January 9, 2026

Inter-University Research Institute Corporation /
Research Organization of Information and Systems
National Institute of Informatics

SVM and Kernel Methods

Data Mining 10 (データマイニング)

Mahito Sugiyama (杉山麿人)

Today's Outline

- Today's topic is support vector machines (SVMs) and kernel methods
- SVM performs binary classification by maximizing the margin
  - It is (was) a popular supervised classification method
- SVM can perform nonlinear classification for structured data using kernel trick
- Graph kernels for classification for graph structured data

Classification Problem Setting

- Given a supervised dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ (feature vector) and $y_i \in C = \{-1, 1\}$ (label)
- Use a decision function (hyperplane) in the form of

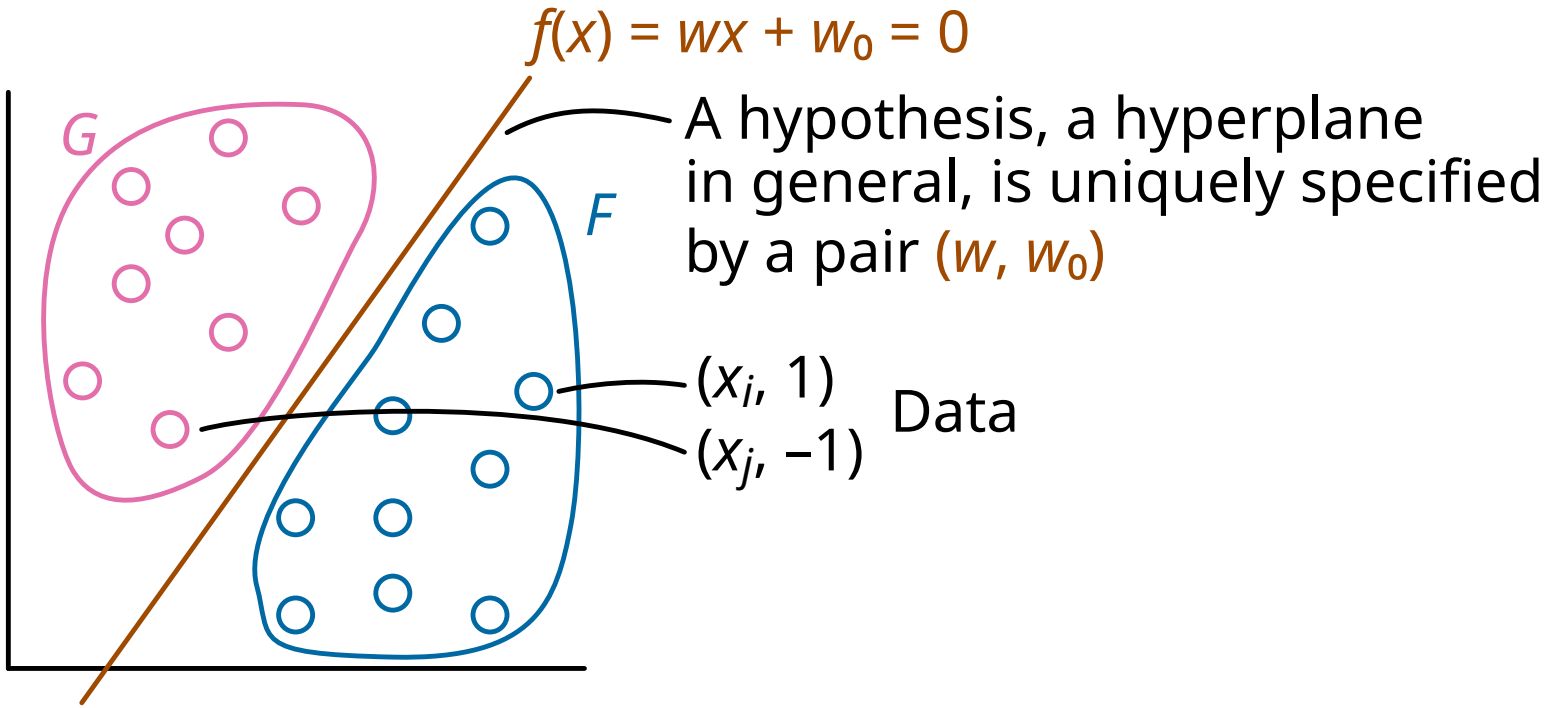
  $$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 = \sum_{j=1}^{d} w^j x^j + w_0$$

- A classifier $g(\boldsymbol{x})$ is given as

  $$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } f(\boldsymbol{x}) > 0, \\ -1 & \text{if } f(\boldsymbol{x}) < 0 \end{cases}$$

- Goal: Find $(\boldsymbol{w}, w_0)$ that correctly classifies the dataset

Classification by Hyperplane



Learning Procedure of Perceptron

1. $\boldsymbol{w} \leftarrow \mathbf{0}$, $w_0 \leftarrow 0$ (or a small random value) // initialization
2. for $i = 1, 2, 3, \ldots$ do
3. Receive i-th pair $(\boldsymbol{x}_i, y_i)$
4. Compute $a = \sum_{j=1}^{d} w^j x_i^j + w_0$
5. if $y_i \cdot a < 0$ then // $\boldsymbol{x}_i$ is misclassified
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{x}_i$ // update the weight
7. $w_0 \leftarrow w_0 + y_i$ // update the bias
8. end if
9. end for

Correctness of Perceptron

- It is guaranteed that a perceptron always converges to a correct classifier (the convergence theorem)
- A correct classifier is a function f s.t.

