November 16, 2022

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

Machine Learning for Graph Structured Data

Introduction to Big Data Science (ビッグデータ概論)

Mahito Sugiyama (杉山麿人)

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... $\rightarrow$ What are succeeding numbers?

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... $\rightarrow$ What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... → What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$

  1, 2, 4, 7, 12, 20, ... $(a_n = a_{n-1} + a_{n-2} + 1)$

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... → What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$
  1, 2, 4, 7, 12, 20, ... $(a_n = a_{n-1} + a_{n-2} + 1)$
  1, 2, 4, 7, 13, 24, ... $(a_n = a_{n-1} + a_{n-2} + a_{n-3})$

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... → What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$
  1, 2, 4, 7, 12, 20, ... $(a_n = a_{n-1} + a_{n-2} + 1)$
  1, 2, 4, 7, 13, 24, ... $(a_n = a_{n-1} + a_{n-2} + a_{n-3})$
  1, 2, 4, 7, 14, 28 (divisors of 28)

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... → What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$
  1, 2, 4, 7, 12, 20, ... $(a_n = a_{n-1} + a_{n-2} + 1)$
  1, 2, 4, 7, 13, 24, ... $(a_n = a_{n-1} + a_{n-2} + a_{n-3})$
  1, 2, 4, 7, 14, 28 (divisors of 28)
  1, 2, 4, 7, 1, 1, 5, ... ($\pi = 3.1415\ldots$ and $e = 2.718\ldots$)

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7,... → What are succeeding numbers?

  1, 2, 4, 7, 11, 16, ... $(a_n = a_{n-1} + n - 1)$
  1, 2, 4, 7, 12, 20, ... $(a_n = a_{n-1} + a_{n-2} + 1)$
  1, 2, 4, 7, 13, 24, ... $(a_n = a_{n-1} + a_{n-2} + a_{n-3})$
  1, 2, 4, 7, 14, 28 (divisors of 28)
  1, 2, 4, 7, 1, 1, 5, ... ($\pi = 3.1415\ldots$ and $e = 2.718\ldots$)

- 1344 results (!) in the online encyclopedia (`https://oeis.org/`)

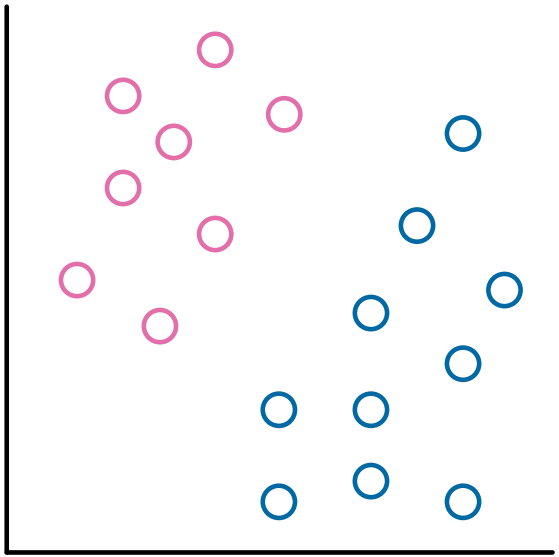
Learning as Scientific Problem

- Which is the correct answer (or generalization) for succeeding numbers of $1, 2, 4, 7, \ldots$?
  - Any answer is possible!

