CS 6780: Advanced Machine Learning **(Due: 03/28/19)**

Assignment #3: Kernels, Kernelization, and Neural networks

Instructor: Thorsten Joachims *Name:* Student name(s), *Netid:* NetId(s)

***Course Policy**: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.*

- *Please typeset your submissions in LaTeX. Use the template provided for your answers. Please include your names and NetIDs on the first page of your submission, as well as date and time.*
- *Assignments are due at 2:55 pm on the due date on **Gradescope** for PDF submissions and CMS for code submission.*
- *Late assignments can be submitted up to Sunday, March 31. This is also when the solutions will be released.*
- *You can do this assignment in groups of 2. Please submit no more than one submission per group.*
- *All sources of material must be cited. The University Academic Code of Conduct will be strictly enforced.*

Problem 1 (10+6+4+10+0=30 points)

Objectives: Understand the definition of kernels. Learn different ways to construct kernels (explicit feature maps, positive semi-definite Gram matrices, and compatible composition of other kernels). Apply kernels to non-vectorial data.

Beyond using kernels to achieve non-linear decision surfaces, one of the key advantages of using kernels is that they can be computed even for non-vectorial objects. In this problem we will analyze kernels over strings. Intuitively, $K(x_i, x_j)$ indicates the similarity between two strings x_i and x_j. The strings are finite length, and consist of characters from an alphabet Σ. $|\Sigma| = a \geq 2$.

(a) Consider the kernel:

$$K_d(x_i, x_j) = \text{ Number of strings of length exactly } d \text{ that } x_i,\ x_j \text{ have in common.}$$

Show that K_d is a valid kernel for any $d > 0$ by constructing an explicit feature map $\phi_d(\cdot)$, such that $K_d(x_i, x_j) = \phi_d(x_i)\phi_d(x_j)$. How does the minimum required dimensionality of $\phi_d(\cdot)$ change as a function of d?

(b) Often, we can compute kernels efficiently even though the feature map ϕ_d may map into an extremely high-dimensional space (e.g., exponential in the size of the inputs x). For the string kernel defined above, create an algorithm that computes the value of $K_d(x_i, x_j)$ for two strings of length at most $n > d$ that has better than exponential running time and space complexity. Clearly show why your algorithm will have sub-exponential complexity in n and d.

(c) We can also construct kernels via composition, and these can usually be more compactly computed than their corresponding feature vector dot-products too. For $D > 1$, define

$$K(x_i, x_j) = \prod_{d=1}^{D} K_d(x_i, x_j).$$

Suppose you constructed an explicit feature map ϕ for this kernel. How does the minimum required dimensionality of $\phi(\cdot)$ change as a function of D? Can your string-kernel algorithm from above be used for computing $K(x_i, x_j)$ so that it runs in time sub-exponential in D? State yes/no and justify in one sentence or less.

(d) Suppose we instead defined a similarity function

$$K'_d(x_i, x_j) = \begin{cases} 1, & \text{if } x_i, x_j \text{ share at least one substring of length exactly d.} \\ 0, & \text{otherwise.} \end{cases}$$

Show that K'_d is **not** a valid kernel for any $d > 0$ and length of $x_i, x_j \geq d$ by constructively proving the Gram matrix is not always positive semi-definite.

(e) Brain Teaser (Optional, No Credit): Let us now define kernels where the x are scalars from the set of natural numbers $\mathcal{N} = \{1, 2, \ldots\}$. For each of the following functions, state whether it is a valid kernel, and provide a detailed proof substantiating your answer.
Hint: Let $p_1, p_2, \ldots$ be the sequence of prime numbers starting at 2. Then let $f(n)_i$ be the frequency of prime p_i in the prime factorization of n. We can write the gcd as $gcd(m, n) = \prod_i p_i^{\min(f(m)_i, f(n)_i)}$.

- $K(x_i, x_j) = lcm(x_i, x_j)$ (least common multiple).
- $K(x_i, x_j) = gcd(x_i, x_j)$ (greatest common divisor).

Problem 2 (10+15+15=40 points)

Objectives: To understand how to kernelize known linear classifier algorithms

In this problem, we will get some hands-on experience with kernelizing three known linear classification algorithms– LDA, Ridge Regression and Regularized Logistic regression. The basic idea is to express the cost function and the classifier in terms of the inner product between data points, and then replace these inner products with kernel values $K(x_i, x_j)$.