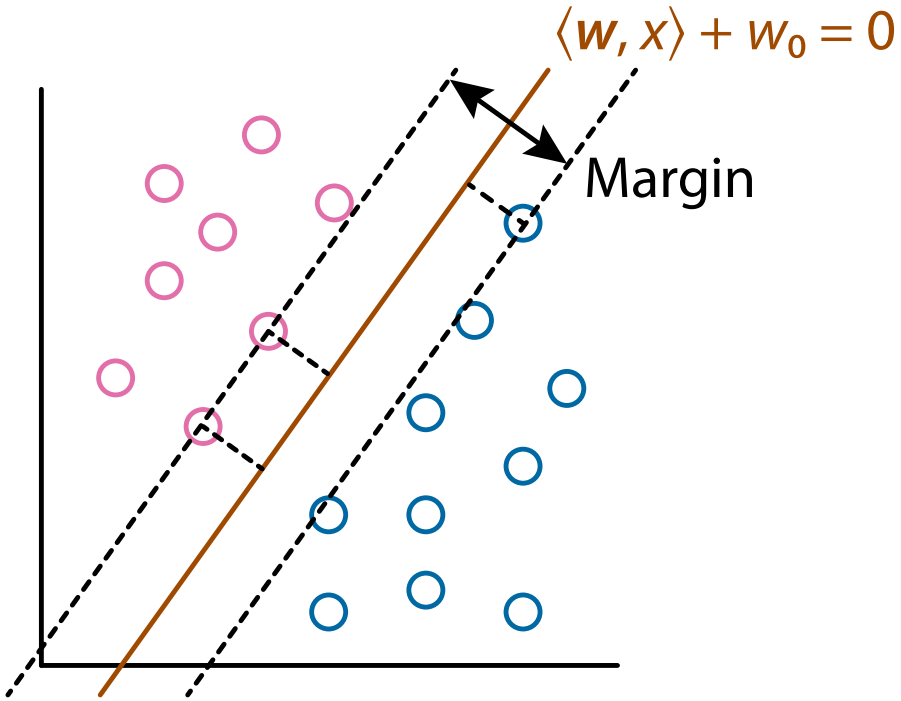
$$\begin{cases} f(\boldsymbol{x}) > 0 & \text{if } y = 1, \\ f(\boldsymbol{x}) < 0 & \text{if } y = -1, \end{cases}$$

- Note: there are (infinitely) many functions that correctly classify F and G
  - A perceptron just converges to one of them
- How can we find the "best" classifier among them...?
  - $\rightarrow$ Support Vector Machines (SVMs)!

Support Vector Machines (SVMs)

- In the following, we denote $\{1, 2, \dots, n\}$ by $[n]$
- A dataset D is separable by f
  $\iff y_i f(\boldsymbol{x}_i) > 0, \forall i \in [n]$
- The margin is the distance from the classification hyperplane to the closest data point
- Support vector machines (SVMs) tries to find a hyperplane that maximizes the margin

Margin



Formulation of SVMs (1/2)

- The distance from $\boldsymbol{x}_i$ to a hyperplane $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 = 0$ is

$$\frac{|f(\boldsymbol{x}_i)|}{\|\boldsymbol{w}\|} = \frac{|\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0|}{\|\boldsymbol{w}\|} = \frac{|\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + w_0|}{\sqrt{(w^1)^2 + (w^2)^2 + \dots (w^d)^2}}$$

- The margin is the distance to the closest point; that is,

$$\min_{i \in [n]} \frac{|f(\boldsymbol{x}_i)|}{\|\boldsymbol{w}\|}$$

- The margin maximization problem is given as

$$\max_{\boldsymbol{w}, w_0} \min_{i \in [n]} \frac{|f(\boldsymbol{x}_i)|}{\|\boldsymbol{w}\|} \quad \text{subject to } \; y_i f(\boldsymbol{x}_i) > 0, \forall i \in [n]$$

- Can we simplify more...?

Formulation of SVMs (2/2)

- Since the hyperplane is invariant up to multiplication by a constant, we can assume

  $\min_{i \in [n]} |f(\boldsymbol{x}_i)| = 1$

- Then the margin is: $\min_{i \in [n]} |f(\boldsymbol{x}_i)| / \|\boldsymbol{w}\| = 1/\|\boldsymbol{w}\|$
- Moreover, if f perfectly classifies the dataset, $\forall i \in [n]$,

  $$\begin{cases} f(\boldsymbol{x}_i) \geq 1 & \text{if } y_i = 1, \\ f(\boldsymbol{x}_i) \leq -1 & \text{if } y_i = -1, \end{cases} \iff y_i f(\boldsymbol{x}_i) \geq 1$$

- Finally, the margin maximization problem is simplified and given as

  $$\max_{\boldsymbol{w}, w_0} 1/\|\boldsymbol{w}\| \quad \text{subject to } y_i f(\boldsymbol{x}_i) \geq 1, \forall i \in [n]$$