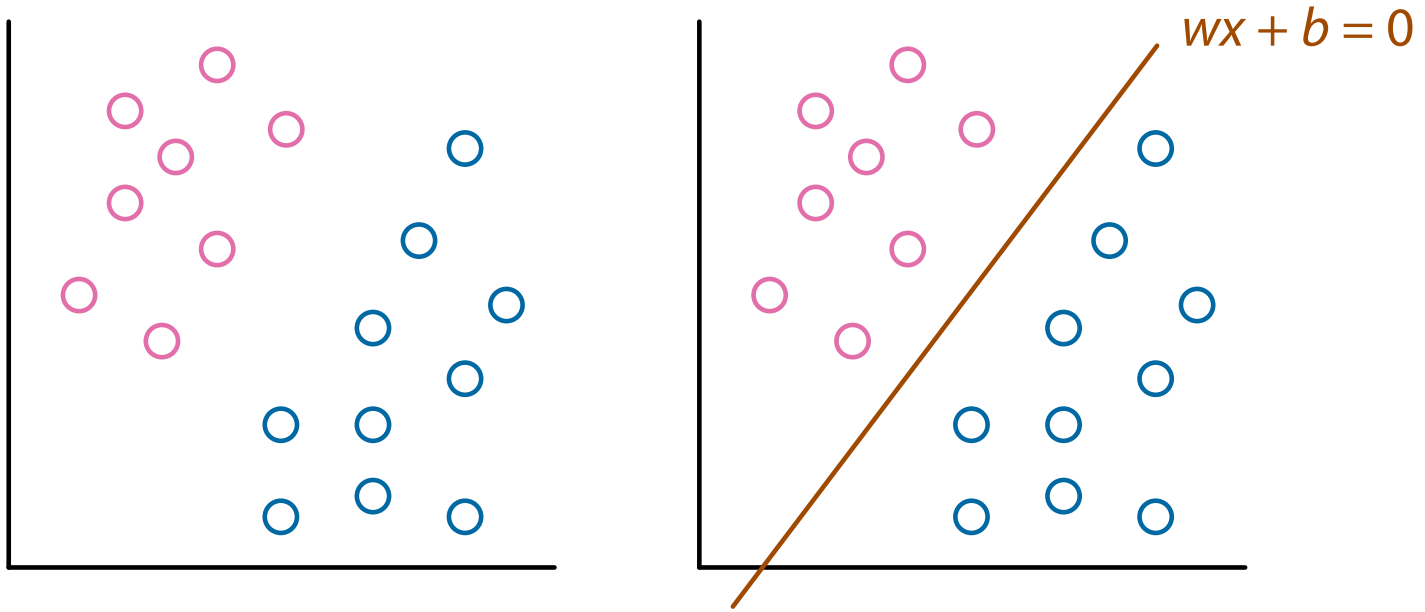
Learning as Scientific Problem

- Which is the correct answer (or generalization) for succeeding numbers of $1, 2, 4, 7, \ldots$?
  - Any answer is possible!
- We should take two points into consideration:
  - (i) We need to formalize the problem of “learning”
    - There are two agents (teacher and learner) in learning, which are different from “computation”
  - (ii) Learning is an infinite process

Learning of Binary Classifier



Learning of Binary Classifier



Example: Perceptron (by F. Rosenblatt, 1958)

- **Learning target**: two subsets $F, G \subseteq \mathbb{R}^d$ s.t. $F \cap G = \emptyset$
  - Assumption: F and G are linearly separable:
    There exists a function (classifier) $f_*(\boldsymbol{x}) = \langle \boldsymbol{w}_*, \boldsymbol{x} \rangle + b$ s.t.
    $f_*(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in F, \qquad f_*(\boldsymbol{x}) < 0 \quad \forall \boldsymbol{x} \in G$

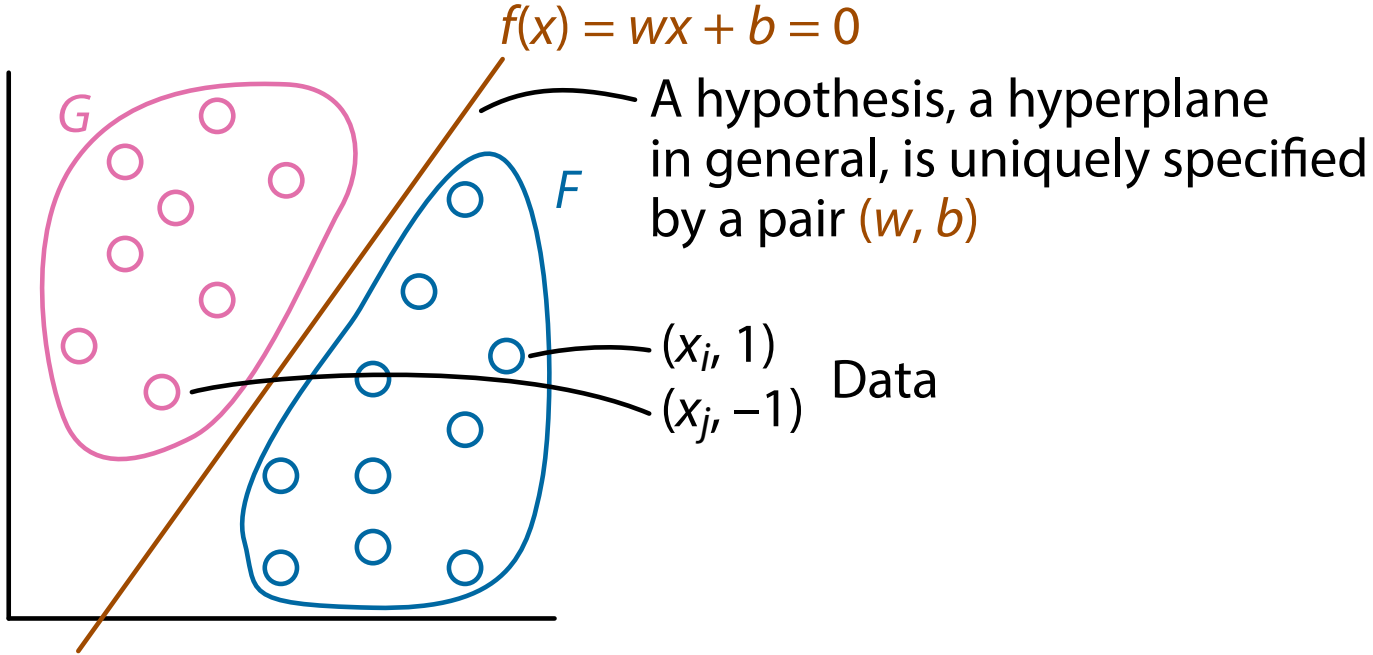
Example: Perceptron (by F. Rosenblatt, 1958)

- **Learning target**: two subsets $F, G \subseteq \mathbb{R}^d$ s.t. $F \cap G = \emptyset$
  - Assumption: F and G are linearly separable:
    There exists a function (classifier) $f_*(\boldsymbol{x}) = \langle \boldsymbol{w}_*, \boldsymbol{x} \rangle + b$ s.t.
    $f_*(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in F, \qquad f_*(\boldsymbol{x}) < 0 \quad \forall \boldsymbol{x} \in G$
- **Hypotheses**: hyperplanes on $\mathbb{R}^d$
  - If we consider a linear equation $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$,
    each line can be uniquely specified by a pair of
    two parameters $(\boldsymbol{w}, b)$ (hypothesis)