Note that none of these algorithms would require you to explicitly compute the mapping $x \mapsto \phi(x)$ since it can be an expensive operation as we saw in the last problem.

(a) Linear Discriminant Analysis (LDA): In Linear Discriminant Analysis, let us consider the following assumption:
Assume that the data from each class is distributed normally around the average of the data points with unit variance i.e.

$$P(X = x|Y = +1) \sim \exp\left(-\frac{1}{2}(x - \mu_+)^2\right)$$
$$P(X = x|Y = -1) \sim \exp\left(-\frac{1}{2}(x - \mu_-)^2\right),$$

where the $\mu_+ = \frac{1}{n_+}\sum_{y_i=+1} x_i$, $\mu_- = \frac{1}{n_-}\sum_{y_i=-1} x_i$ are the averages of the datapoints in each class. Given the dataset, one can also estimate the priors of each of the classes $P(Y = +1)$, and $P(Y = -1)$. Using the above, we define the following classification rule:

$$h_{\text{LDA}} = \underset{y \in \{+1,-1\}}{\arg\max} \left(P(Y = y) \exp(-\frac{1}{2}(x - \mu_y)^2\right)$$

Kernelize the LDA classification rule to obtain a form that doesn't require computation of explicit feature maps by expressing it in terms of inner products and replacing the inner products as kernel function.

(b) Ridge Regression: To train a ridge regression we use the following training objective:

$$\hat{w} = \arg\min_w \left[\frac{1}{2} w \cdot w + C \sum_{i=1}^{N} (w \cdot x_i - y_i)^2\right]$$

The closed form solution of this objective in matrix form is:

$$\hat{w} = ((1/C)I + X^T X)^{-1} X^T y,$$

where I is the identity matrix of size $D \times D$, X is a matrix of size $N \times D$ containing all the training data points, y is the vector of labels, with D being the dimensionality of the feature vector and N being the size of the dataset. Hence for a x_{test}, the prediction is done as follows:

$$\hat{y}_{test} = x_{test}((1/C)I + X^T X)^{-1} X^T y$$

Now, show the following:

- Prove the representer theorem for Ridge regression i.e. $\hat{w}$ can be written as a combination of the data X, $\hat{w} = X^T A$ for some vector A.

  Hint: Use the property $(PQ + I_N)^{-1}P = P(QP + I_M)^{-1}$ *where* P *is an* $N \times M$ *matrix and* Q *is an* $M \times N$ *matrix.*

- Kernelize the closed form prediction expression for ridge regression above in terms of the kernel funciton $k(x, x')$.

(c) Regularized Logistic Regression: The training objective of regularized logistic regression is as follows:

$$\hat{w} = \arg\min_{w} \left[\frac{1}{2} w \cdot w + C \sum_{i=1}^{N} \log(1 + \exp(-y_i w \cdot x_i)) \right]$$

For the objective above, prove that the Representer Theorem i.e. for the $\hat{w}$ that optimizes the objective above, prove that $\hat{w}$ can be written as a sum of $\alpha_i x_i$ for some value of α_i. Clearly specify the values α_i which may depend on $\hat{w}$. Further, using this result, kernelize the objective function and express it in terms of optimizing for α_i as the variables. Hint: Use that this is a strictly convex objective, such that the derivative is zero only at the optimum $\hat{w}$.

Problem 3 (30 points)

Objectives: To get a hands-on experience with a deep learning library all the way from model construction and loss functions to training and evaluating.

To attempt this problem, you will use the Jupyter notebook provided in the zip file accompanying the assignment. To open the notebook on your computer, you will need a Jupyter notebook package installed on your computer. To run the code in the notebook, you will require the following packages: `numpy, scikit-learn, torch, matplotlib`. The easiest way to have all of them is to first install the Python 3 version of Anaconda distribution[1], and do `conda install <name>` or `pip install <name>` for the packages mentioned above. For torch, check out the installation instructions specific to your device at: https://pytorch.org/. (Note that if you also have a Python 2.7 installed on your system already and had to install Python/Anaconda 3 parallelly, you might have to run `pip3` rather than `pip` and also make sure your Jupyter notebook is running a Python 3 kernel.) Feel free to search online or ask questions on Piazza if the setup and opening the notebook don't work for you.

The instructions for the subproblems are in the notebook provided.

Submit the Jupyter notebook as a PDF on Gradescope attached to the rest of the assignment. Look for instructions to save as PDF at the bottom of the notebook.

[1] https://www.anaconda.com/distribution/