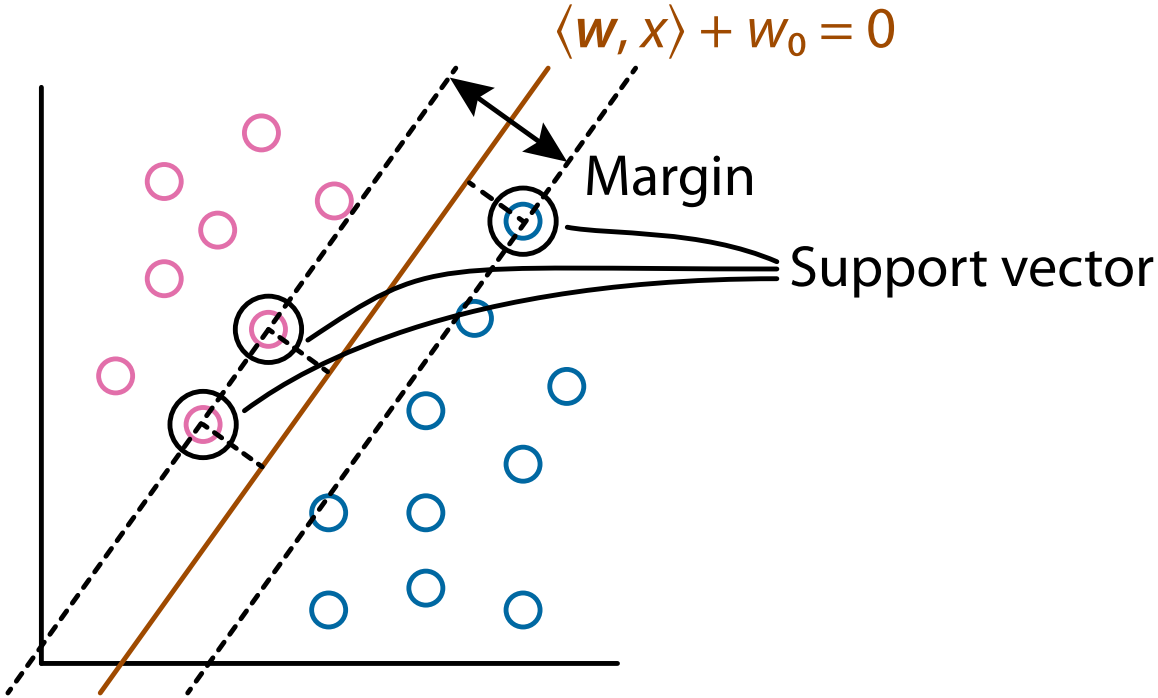
Hard Margin SVMs

- We can replace "max" with "min" and obtain

  $$\min_{\boldsymbol{w}, w_0} \|\boldsymbol{w}\| \quad \text{subject to } y_i f(\boldsymbol{x}_i) \geq 1, \forall i \in [n]$$

- This is the standard formulation of hard margin SVMs
- Points $\boldsymbol{x}_i$ satisfying $y_i f(\boldsymbol{x}_i) = 1$ are called support vectors
- The solution does not change even data points that are not support vectors are removed

Margin



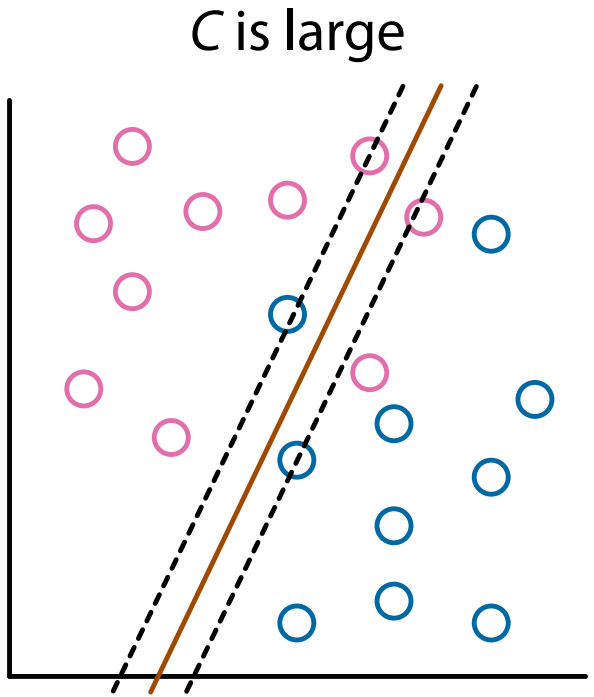
Soft Margin

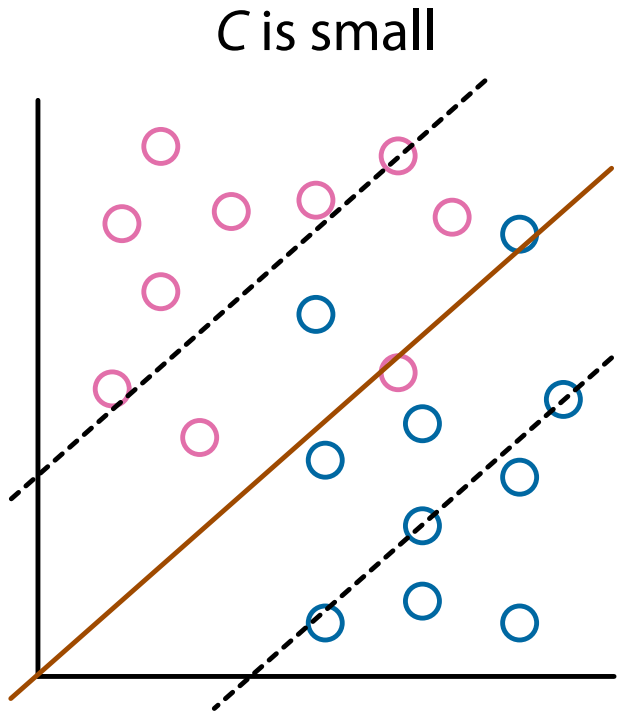
- Datasets are not often separable
- Extend SV classification to soft margin by relaxing the constraint
- Change the constraint $y_i f(\boldsymbol{x}_i) \geq 1$ using the slack variable ξ_i to $y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i$
- The formulation of soft margin SVM (C-SVM) is

$$\min_{\boldsymbol{w}, w_0, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i \in [n]} \xi_i \quad \text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0, i \in [n]$$

  - C is called the regularization parameter

Soft Margin





Location of Data Points

- $y_i f(\boldsymbol{x}_i) > 1$: $\boldsymbol{x}_i$ is outside margin
  - These points do not affect to the classification hyperplane
- $y_i f(\boldsymbol{x}_i) = 1$: $\boldsymbol{x}_i$ is on margin
- $y_i f(\boldsymbol{x}_i) < 1$: $\boldsymbol{x}_i$ is inside margin
  - These points do not exist in hard margin
- Points are support vectors if they are on margin or inside margin

