CS480/680: Introduction to Machine Learning

Homework 5

Due: 11:59 pm, June 15, 2020, submit on LEARN.

Include your name and student number!

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

Exercise 1: Graph Kernels (10 pts)

One cool way to construct a new kernel from an existing set of (base) kernels is through graphs. Let $\mathcal{G} = (V, E)$ be a directed acyclic graph (DAG), where V denotes the nodes and E denotes the arcs (directed edges). For convenience let us assume there is a source node s that has no incoming arc and there is a sink node t that has no outgoing arc. We put a base kernel κ_e (that is, a function $\kappa_e : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$) on each arc $e = (u \to v) \in E$. For each path $P = (u_0 \to u_1 \to \cdots \to u_d)$ with $u_{i-1} \to u_i$ being an arc in E, we can define the kernel for the path P as the product of kernels along the path:

$$\forall \mathbf{x}, \mathbf{z} \in \mathcal{X},\ \kappa_P(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{d} \kappa_{u_{i-1} \to u_i}(\mathbf{x}, \mathbf{z}). \tag{1}$$

Then, we define the kernel for the graph $\mathcal{G}$ as the sum of all possible $s \to t$ path kernels:

$$\forall \mathbf{x}, \mathbf{z} \in \mathcal{X},\ \kappa_{\mathcal{G}}(\mathbf{x}, \mathbf{z}) = \sum_{P \in \text{path}(s \to t)} \kappa_P(\mathbf{x}, \mathbf{z}). \tag{2}$$

1. (1 pt) Prove that $\kappa_{\mathcal{G}}$ is indeed a kernel.
2. (1 pt) Let $\mathbf{x} = 1$ and $\mathbf{z} = -1$. Compute the kernel value $\kappa_{\mathcal{G}}(1, -1)$ for the graph in Figure 1.
3. (1 pt) Let $\kappa_i, i = 1, \ldots, n$ be a set of given kernels. Construct a graph $\mathcal{G}$ (with appropriate base kernels) so that the graph kernel $\kappa_{\mathcal{G}} = \sum_{i=1}^{n} \kappa_i$.
4. (1 pt) Let $\kappa_i, i = 1, \ldots, n$ be a set of given kernels. Construct a graph $\mathcal{G}$ (with appropriate base kernels) so that the graph kernel $\kappa_{\mathcal{G}} = \prod_{i=1}^{n} \kappa_i$.
5. (1 pt) Consider the subgragh in Figure 1 that includes nodes s, a, b, c (and arcs connecting them). Compute the graph kernel where s and c play the role of source and sink, respectively.
6. (1 pt) Consider the subgragh in Figure 1 that includes nodes s, a, b, c, d (and arcs connecting them). Compute the graph kernel where s and d play the role of source and sink, respectively.
7. (4 pts) Find an efficient algorithm to compute the graph kernel $\kappa_{\mathcal{G}}(\mathbf{x}, \mathbf{z})$ (for two fixed inputs $\mathbf{x}$ and $\mathbf{z}$) in time $O(|V| + |E|)$, assuming each base kernel κ_e costs $O(1)$ to evaluate. You may assume there is always at least one $s - t$ path. State and justify your algorithm is enough; no need (although you are encouraged) to give a full pseudocode.

   [Note that the total number of paths in a DAG can be exponential in terms of the number of nodes $|V|$, so naive enumerating would not work. For example, replicating the intermediate nodes in Figure 1 n times creates 2^n paths from s to t.]

   [Hint: Recall that we can use topological sorting to rearrange the nodes in a DAG such that all arcs go from a "smaller" node to a "bigger" one.]

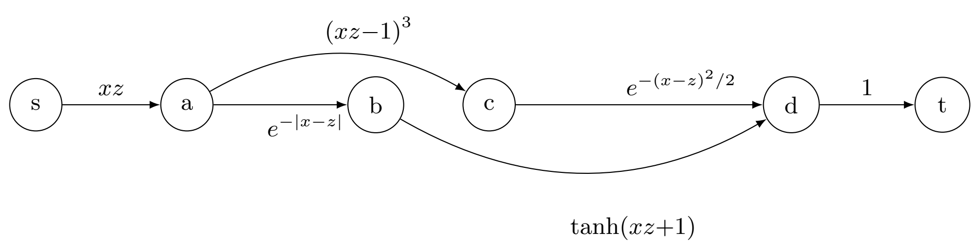

Figure 1: DAG with base kernels. The constant 1 denotes the trivial kernel $\kappa(x, z) \equiv 1$.