Example: Perceptron (by F. Rosenblatt, 1958)

- **Data**: a sequence of pairs $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots$
  - $(\boldsymbol{x_i}, y_i)$: (a real-valued vector in $\mathbb{R}^d$, a label)
  - $\boldsymbol{x}_i \in F \cup G$, $y_i \in \{1, -1\}$,
    $y_i = 1$ ($y_i = -1$) if $\boldsymbol{x}_i \in F$ ($\boldsymbol{x}_i \in G$)

Learning Model for Perceptron



Learning Procedure of Perceptron

1. $\boldsymbol{w} \leftarrow 0, b \leftarrow 0$ (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 $a = \sum_{j=1}^{d} w^j x_i^j + b$
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. $b \leftarrow b + y_i$ // update the bias
8. end if
9. end for

Correctness of Perceptron

- It is guaranteed that a perceptron always converges to a correct classifier
  - A correct classifier is a function f s.t.
    $$f(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in F,$$
    $$f(\boldsymbol{x}) < 0 \quad \forall \boldsymbol{x} \in G$$
  - The convergence theorem
- Note: there are (infinitely) many functions that correctly classify F and G
  - A perceptron converges to one of them

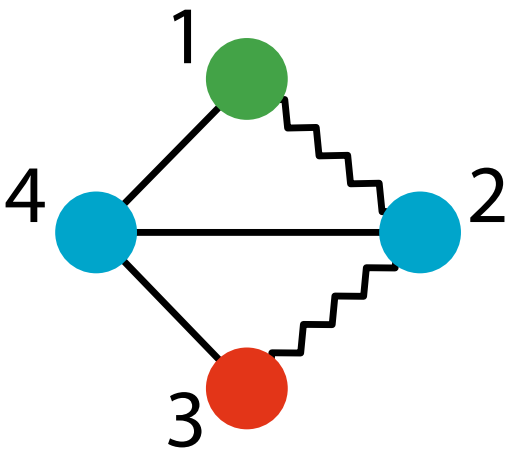
Summary: Perceptron

Target	Two disjoint subsets of $\mathbb{R}^d$
Representation	Two parameters $(\boldsymbol{w}, b)$ of linear equation $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$
Data	Real vectors from target subsets
Algorithm	Perceptron
Correctness	Convergence theorem

What is Graph?

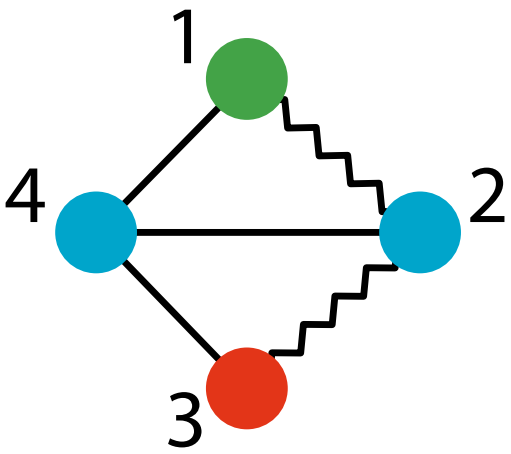
- A graph is an object consisting of vertices (nodes) connected with edges
  - Many examples in real-world, e.g., chemical compounds
- A graph is directed if the edges are directed, otherwise it is undirected
- A graph is written as $G = (V, E)$, where V is a vertex set and E is an edge set
- Labels can be associated with vertices and/or edges

Example of Graph



- A graph $G = (V, E, \phi)$
  - $V = \{1, 2, 3, 4\}$
  - $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$
  - $\phi(1) =$ green, $\phi(2) =$ blue,
    $\phi(3) =$ red, $\phi(4) =$ blue
  - $\phi(\{1, 2\}) =$ zigzag, $\phi(\{1, 4\}) =$ straight,
    $\phi(\{2, 3\}) =$ zigzag, $\phi(\{2, 4\}) =$ straight,
    $\phi(\{3, 4\}\}) =$ straight

Example of Graph



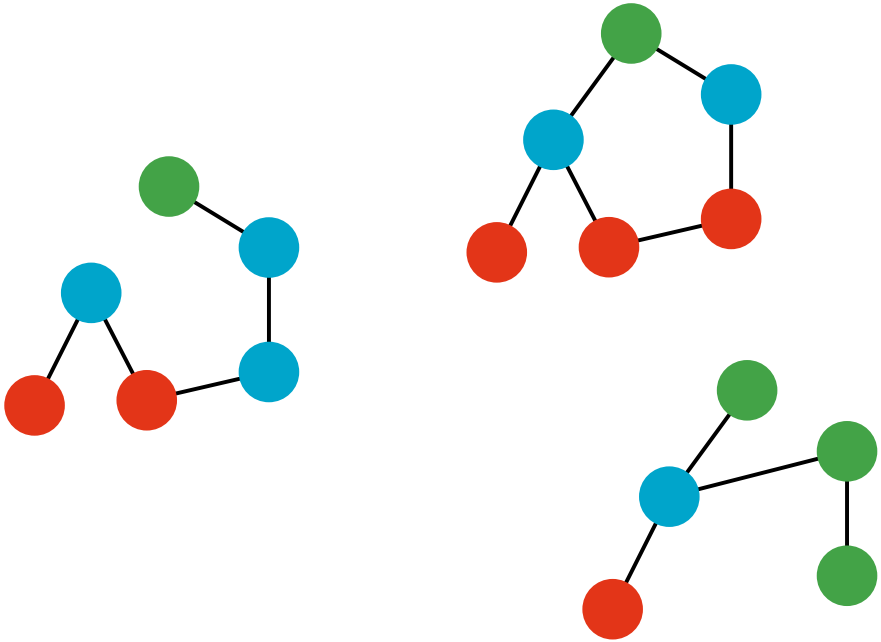
- The adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