Dual Problem (1/4)

- The original formulation of C-SVM

$$\min_{\boldsymbol{w}, w_0, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\{1,2,\dots,n\}} \xi_i \quad \text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0, i \in \{1, 2, \dots, n\}$$

  is called the primal problem

- This is usually solved via the dual problem with the Lagrange function using $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n), \boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$:

$$L(\boldsymbol{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$$

$$= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in[n]} \xi_i - \sum_{i\in[n]} \alpha_i\left(y_i f(\boldsymbol{x}_i) - (1 - \xi_i)\right) - \sum_{i\in[n]} \mu_i \xi_i$$

Dual Problem (2/4)

- Let us consider

  $D(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \min_{\boldsymbol{w}, w_0, \boldsymbol{\xi}} L(\boldsymbol{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$

  and its maximization (dual problem):

  $\max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\mu} \geq 0} D(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\mu} \geq 0} \min_{\boldsymbol{w}, w_0, \boldsymbol{\xi}} L(\boldsymbol{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$

- The above dual problem is equivalent to the primal problem
- Note that there is no constraint on $\boldsymbol{w}$, w_0, and $\boldsymbol{\xi}$ in the dual problem

Dual Problem (3/4)

- The inside of minimization is achieved when
$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i \in [n]} \alpha_i y_i \boldsymbol{x}_i = 0, \quad \frac{\partial L}{\partial w_0} = -\sum_{i \in [n]} \alpha_i y_i = 0, \quad \frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$
- Putting the three conditions to the Lagrange function to remove $\boldsymbol{w}$, w_0, and $\boldsymbol{\xi}$, yielding
$$\begin{aligned} L &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i \in [n]} \alpha_i y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - w_0 \sum_{i \in [n]} \alpha_i y_i + \sum_{i \in [n]} \alpha_i + \sum_{i \in [n]} (C - \alpha_i - \mu_i)\xi_i \\ &= -\frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i \in [n]} \alpha_i \end{aligned}$$

Dual Problem (4/4)

- Finally, the soft margin SVM is solved via the following optimization:

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i,j\in[n]} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i\in[n]} \alpha_i$$

$$\text{subject to } \sum_{i\in[n]} \alpha_i y_i = 0,\ 0 \le \alpha_i \le C,\ i \in [n]$$

KKT (Karush-Kuhn-Tucker) condition

- The necessary conditions for a solution to be optimal:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i \in [n]} \alpha_i y_i \boldsymbol{x}_i = 0, \ \frac{\partial L}{\partial w_0} = -\sum_{i \in [n]} \alpha_i y_i = 0, \ \frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$-(y_i f(\boldsymbol{x}_i) - 1 + \xi_i) \leq 0, \ -\xi_i \leq 0,$$

$$\alpha_i \geq 0, \ \mu_i \geq 0,$$

$$\alpha_i (y_i f(\boldsymbol{x}_i) - 1 - \xi_i) = 0, \ \mu_i \xi_i = 0,$$

$$i \in [n]$$

Recovering Primal Variables

- Using these conditions, from the optimal $\boldsymbol{\alpha}$, we have

$$f(\boldsymbol{x}) = \sum_{i \in [n]} \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + w_0,$$

$$w_0 = y_i - \sum_{j \in [n]} \alpha_j y_j \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle, \quad \forall i \in \{ i \in [n] \mid 0 < \alpha_i < C \}$$

- Location of points:
  - $y_i f(\boldsymbol{x}_i) > 1 \iff \alpha_i = 0$: $\boldsymbol{x}_i$ is outside margin
  - $y_i f(\boldsymbol{x}_i) = 1 \iff 0 < \alpha_i < C$: $\boldsymbol{x}_i$ is on margin
  - $y_i f(\boldsymbol{x}_i) < 1 \iff \alpha_i = C$: $\boldsymbol{x}_i$ is inside margin

How to Solve?

- The (dual) problem:

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} + \mathbf{1}^T\boldsymbol{\alpha} \quad \text{s.t. } \boldsymbol{y}^T\boldsymbol{\alpha} = 0,\ 0 \le \boldsymbol{\alpha} \le C\mathbf{1}$$

  - $Q \in \mathbb{R}^{n\times n}$ is the matrix such that $q_{ij} = y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$

- Since analytical solution is not available, iterative approach for continuous optimization with constraints is needed
- One of standard methods is the active set method

Active Set Method

- Divide the set $[n]$ of indices into three sets:

  $$O = \{i \in [n] \mid \alpha_i = 0\}$$

  $$M = \{i \in [n] \mid 0 < \alpha_i < C\}$$

  $$I = \{i \in [n] \mid \alpha_i = C\}$$

  - O and I are called active sets
- The problem can be solved w.r.t. $i \in M$, yielding

  $$\begin{bmatrix} Q_M & \boldsymbol{y}_M \\ \boldsymbol{y}_M^T & 0 \end{bmatrix} \begin{bmatrix} \alpha_M \\ \nu \end{bmatrix} = -C \begin{bmatrix} Q_{M,I} & \mathbf{1} \\ \mathbf{1}^T & \boldsymbol{y}_I \end{bmatrix} + \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix}$$

  - This can be directly solved if Q_M is positive definite

Algorithm 1: Active Set Method

ACTIVESETMETHOD(D)

  Initialize M, I, O

  while *there exists* i *s.t.* $y_i f(\boldsymbol{x}_i) < 1$, $i \in O$ *or* $y_i f(\boldsymbol{x}_i) > 1$, $i \in I$ **do**

    Update M, I, O

    repeat

      $\boldsymbol{\alpha}_M^{\text{new}} \leftarrow$ the solution of the above equation

