*COMS 4771 Machine Learning* *Fall 2023*

# Homework 4

*Due MONDAY November 13, 2023 at 11:59 PM*

Instructions for preparing and submitting your homework write-up are available here: `https://www.cs.columbia.edu/~djhsu/coms4771-f23/policies-homework.html`

**Please submit your Python code for Problems 3, 5 and 6 on Gradescope.**

## Kernels and kernel ridge regression

**Problem 1.** Suppose each of $\mathrm{k}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $\tilde{\mathrm{k}}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel function. For each of the following functions, determine whether it must also be a positive definite kernel function or not. (Answer with "POSITIVE DEFINITE" or "NOT NECESSARILY POSITIVE DEFINITE".) Give a brief justification of your answer.

(a) $\mathrm{k_a}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by $\mathrm{k_a}(x, z) = \mathrm{k}(x, z) + 2$.

(b) $\mathrm{k_b}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by $\mathrm{k_b}(x, z) = 2\,\mathrm{k}(x, z)$.

(c) $\mathrm{k_c}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by $\mathrm{k_c}(x, z) = \mathrm{k}(x, z) + \tilde{\mathrm{k}}(x, z)$.

(d) $\mathrm{k_d}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by $\mathrm{k_d}(x, z) = \mathrm{k}(x, z) - \tilde{\mathrm{k}}(x, z)$.

(e) (*Optional.*) $\mathrm{k_e}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by $\mathrm{k_e}(x, z) = \mathrm{k}(x, z) \cdot \tilde{\mathrm{k}}(x, z)$.

*Hint*: look up and use Hadamard product and Schur product theorem.

**Problem 2.** Explain why the following identity holds for any $n \times p$ matrix $A$ and any $\lambda > 0$:

$$(A^\intercal A + \lambda I)^{-1} A^\intercal = A^\intercal (A A^\intercal + \lambda I)^{-1}.$$

Let $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$ be training data from $\mathbb{R}^d \times \mathbb{R}$ for a regression problem. *Kernel ridge regression* returns a solution to the following system of linear equations in $\alpha \in \mathbb{R}^n$:

$$(K + \lambda I)\alpha = b,$$

where

$$K = \begin{bmatrix} \mathrm{k}(x^{(1)}, x^{(1)}) & \cdots & \mathrm{k}(x^{(1)}, x^{(n)}) \\ \vdots & \ddots & \vdots \\ \mathrm{k}(x^{(n)}, x^{(1)}) & \cdots & \mathrm{k}(x^{(n)}, x^{(n)}) \end{bmatrix}$$

is the $n \times n$ kernel matrix corresponding to the training data and the kernel function $\mathrm{k}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $b = (y^{(1)}, \ldots, y^{(n)}) \in \mathbb{R}^n$ is the vector of training labels, and $\lambda \in \mathbb{R}$ is the regularization hyperparameter. The predictor $f\colon \mathbb{R}^d \to \mathbb{R}$ associated with $\alpha = (\alpha_1, \ldots, \alpha_n) \in$

$\mathbb{R}^n$ is given by

$$f(x) = \sum_{i=1}^{n} \alpha_i \, \mathrm{k}(x^{(i)}, x).$$

(Do you see the relevance of the identity in Problem 2 to kernel ridge regression?)

The file `hw4reg.pkl` contains data from $\mathbb{R} \times \mathbb{R}$ for a regression problem, split into training, validation, and test data.

```
import pickle
hw4reg = pickle.load(open('hw4reg.pkl', 'rb'))
```

Training data (features and labels) are in `hw4reg['data']` and `hw4reg['labels']`, validation data are in `hw4reg['valdata']` and `hw4reg['vallabels']`, and test data are in `hw4reg['testdata']` and `hw4reg['testlabels']`.

**Problem 3.** Consider the similarity function $\mathrm{k} \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ given by

$$\mathrm{k}(x, z) = \min\{x, z\}.$$

(a) Explain why k is a positive definite kernel.

*Hint:* $\min\{x, z\} = \int_0^\infty \mathbb{1}\{t \leq \min\{x, z\}\} \, \mathrm{d}t = \int_0^\infty \mathbb{1}\{t \leq x\} \mathbb{1}\{t \leq z\} \, \mathrm{d}t$.

(b) Use kernel ridge regression with the kernel function given above to solve the regression problem from `hw4reg.pkl`. Use the validation data to choose the regularization parameter $\lambda$. Specifically, try using values of $\lambda$ in the range $\{2^{-20}, 2^{-19}, \ldots, 2^{19}, 2^{20}\}$, report all validation mean squared errors (using a plot or a table), and then choose (and report) the value of $\lambda$ that leads to the smallest validation mean squared error. Report the test mean squared error of the final predictor using the chosen value of $\lambda$. Also plot the predictions of your final predictor $f \colon \mathbb{R} \to \mathbb{R}$ on the "grid" of 1000 equally-spaced points in the interval $[0, 1]$ supplied in `hw4reg['grid']`; superimpose this plot over the training data, and include the plot in your write-up. The following code may be helpful for this last part (although you are free to use your own code):

```
import matplotlib.pyplot as plt
pred = f(hw4reg['grid']) # change this line as needed
plt.figure()
plt.plot(hw4reg['data'], hw4reg['labels'], '.')
plt.plot(hw4reg['grid'], pred, linestyle='-')
plt.legend(['training data', '$f(x)$'], loc='lower left')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('hw4reg.pdf', bbox_inches='tight')
```

(c) (*Optional.*) Repeat Part (b) using the Gaussian kernel

$$\mathrm{k}_\sigma(x, z) = \exp\left(-\frac{(x - z)^2}{2\sigma^2}\right).$$

(Of course, you will need to also tune the hyperparameter $\sigma > 0$.) How does the final predictor compare to what you obtained in Part (b)?