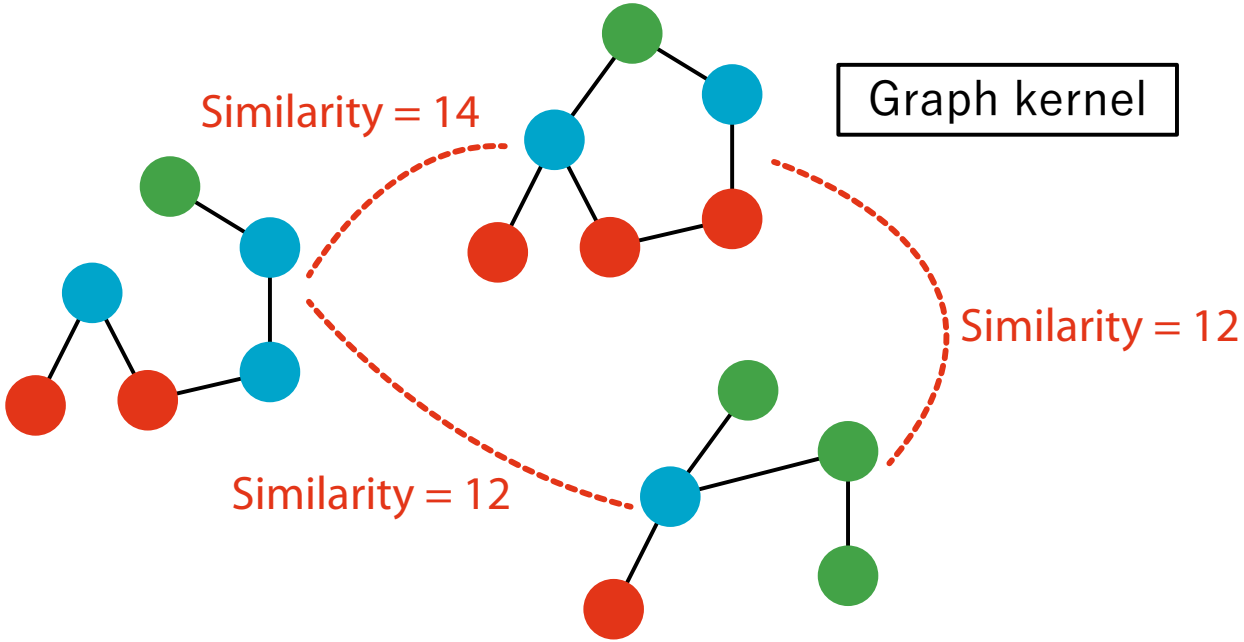
ML on Graphs

- How to perform machine learning on graphs?
  - Each object is a graph, so we have a collection of graphs
- Classification (or regression) on graphs is nontrivial problem
  - The difficulty comes from the fact that measuring the similarity (or distance) between graphs is nontrivial
- Graph kernel computes the similarity between graphs
- Graph Neural Networks are recently studied, while there is no significant difference between their performances
  - They share the core idea (message passing)

Similarity between Graphs



Similarity between Graphs



Kernels on Structured Data

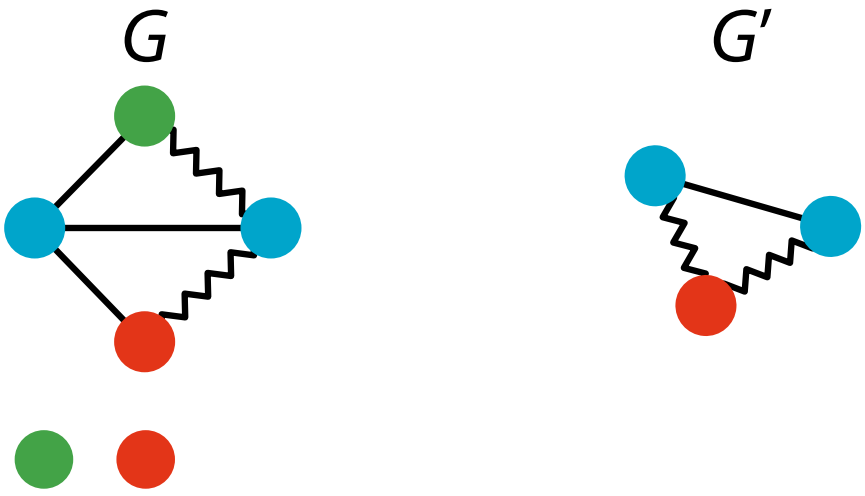
- Given objects X and Y, decompose them into substructures S and T
- The R-convolution kernel K_R by Haussler (1999) is:

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

  - e.g. X is a graph and S is the set of all subgraphs
- Since naïvely computing this kernel is expensive, many efficient graph kernels have been propsoed