      $\boldsymbol{d} \leftarrow \boldsymbol{\alpha}_M^{\text{new}} - \boldsymbol{\alpha}_M$

      $\boldsymbol{\alpha}_M \leftarrow \boldsymbol{\alpha}_M + \eta \boldsymbol{d}$; // max. η satisfying $\boldsymbol{\alpha}_M \in [0, C]^{|M|}$

      Move $i \in M$ from M to I or O if $\alpha_i = C$ or $\alpha_i = 0$

    until $\boldsymbol{\alpha}_M = \boldsymbol{\alpha}_M^{new}$;

Extension to Nonlinear Classification

- To achieve nonlinear classification, convert each data point $\boldsymbol{x}$ to some point $\phi(\boldsymbol{x})$, and $f(\boldsymbol{x})$ becomes

  $$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle + w_0$$

- The dual problem becomes

  $$\max_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle + \sum_{i \in [n]} \alpha_i$$

  $$\text{subject to } \sum_{i \in [n]} \alpha_i y_i = 0,\ 0 \le \alpha_i \le C, i \in [n]$$

  - Only the dot product $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ is used!
  - We do not even need to know $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$

C-SVM with Kernel Trick

- Use a kernel function: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, we have

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i \in [n]} \alpha_i$$

$$\text{subject to } \sum_{i \in [n]} \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C, i \in [n]$$

- The technique of using K is called kernel trick
- Intuitively, $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ represents the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
- The kernel needs to be positive semidefinite so that the resulting feature space exists $\rightarrow$ the optimization won't be convex

Kernel Regression

- From regression:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2$$

to kernel regression:

$$\min \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2 = \min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{N} \alpha_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) \right)^2$$

  - Solved as $\boldsymbol{\alpha} = K^{-1} \boldsymbol{y}$
  - For a new data point $\boldsymbol{x}'$, its prediction is given as $\sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}')$
  - (Kernel) ridge regression (by adding $\lambda \|\boldsymbol{\beta}\|_2^2$) is often used

Positive Definite Kernel

- A kernel $K : \Omega \times \Omega \to \mathbb{R}$ is a positive definite kernel if
  - (i) $K(x, y) = K(y, x)$
  - (ii) For $x_1, x_2, \dots, x_n$, the $n \times n$ matrix (called Gram matrix)

$$(K_{ij}) = \begin{bmatrix} K(x_1, x_1) & K(x_2, x_1) & \dots & K(x_n, x_1) \\ K(x_1, x_2) & K(x_2, x_2) & \dots & K(x_n, x_2) \\ \dots & \dots & \dots & \dots \\ K(x_1, x_n) & K(x_2, x_n) & \dots & K(x_n, x_n) \end{bmatrix}$$

  is positive semidefinite. Equivalent conditions of PSD are
    - There exists B s.t. $(K_{ij}) = B^T B$
    - $\boldsymbol{c}^T (K_{ij}) \boldsymbol{c} \geq 0$ for any $\boldsymbol{c} \in \mathbb{R}^n$
    - All eigenvalues of (K_{ij}) are nonnegative

Popular Positive Definite Kernels

- Linear Kernel

  $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$

- Gaussian (RBF) kernel

  $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right)$

- Polynomial Kernel

  $K(\boldsymbol{x}, \boldsymbol{y}) = \left(\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c\right)^c \quad c, d \in \mathbb{R}$

Simple Kernels

- The all-ones kernel

  $K(\boldsymbol{x}, \boldsymbol{y}) = 1$

- The delta (Dirac) kernel

  $$K(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{y}, \\ 0 & \text{otherwise} \end{cases}$$

Closure Properties of Kernels

- For two kernels K_1 and K_2, $K_1 + K_2$ is a kernel
- For two kernels K_1 and K_2, the product $K_1 \cdot K_2$ is a kernel
- For a kernel K and a positive scalar $\lambda \in \mathbb{R}^+$, λK is a kernel
- For a kernel K on a set D, its zero-extension:

  $$K_0(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} K(\boldsymbol{x}, \boldsymbol{y}) & \text{if } \boldsymbol{x}, \boldsymbol{y} \in D, \\ 0 & \text{otherwise} \end{cases}$$

  is a kernel

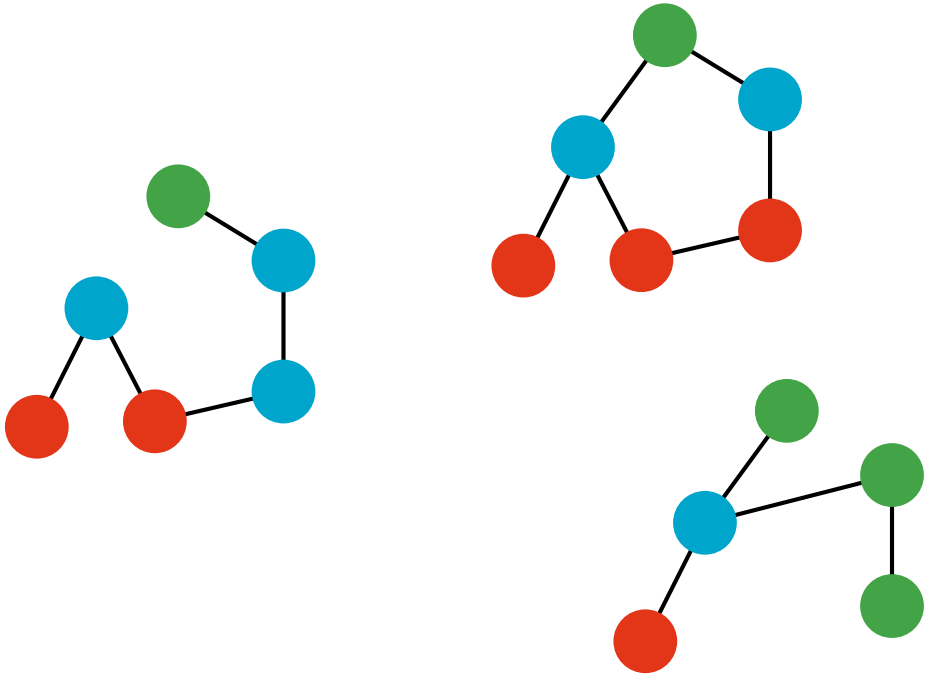
Kernels on Structured Data

- Given objects X and Y, decompose them into substructures S and T
- The R-convolution kernel K_R by Haussler (1999) is given as