# SVD and LSA

**Problem 4.** Suppose the $n \times d$ matrix $A$ of rank $r \geq 2$ has singular value decomposition

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\mathsf{T},$$

where $\sigma_1 \geq \cdots \geq \sigma_r$ are the singular values of $A$, $u_1, \ldots, u_r \in \mathbb{R}^n$ are corresponding left singular vectors of $A$, and $v_1, \ldots, v_r \in \mathbb{R}^d$ are corresponding right singular vectors of $A$.

(a) TRUE or FALSE: If $q_1$ and $q_2$ are orthonormal vectors in $\mathbb{R}^d$ such that the subspace spanned by $\{Aq_1, Aq_2\}$ is the same as the subspace spanned by $\{u_1, u_2\}$, then $\|Aq_1\|^2 + \|Aq_2\|^2 = \sigma_1^2 + \sigma_2^2$. Briefly justify your answer.

(b) TRUE or FALSE: If $q_1$ and $q_2$ are orthonormal vectors in $\mathbb{R}^d$ such that $\|Aq_1\|^2 + \|Aq_2\|^2 = \sigma_1^2 + \sigma_2^2$, then the subspace spanned by $\{Aq_1, Aq_2\}$ is the same as the subspace spanned by $\{u_1, u_2\}$. Briefly justify your answer.

*Latent semantic analysis (LSA)* is a technique based on SVD for studying the relationship between documents and the words they contain. In this next problem, you will apply a simple version of LSA to the Yelp restaurant review dataset.

The file `reviews_limited_vocab.txt` contains a subset of $n = 100000$ reviews from the training dataset (without the labels), with one review per line. Several insufficiently-frequent words have been removed from the reviews, leaving a vocabulary of just $d = 1731$ words. Using this data, we define the $n \times d$ "document-term" matrix $A$, where $A_{i,j}$ is the number of times word $j$ appears in review $i$.

- The rows of the matrix $A$ represent the reviews in the dataset in terms of the words they contain.
- The columns of the matrix $A$ represent the words in the vocabulary in terms of how many times they appear in the reviews from the dataset.

In LSA, the representations of the reviews and words are not taken directly from $A$, but rather, they are obtained using the truncated SVD of $A$. Take, for instance, the representations for the vocabulary words. Instead of using the columns of $A$, we use the orthogonal projections of these columns onto the subspace spanned by left singular vectors of $A$ corresponding to the $k$ largest singular values of $A$. Here, $k$, of course, is a hyperparameter of the LSA method. This projection is thought to remove some of the "noise" in the data that is due to spurious occurences of the words in reviews. (This vague statement can be made more precise using a probabilistic model; see, e.g., the JCSS article by Papadimitriou, Raghavan, Tamaki, and Vempala from 2000.)

What can one do with the vector representations of the words? One common application is to find other words that are "similar" to a given word. Similarity is usualy measured by the cosine of the angle between the corresponding vectors. This is called "cosine similarity". Words whose representations have high cosine similarity (i.e., close to 1) are consisdered

similar, while those that have low cosine similarity (i.e., close to 0 or perhaps even negative) are considered dissimilar.

**Problem 5.** Construct the document-term matrix $A$ from the Yelp review data. For each value of $k \in \{2, 4, 8\}$, apply LSA with the given $k$, and then compute the cosine similarity between all pairs of words in the following list:

```
words_to_compare = ['excellent', 'amazing', 'delicious', 'fantastic', 'gem',
↪  'perfectly', 'incredible', 'worst', 'mediocre', 'bland', 'meh', 'awful',
↪  'horrible', 'terrible']
```

Present your results in three separate tables (one per choice of $k$), where the rows and columns are the words in the list above, and the entries are the cosine similarities. (Feel free to be creative and additionally present the results in an interesting way.)
*Note:* The `numpy.linalg.svd` function can be used to compute the SVD of a matrix, but you should use the `full_matrices=False` option.

**Problem 6.** Sometimes LSA can be improved by ignoring the direction corresponding to the largest singular value. So, instead of using the left singular vectors corresponding to $\sigma_1, \ldots, \sigma_k$, one would use the left singular vectors corresponding to $\sigma_2, \ldots, \sigma_{k+1}$ instead. Repeat Problem 5 using this modification and compare the results. Do you notice any improvement? Feel free to consider other variants of LSA and show these results as well. See the Wikipedia article on LSA for some ideas.