Example

Vertex Label Histogram Kernel



G	2	1	1
G'	2	0	1

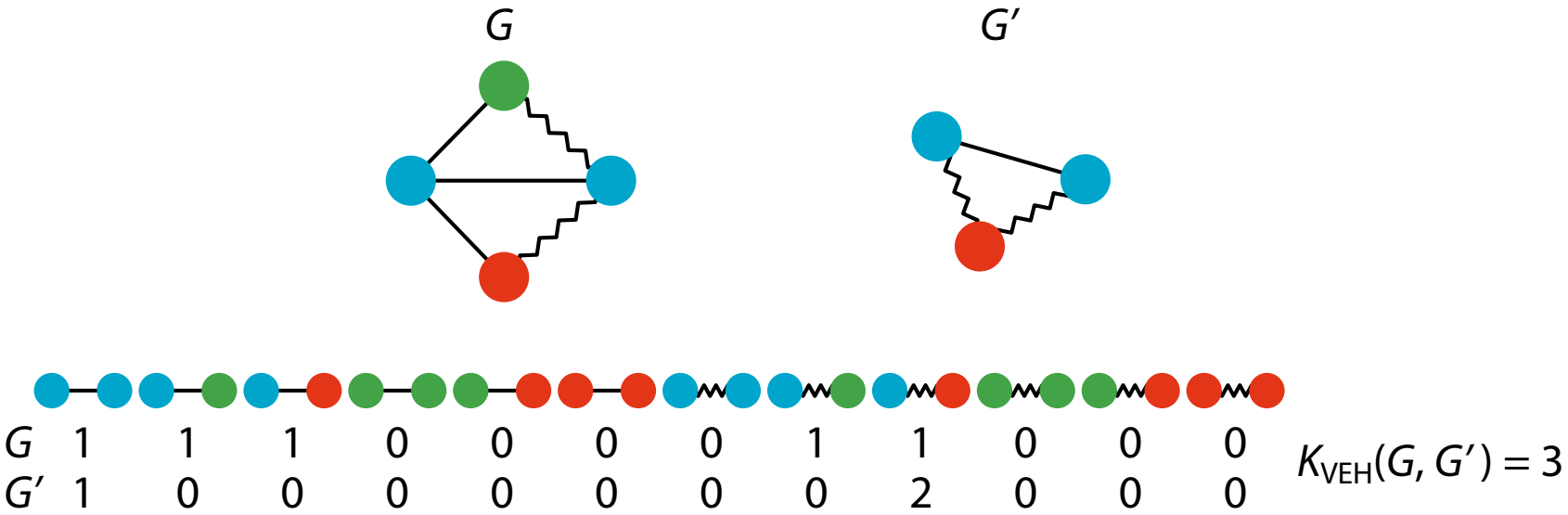
$$K_{VH}(G, G') = 2 \cdot 2 + 1 \cdot 0 + 1 \cdot 1 = 5$$

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

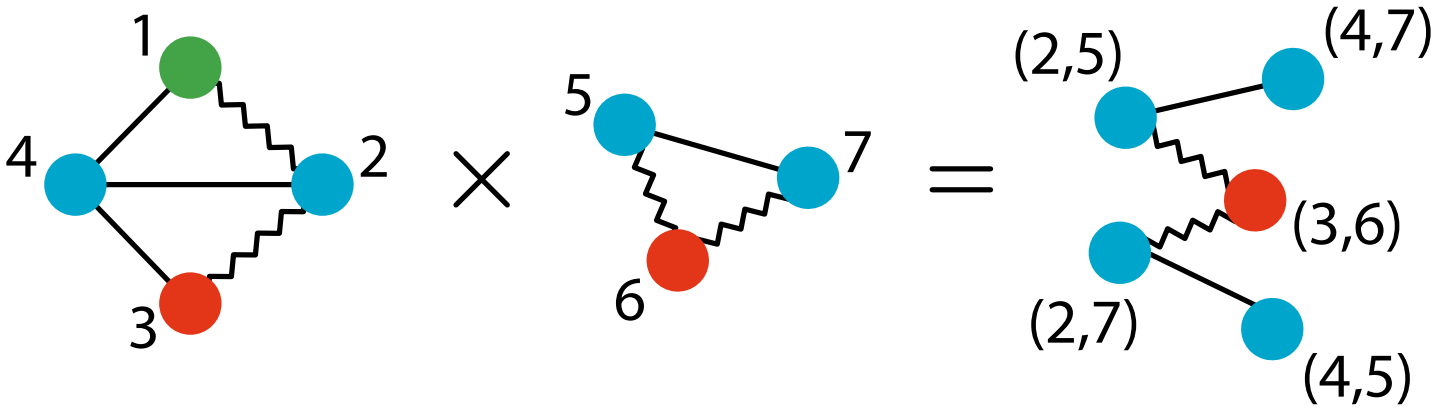
- The direct product $G_\times = (V_\times, E_\times, \phi_\times)$ of $G = (V, E, \phi)$, $G' = (V', E', \phi')$:

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

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

  - All labels are inherited

Example of Product Graph



k-Step Random Walk Kernal

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

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

$$(\lambda_l > 0)$$

  - $A_{\times}$: 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 (1/2)

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

$$K_{\mathrm{GR}}(G, G') = \sum_{i,j=1}^{|V_\times|} \left[\lambda^0 A_\times^0 + \lambda^1 A_\times^1 + \lambda^2 A_\times^2 + \lambda^3 A_\times^3 + \cdots\right]_{ij}$$