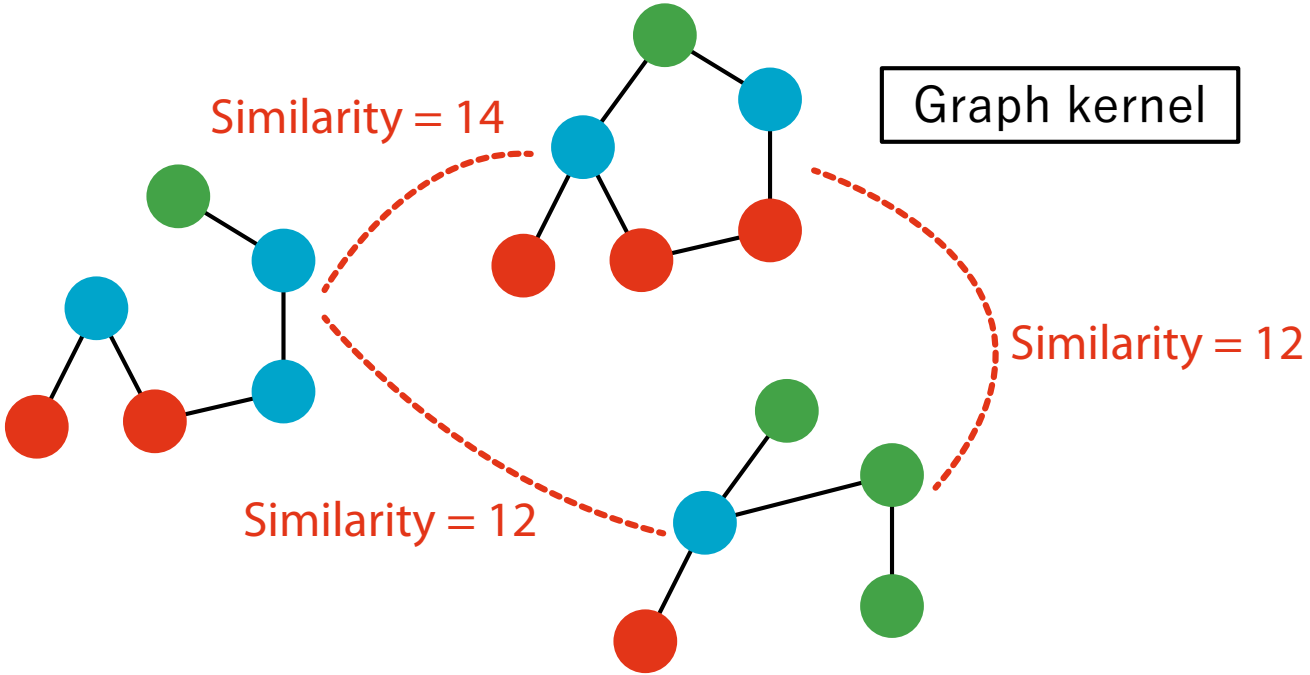
$$K_R(X, Y) = \sum_{s \in S, t \in T} K_{\text{base}}(s, t)$$

  - K_{base} is an arbitrary base kernel, often the delta kernel
- For example, X (resp. Y) is a graph and S (resp. T) is the set of all subgraphs of X (resp. Y)

