values.

4. Theoretical analyses on the implications of the kernel choice for the success of kernel-based methods.