CS 189:289A Introduction to Machine Learning
Fall 2021 Jennifer Listgarten and Jitendra Malik

HW4

Due 11/01 at 11:59pm

- We prefer that you typeset your answers using LaTeX or other word processing software. If you haven't yet learned LaTeX, one of the crown jewels of computer science, now is a good time! Neatly handwritten and scanned solutions will also be accepted for the written questions.
- In all of the questions, **show your work**, not just the final answer.

Deliverables:

1. Submit a PDF of your homework to the Gradescope assignment entitled "HW4 Write-Up". **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** put them in an appendix. We need each solution to be self-contained on pages of its own.
    - In your write-up, please state with whom you worked on the homework. This should be on its own page and should be the first page that you submit.
    - In your write-up, please copy the following statement and sign your signature next to it. (Mac Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadvertently cheats. *"I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."*
    - **Replicate all your code in an appendix**. Begin code for each coding question in a fresh page. Do not put code from multiple questions in the same page. When you upload this PDF on Gradescope, *make sure* that you assign the relevant pages of your code from appendix to correct questions.

1 Kernels

For a function $k(x_i, x_j)$ to be a valid kernel, it suffices to show either of the following conditions is true:

1. k has an inner product representation: $\exists\ \Phi : \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H}$ is some (possibly infinite-dimensional) inner product space such that $\forall x_i, x_j \in \mathbb{R}^d,\ k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.
2. For every sample $x_1, x_2, \ldots, x_n \in R^d$, the kernel matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & k(x_i, x_j) & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

   is positive semidefinite.

   Starting with part (c), you can use either condition (1) or (2) in your proofs.

(a) Show that the first condition implies the second one, i.e. if $\forall x_i, x_j \in \mathbb{R}^d,\ k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ then the kernel matrix K is PSD.

(b) Show that if the second condition holds, then for any finite set of vectors, $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, in $\mathbb{R}^d$ there exists a feature map $\Phi_{\mathcal{X}}$ that maps the finite set $\mathcal{X}$ to $\mathbb{R}^n$ such that, for all $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{X}$, we have $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi_{\mathcal{X}}(\mathbf{x}_i), \Phi_{\mathcal{X}}(\mathbf{x}_j) \rangle$.

(c) Given a positive semidefinite matrix A, show that $k(x_i, x_j) = x_i^\top A x_j$ is a valid kernel.

(d) Give a counterexample that shows why $k(x_i, x_j) = x_i^\top (\text{rev}(x_j))$ (where rev(x) reverses the order of the components in x) is *not* a valid kernel.

(e) Show that when $k \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a valid kernel, for all vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ we have

$$k(\mathbf{x}_1, \mathbf{x}_2) \leq \sqrt{k(\mathbf{x}_1, \mathbf{x}_1) k(\mathbf{x}_2, \mathbf{x}_2)}.$$

   Show how the classical Cauchy-Schwarz inequality is a special case.

(f) Suppose k_1 and k_2 are valid kernels with feature maps $\Phi_1 \colon \mathbb{R}^d \to \mathbb{R}^p$ and $\Phi_2 \colon \mathbb{R}^d \to \mathbb{R}^q$ respectively, for some finite positive integers p and q. Construct a feature map for the product of the two kernels in terms of Φ_1 and Φ_2, i.e. construct Φ_3 such that for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ we have

$$k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) k_2(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi_3(\mathbf{x}_1), \Phi_3(\mathbf{x}_2) \rangle.$$

   Hint: Recall that the inner product between two matrices $A, B \in R^{p \times q}$ is defined to be

$$\langle A, B \rangle = \text{tr}\left(A^\top B\right) = \sum_{i=1}^{p} \sum_{j=1}^{q} A_{ij} B_{ij}.$$

2 Kernel Ridge Regression: Theory

(a) As we already know, the following procedure gives us the solution to ridge regression in feature space:

$$\arg\min_{\mathbf{w}} \|\mathbf{\Phi w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \tag{1}$$

Recall from Homework 1 that the solution to ridge regression is given by

$$\hat{\mathbf{w}} = (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda I_d)^{-1} \mathbf{\Phi}^\top y$$

Show that we can rewrite $\hat{\mathbf{w}}$ as

$$\hat{\mathbf{w}} = \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda I_n)^{-1} y$$

You may have previously seen this in a past discussion.

(b) The prediction for a test point $\mathbf{x}$ is given by $\phi(\mathbf{x})^\top \hat{\mathbf{w}}$, where $\hat{\mathbf{w}}$ is the solution to (1). In this part we will show how $\phi(\mathbf{x})^\top \hat{\mathbf{w}}$ can be computed using only the kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. Denote the following object:

$$\mathbf{k}(\mathbf{x}) := [k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \ldots, k(\mathbf{x}, \mathbf{x}_n)]^\top$$

Using the result from part (a), show that

$$\phi(\mathbf{x})^\top \hat{\mathbf{w}} = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda I)^{-1} \mathbf{y}.$$

In other words, if we define $\hat{\alpha} := (\mathbf{K} + \lambda I)^{-1} \mathbf{y}$, then

$$\phi(\mathbf{x})^\top \hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

— our prediction is a linear combination of kernel functions at different data points.