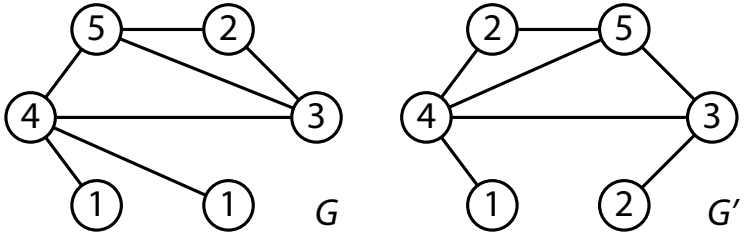
$$= \sum_{i,j=1}^{|V_\times|} \left[\sum_{\ell=0}^{\infty} \lambda^\ell A_\times^\ell\right]_{ij} = \sum_{i,j=1}^{|V_\times|} \left[(\mathbf{I} - \lambda A_\times)^{-1}\right]_{ij}$$

Geometric Random Walk Kernel (2/2)

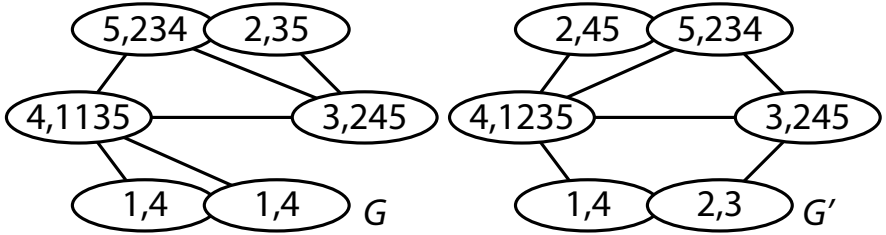
- Geometric random walk kernel is well-defined only if $\lambda < 1/\mu_{\times,\max}$ ($\mu_{\times,\max}$ is the max. eigenvalue of $A_\times$)
- $\delta_\times$ (min. degree) $\leq \bar{d}_\times$ (average degree) $\leq \mu_{\times,\max} \leq \Delta_\times$ (max. degree)

Weisfeiler–Lehman Kernel

Given graphs



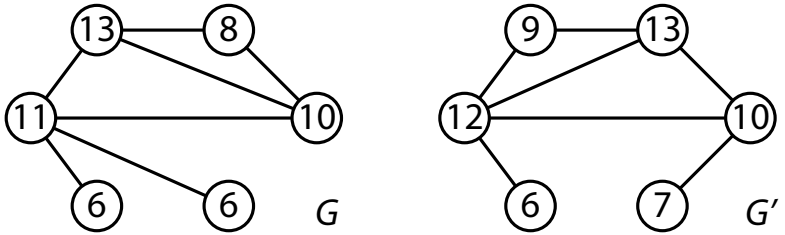
1st iteration



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

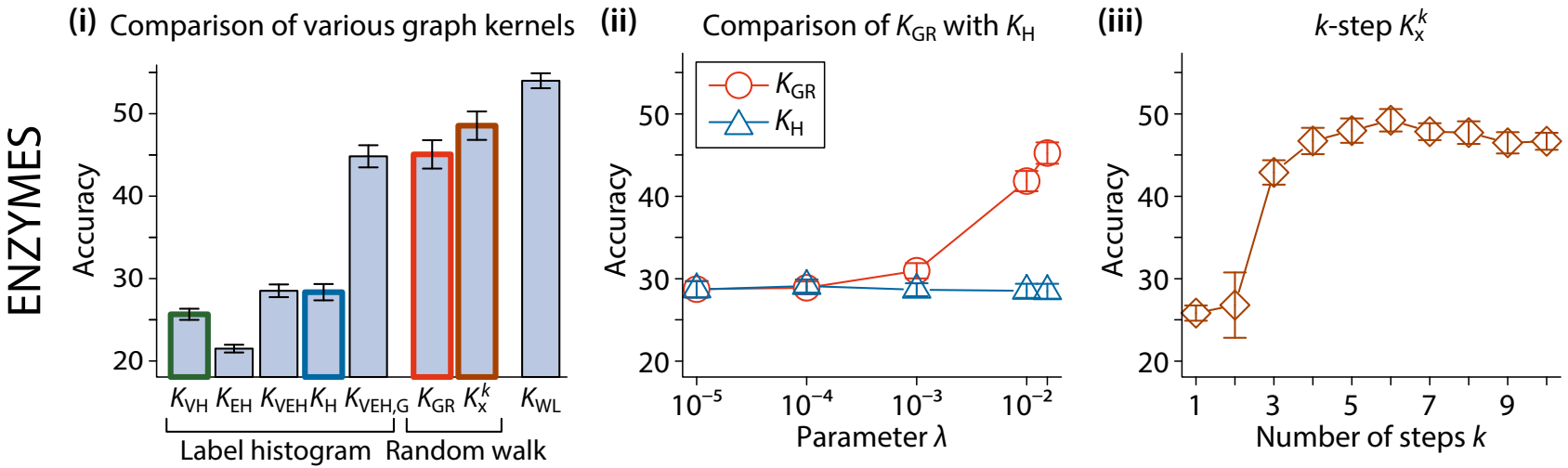
- The kernel value becomes:

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

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

- An important building block of GNNs

Performance Comparison



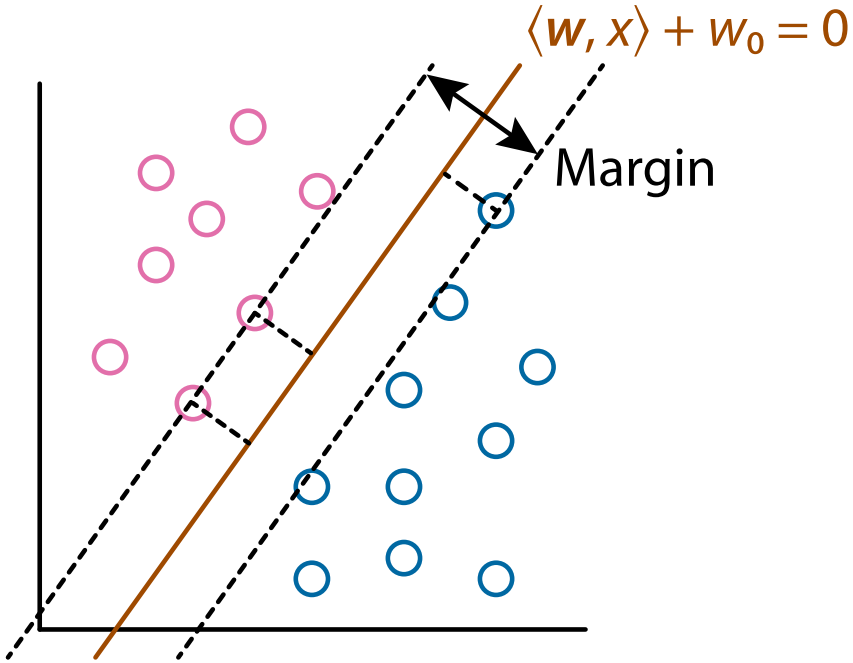
graphkernels Package

- 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

Kernel-based Classification: SVM

- A dataset D is separable by $f \iff y_i f(\boldsymbol{x}_i) > 0, \forall i \in \{1, 2, \dots, n\}$
- 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
  - Can be viewed as an extension of perceptron

Margin



Formulation of SVMs

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

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

- The margin maximization problem can be written as

$$\max_{\boldsymbol{w}, w_0, M} \frac{M}{\|\boldsymbol{w}\|} \quad \text{subject to } y_i f(\boldsymbol{x}_i) \geq M, i \in \{1, 2, \dots, n\}$$

  - $M = \min_{i \in \{1,2,\dots,n\}} |\langle \boldsymbol{w}, x_i \rangle + w_0|$

Hard Margin SVMs

- We can eliminate M and obtain

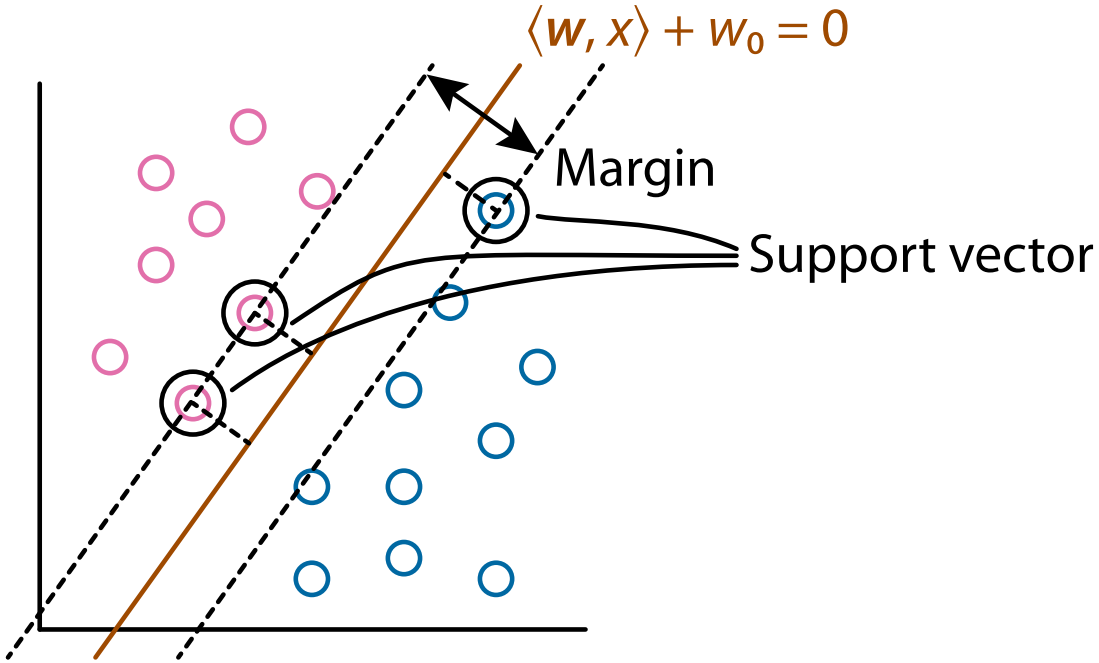
  $$\max_{\boldsymbol{w},w_0} \frac{1}{\|\boldsymbol{w}\|} \quad \text{subject to } y_i f(\boldsymbol{x}_i) \geq 1, i \in \{1, 2, \dots, n\}$$

- This is equivalent to

  $$\min_{\boldsymbol{w},w_0} \|\boldsymbol{w}\|^2 \quad \text{subject to } y_i f(\boldsymbol{x}_i) \geq 1, i \in \{1, 2, \dots, n\}$$