Similarity between Graphs

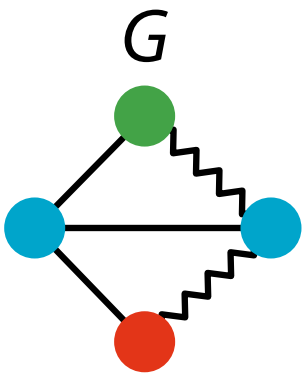


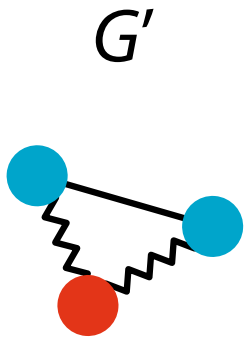
Similarity between Graphs



Example

Vertex Label Histogram Kernel





G	2	1	1
G'	2	0	1

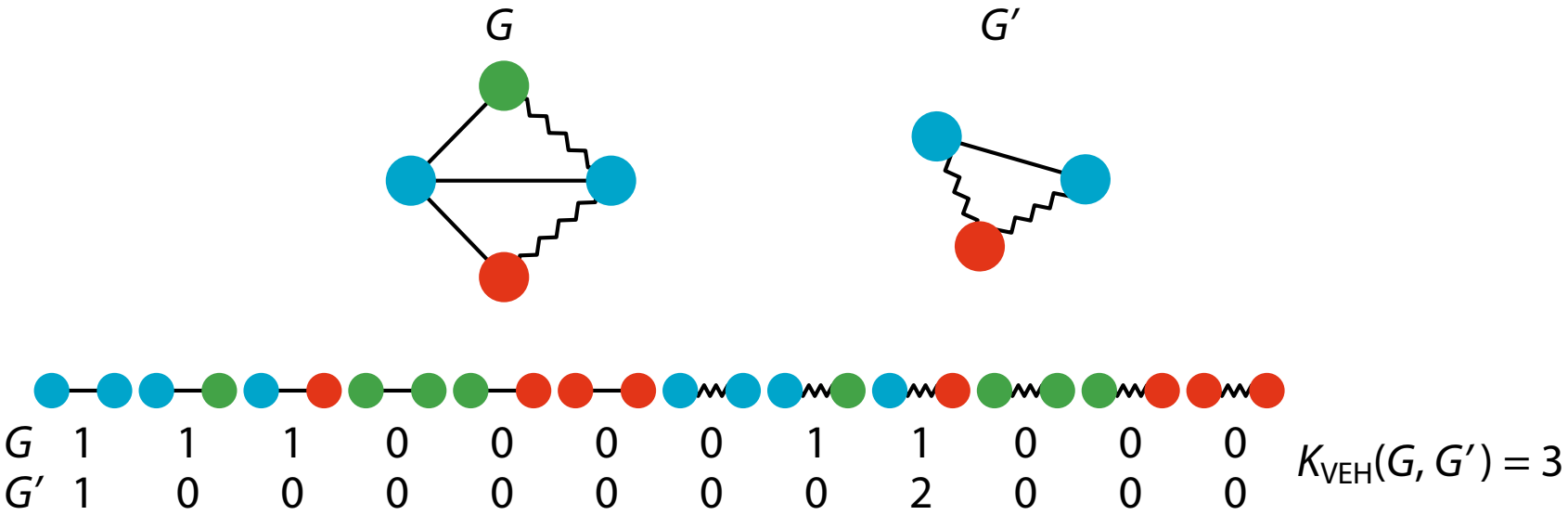
$$K_{VH}(G, G') = 2 \cdot 2 + 1 \cdot 0 + 1 \cdot 1 = 5$$

Edge Label Histogram Kernel

	—	≈
G	3	2
G'	1	2

$$K_{\mathrm{EH}}(G, G') = 3 \cdot 1 + 2 \cdot 2 = 7$$

Vertex-Edge Label Histogram Kernel



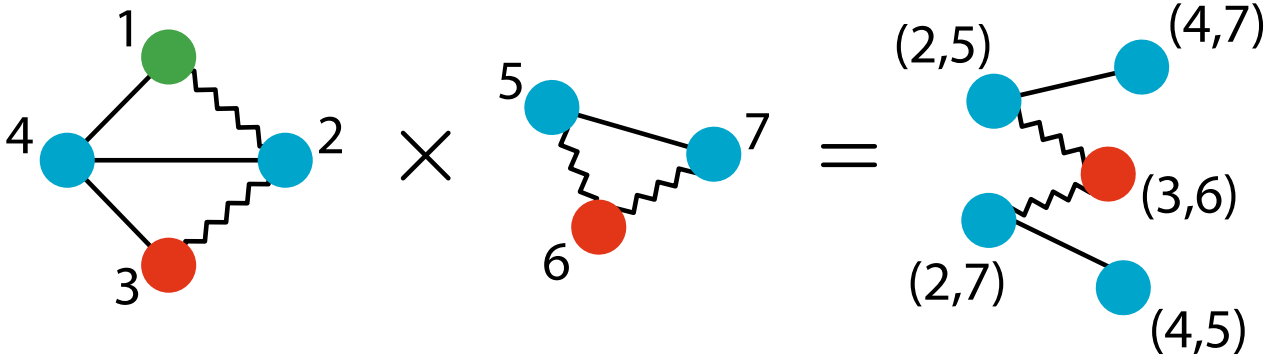
Product Graph

- The direct product $G_\times = (V_\times, E_\times, \phi_\times)$ of G and G':

$$V_\times = \{\, (v, v') \in V \times V' \mid \phi(v) = \phi'(v') \,\},$$

$$E_\times = \left\{ \; ((u, u'), (v, v')) \in V_\times \times V_\times \;\middle|\; \begin{array}{l} (u, v) \in E,\ (u', v') \in E', \\ \phi(u, v) = \phi'(u', v') \end{array} \right\}$$

  - All labels are inherited



k-Step Random Walk Kernal

- The k-step (fixed-length-k) random walk kernel between G and G':

$$K_\times^k(G, G') = \sum_{i,j=1}^{|V_\times|} \left[\lambda_0 A_\times^0 + \lambda_1 A_\times^1 + \lambda_2 A_\times^2 + \cdots + \lambda_k A_\times^k\right]_{ij} \quad (\lambda_l > 0)$$

  - $A_\times$: The adjacency matrix of the product graph
  - The ij entry of $A_\times^n$ shows the number of paths from i to j

Geometric Random Walk Kernel

- $K_{\times}^{\infty}$ can be directly computed if $\lambda_{\ell} = \lambda^{\ell}$ for each $\ell \in \{0, \ldots, k\}$ (geometric series), resulting in the geometric random walk kernel:

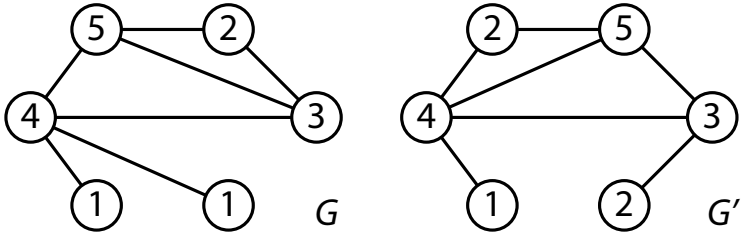
$$K_{\mathrm{GR}}(G, G') = \sum_{i,j=1}^{|V_{\times}|} \left[\lambda^0 A_{\times}^0 + \lambda^1 A_{\times}^1 + \lambda^2 A_{\times}^2 + \cdots\right]_{ij} = \sum_{i,j=1}^{|V_{\times}|} \left[\sum_{\ell=0}^{\infty} \lambda^{\ell} A_{\times}^{\ell}\right]_{ij}$$

$$= \sum_{i,j=1}^{|V_{\times}|} \left[(\mathbf{I} - \lambda A_{\times})^{-1}\right]_{ij}$$