(c) We will now consider kernel functions that do not directly correspond to a finite-dimensional featurization of the input points. For simplicity, we will stick to a scalar underlying raw input x. (The same style of argument can help you understand the vector case as well.) Consider the radial basis function (RBF) kernel function

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

for some fixed hyperparameter σ. It turns out that this kernel does not correspond to any finite-dimensional featurization $\phi(x)$. However, there exists a series $\phi_d(x)$ of d-dimensional features, such that $\phi_d(x)^T \phi_d(z)$ converges as $d \to \infty$ to $k(x, z)$. Using Taylor expansions, find $\phi_d(x)$.

(Hint: focus your attention on the Taylor expansion of $e^{\frac{xz}{\sigma^2}}$.)

3 Kernel Ridge Regression: Practice

In the following problem, you will implement Polynomial Ridge Regression and its kernel variant Kernel Ridge Regression, and compare them with each other. You will be dealing with a 2D regression problem, i.e., $\mathbf{x}_i \in \mathbb{R}^2$. We give you three datasets, `circle.npz` (small dataset), `heart.npz` (medium dataset), and `asymmetric.npz` (large dataset). In this problem, the labels are actually discrete $y_i \in \{-1, +1\}$, so in practice you should probably use a different model such as kernel SVMs, kernel logistic regression, or neural networks. The use of ridge regression here is for your practice and ease of coding.

You are only allowed to use `numpy.*`, `numpy.linalg.*`, and `matplotlib` in the following questions. Make sure to include plots and results in your writeups.

(a) Use `matplotlib` to visualize all the datasets and attach the plots to your report. Label the points with different y values with different colors and/or shapes.

(b) Implement polynomial ridge regression (non-kernelized version) to fit the datasets `circle.npz`, `asymmetric.npz`, and `heart.npz`. The data is already shuffled. Use the first 80% data as the training dataset and the last 20% data as the validation dataset. Report both the average training squared loss and the average validation squared loss for polynomial order $p \in \{2, 4, 6, 8, 10, 12\}$. Use the regularization term $\lambda = 0.001$ for all p. Visualize your result and attach the heatmap plots for the learned predictions over the entire 2D domain for $p \in \{2, 4, 6, 8, 10, 12\}$ in your report. Code for generating polynomial features and heatmap plots is included for your convenience.

(c) Implement kernel ridge regression to fit the datasets `circle.npz`, `heart.npz`, and optionally (due to the computational requirements), `asymmetric.npz`. Use the polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^p$. Use the first 80% data as the training dataset and the last 20% data as the validation dataset. Report both the average training squared loss and the average validation squared loss for polynomial order $p \in \{1, \ldots, 16\}$. Use the regularization term $\lambda = 0.001$ for all p. For `circle.npz`, also report the average training squared loss and validation squared loss for polynomial order $p \in \{1, \ldots, 24\}$ when you use only the first 15% data as the training dataset and the rest 85% data as the validation dataset. Based on the error, comment on when you want to use a high-order polynomial in linear/ridge regression. Lastly, comment on which of polynomial versus kernel ridge regression runs faster, and why.

(d) A popular kernel function that is widely used in various kernelized learning algorithms is called the radial basis function kernel (RBF kernel). It is defined as

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right). \tag{2}$$

Implement the RBF kernel function for kernel ridge regression to fit the dataset `heart.npz`. Use the regularization term $\lambda = 0.001$. Report the average squared loss, visualize your result and attach the heatmap plots for the fitted functions over the 2D domain for $\sigma \in \{10, 3, 1, 0.3, 0.1, 0.03\}$ in your report. You may want to vectorize your kernel functions to speed up your implementation.

4 Eigenfaces

In this question we will perform and analyze PCA on a dataset consisting of images of faces. Each datapoint, $\mathbf{x}$, consists of a flattened $62x47$ pixel image (i.e. $\mathbf{x} \in \mathbb{R}^{2914}$). The dataset can be downloaded using the *sklearn* library as follows:

```
dataset = sklearn.datasets.fetch_lfw_people()
X = dataset['data']
```

This may take a few minutes to run. The file sizes are 250Mb so be sure that you have enough space on your computer to store the images. The *sklearn* library should only be used for downloading the data. For the rest of the problem you may use *matplotlib*, *numpy.**, *numpy.linalg.eig* and *numpy.linalg.svd*.

(a) Plot the first 20 images to get familiar with the dataset.

Note: when plotting the images, be sure to reshape them to be a matrix of size 62 x 47. Images can be plotted with matplotlib.pyplot.imshow. The argument cmap=matplotlib.pyplot.cm.gray provides the best colormap to view the images

(b) Recall that in order to perform PCA, we must first center our data. Compute the average face of the dataset, center the data, and plot the average face.

(c) Perform PCA on the dataset. Plot the first 20 images reconstructed after being projected onto the top 10 principal components (PCs). Do the same after projecting onto the top 100 PCs and the top 1000 PCs.

(d) For this dataset, we refer to the PCs as "eigenfaces". Plot the top 20 eigenfaces.

(e) Recall from lecture that we can compute the percent variance explained by a certain number of PCs by using the eigenvalues of the covariance matrix. Plot the percent variance explained as a function of the number of PCs and determine how many PCs are needed to explain 95% of the variance.

(f) Use the first 80% of the dataset as your training set and the remaining 20% as the test set. We will use the training set to compute the PCs and we will evaluate our reconstruction loss on both the training and test set. For the following number of PCs, $[10, 20, 50, 100, 500, 1000, 2914]$, perform PCA using the training set and compute the average reconstruction loss for both the training and test set. Plot the error for both the training and test set as a function of the number of PCs.