  - The standard formulation of hard margin SVMs
  - There are data points x_i satisfying $y_i f(\boldsymbol{x}_i) = 1$, called support vectors

Margin



Soft Margin (1/2)

- Datasets are not often separable
- Extend SV classification to soft margin by relaxing $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 \geq 1$
- Change the constraint $y_i f(\boldsymbol{x}_i) \geq 1$ using the slack variable ξ_i to

$$y_i f(\boldsymbol{x}_i) = y_i \left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 \right) \geq 1 - \xi_i, \quad i \in \{1, 2, \dots, n\}$$

Soft Margin (2/2)

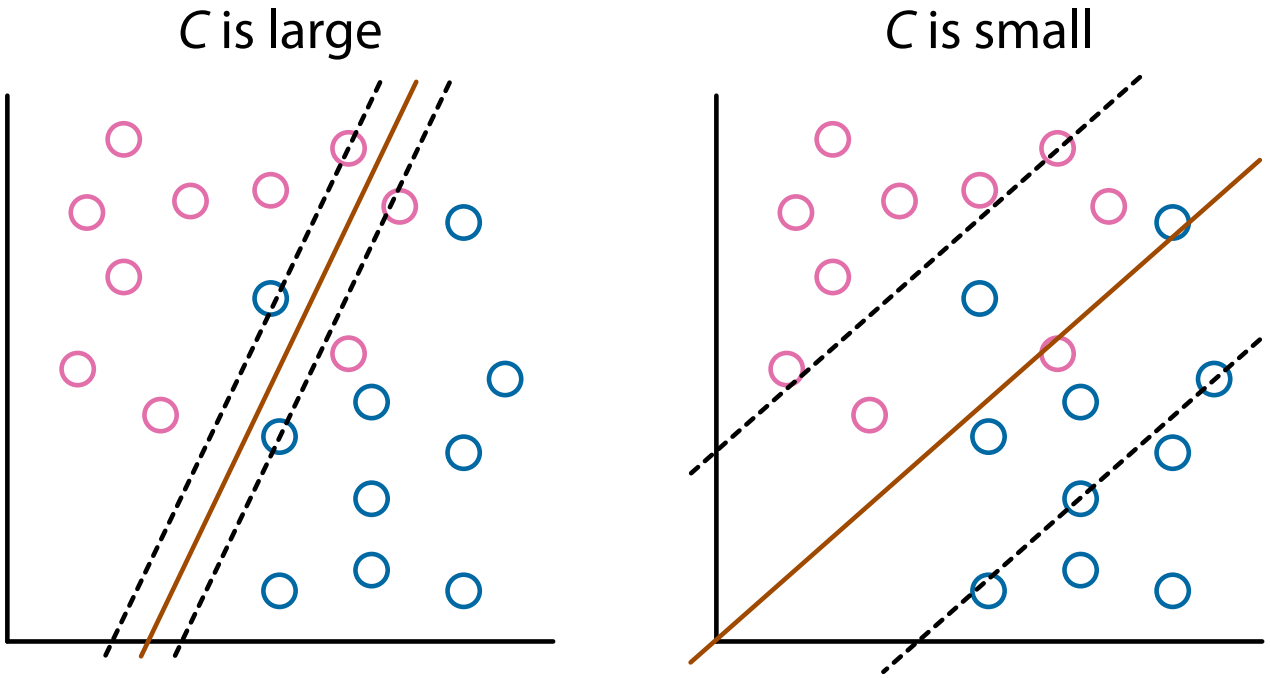
- The formulation of soft margin SVM (C-SVM) is

$$\min_{\boldsymbol{w},w_0,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\{1,2,\ldots,n\}} \xi_i$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0, i \in \{1, 2, \ldots, n\}$$

  - C is called the regularization parameter

Soft Margin



Data Point Location

- $y_i f(\boldsymbol{x}_i) > 1$: $\boldsymbol{x}_i$ 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 on margin and inside margin are support vectors

Dual Problem (1/2)

- The formulation of C-SVM

$$\min_{\boldsymbol{w},w_0,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\{1,2,\ldots,n\}} \xi_i$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0, i \in \{1, 2, \ldots, n\}$$

  is called the primal problem

- This is usually solved via the dual problem

Dual Problem (2/2)

- The dual problem is given as

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

$$\text{s.t.} \sum_{i\in[n]} \alpha_i y_i = 0,\ 0 \le \alpha_i \le C, i \in [n]$$

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

  $$\max_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle + \sum_{i \in [n]} \alpha_i \text{ s.t. } \sum_{i \in [n]} \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C, i \in$$

  - 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

- Using the kernel function K such that $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, we have

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

$$\text{s.t.} \sum_{i \in [n]} \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C, i \in [n]$$

  - The technique of using K is called kernel trick

- We can use graph kernels for K!

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

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

    is positive (semi-)definite, that is, $\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$ for any $c_1, c_2, \dots, c_n \in \mathbb{R}$
  - $(K_{ij}) \in \mathbb{R}^{n \times n}$ is called the Gram matrix

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

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

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

Summary

- Introduction to ML and its basic concepts
- Graph kernels for graph structured data
- Kernel-based ML methods, such as SVM
  - There are many other options, e.g. kernel PCA, kernel k-means, kernel ridge regression, ...