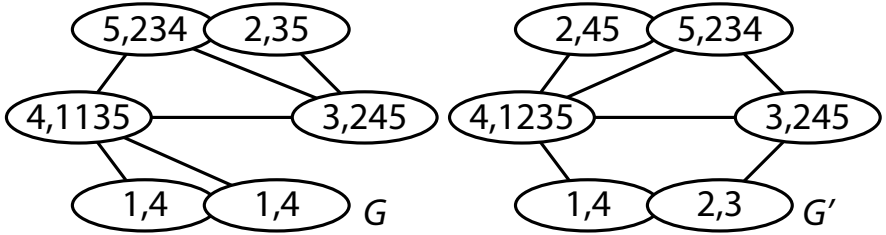
  - Well-defined only if $\lambda < 1/\mu_{\times,\max}$ ($\mu_{\times,\max}$ is the max. eigenvalue of $A_{\times}$)
  - $\delta_{\times}$ (min. degree) $\leq \overline{d}_{\times}$ (average degree) $\leq \mu_{\times,\max} \leq \Delta_{\times}$ (max. degree)

Weisfeiler–Lehman Kernel

Given graphs



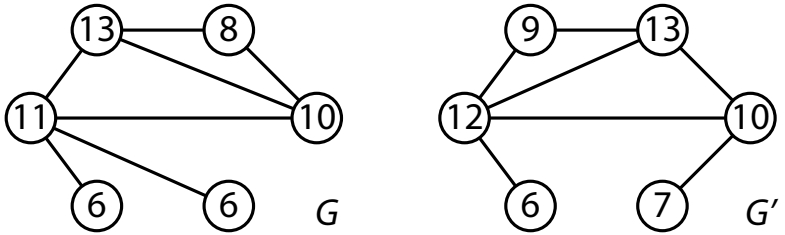
1st iteration



Re-labeling after 1st iteration

1,4	→ 6	3,245	→ 10
2,3	→ 7	4,1135	→ 11
2,35	→ 8	4,1235	→ 12
2,45	→ 9	5,234	→ 13

After 1st iteration



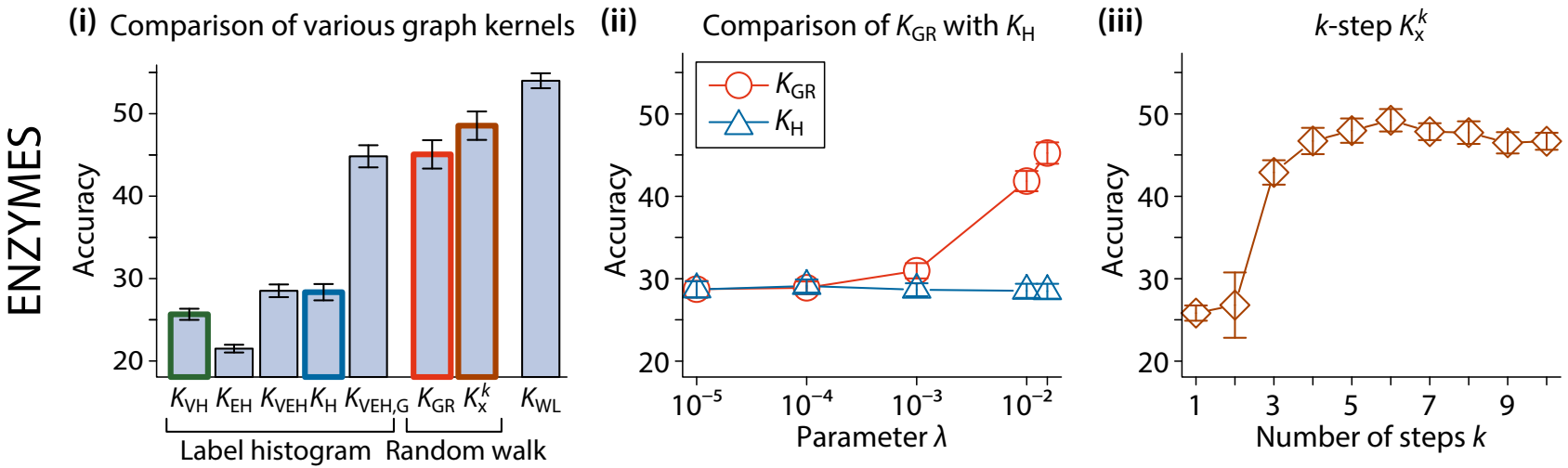
Weisfeiler–Lehman Kernel

- The kernel value becomes:

$$\begin{bmatrix} \text{label} \\ \phi(G)^{(1)} \\ \phi(G')^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 2 & 1 & 1 & 1 & 1 & 2 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix},$$

$$K^1_{\text{WL}}(G, G') = 11$$

Performance Comparison



graphkernels Package

- A package for graph kernels available in R and Python
- R:
  https://CRAN.R-project.org/package=graphkernels
- Python:
  https://pypi.org/project/graphkernels/
- Paper:
  https://doi.org/10.1093/bioinformatics/btx602

Summary

- SVM finds the “best” classification hyperplane
  - The margin is maximized
- Although the original SVM can perform only linear classification, it can be extended to nonlinear classification for structured data using kernels
- Gaussian kernel + C-SVM can be the first choice for numerical data
- WL kernel can be the first choice for graph data
  - It is even as powerful as graph neural networks