CS 189 Introduction to Machine Learning
Summer 2018 HW4

1 Getting Started

Read through this page carefully. You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup, **with an appendix for your code**, to assignment on Gradescope, "HW4 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.
2. If there is code, submit all code needed to reproduce your results, "HW4 Code".
3. If there is a test set, submit your test set evaluation results, "HW4 Test Set".

After you've submitted your homework, watch out for the self-grade form.

(a) Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it. We just want to make it *extra* clear so that no one inadverdently cheats.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

This homework is due **Monday, July 16 at 11:59pm.**

2 Kernel Ridge Regression: Theory

In ridge regression, we are given a vector $\mathbf{y} \in \mathbb{R}^n$ and a matrix $\mathbf{X} \in \mathbb{R}^{n \times \ell}$, where n is the number of training points and ℓ is the dimension of the raw data points. In most settings we don't want to work with just the raw feature space, so we augment the data points with features and replace $\mathbf{X}$ with $\mathbf{\Phi} \in \mathbb{R}^{n \times d}$, where $\boldsymbol{\phi}_i^\top = \boldsymbol{\phi}(\mathbf{x}_i) \in \mathbb{R}^d$. Then we solve a well-defined optimization problem that involves the matrix $\mathbf{\Phi}$ and $\mathbf{y}$ to find the parameters $\mathbf{w} \in \mathbb{R}^d$. Note the problem that arises here. If we have polynomial features of degree at most p in the raw ℓ dimensional space, then there are $d = \binom{\ell+p}{p}$ terms that we need to optimize, which can be very, very large (much larger than the number of training points n). Wouldn't it be useful, if instead of solving an optimization problem over d variables, we could solve an equivalent problem over n variables (where n is potentially much smaller than d), and achieve a computational runtime independent of the number of augmented features? As it turns out, the concept of kernels (in addition to a technique called the kernel trick) will allow us to achieve this goal.

(a) (Dual perspective of the kernel method) In lecture, you saw a derivation of kernel ridge regression involving Gaussians and conditioning. There is also a pure optimization perspective that uses Lagrangian multipliers to find the dual of the ridge regression problem. First, we could rewrite the original problem as

$$\begin{aligned} \underset{\mathbf{w},\mathbf{r}}{\text{minimize}} \quad & \frac{1}{2}\left[\|\mathbf{r}\|_2^2 + \lambda \|\mathbf{w}\|_2^2\right] \\ \text{subject to} \quad & \mathbf{r} = \mathbf{X}\mathbf{w} - \mathbf{y}. \end{aligned}$$

Show that the solution of this is equivalent to

$$\min_{\mathbf{w},\mathbf{r}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \mathbf{r}, \boldsymbol{\alpha}) := \min_{\mathbf{w},\mathbf{r}} \max_{\boldsymbol{\alpha}} \left[\frac{1}{2}\|\mathbf{r}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \boldsymbol{\alpha}^\top(\mathbf{r} - \mathbf{X}\mathbf{w} + \mathbf{y})\right], \tag{1}$$

where $L(\mathbf{w}, \mathbf{r}, \boldsymbol{\alpha})$ is the Lagrangian function.

(b) Using the minmax theorem[1], we can swap the min and max (think about what the order of min and max means here and why it is important):

$$\min_{\mathbf{w},\mathbf{r}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \mathbf{r}, \boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \min_{\mathbf{w},\mathbf{r}} L(\mathbf{w}, \mathbf{r}, \boldsymbol{\alpha}). \tag{2}$$

Argue that the right hand side is equal to

$$\min_{\boldsymbol{\alpha}} \left[\frac{1}{2}\boldsymbol{\alpha}^\top(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} - \lambda\boldsymbol{\alpha}^\top\mathbf{y}\right] \text{ where } \mathbf{K} = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{n \times n}. \tag{3}$$

You can do this by setting the appropriate partial derivative of the Lagrangian L to zero. This is often call *the Lagrangian dual problem* of the original optimization problem.

[1] https://www.wikiwand.com/en/Minimax_theorem

(c) **Finally, prove that the optimal $\mathbf{w}^*$ can be computed using**

$$\mathbf{w}^* = \mathbf{X}^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}. \tag{4}$$

(d) (Polynomial Regression from a kernelized view) In this part, we will show that polynomial regression with a particular Tiknov regularization is the same as kernel ridge regression with a polynomial kernel for second-order polynomials. Recall that a degree 2 polynomial kernel function on $\mathbb{R}^d$ is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2, \tag{5}$$

for any $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^\ell$. Given a dataset $(\mathbf{x}_i, y_i)$ for $i = 1, 2, \ldots, n$, **show the solution to kernel ridge regression is the same as the regularized least square solution to polynomial regression (with unweighted monomials as features) for $p = 2$ given the right choice of Tikhonov regularization for the polynomial regression.** That is, show for any new point $\mathbf{x}$ given in the prediction stage, both methods give the same prediction $\hat{y}$ with the same training data. **What is the Tikhonov regularization matrix here?**

Hint: You may or may not use the following matrix identity:

$$\mathbf{A}(a\mathbf{I}_d + \mathbf{A}^\top \mathbf{A})^{-1} = (a\mathbf{I} + \mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A}, \tag{6}$$

for any matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and any positive real number a.

(e) In general, for any polynomial regression with pth order polynomial on $\mathbb{R}^\ell$ with an appropriately specified Tikhonov regression, we can show the equivalence between it and kernel ridge regression with a polynomial kernel of order p. **Comment on the computational complexity of doing least squares for polynomial regression with this Tikhonov regression directly and that of doing kernel ridge regression in the training stage.** (That is, the complexity of finding $\boldsymbol{\alpha}$ and finding $\mathbf{w}$.) **Compare with the computational complexity of actually doing prediction as well.**

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, we choose $y_i \in \{-1, +1\}$, so you may view this question as a classification problem. Later on in the course we will learn about logistic regression and SVMs, which can solve classification problems much better and can also leverage kernels.

(a) **Use `matplotlib.pyplot` 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. You are only allow to use `numpy.*` and `numpy.linalg.*` in the following questions.

(b) **Implement polynomial ridge regression** (non-kernelized version that you should already have implemented in your previous homework) **to fit the datasets `circle.npz`, `asymmetric.npy`, and `heart.npz`**. 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 for polynomial order** $p \in \{1, \ldots, 16\}$. 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.** You can start with the code from previous homeworks.

(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. The sample code for generating heatmap plot is included in the start kit. **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.**

(d) (Diminishing influence of the prior with growing amount of data) With increasing of amount of data, the prior (from the statistical view) and regularization (from the optimization view) will be washed away and become less and less important. Sample the training data from the first 80% data from `asymmetric.npz` and use the data from the last 20% data for validation. **Make a plot whose** x **axis is the amount of the training data and** y **axis is the validation squared loss of the non-kernelized ridge regression algorithm. Optionally, repeat the same for kernel ridge regression.** Include 6 curves for hyper-parameters $\lambda \in \{0.0001, 0.001, 0.01\}$ and $p = \{5, 6\}$. Your plot should demonstrate that with same p, the validation squared loss will converge with enough data, regardless of the choice of λ and/or the regularizer matrix. You can use log plot on x axis for clearity and you need to resample the data multiple times for the given p, λ, and the amount of training data in order to get a smooth curve.

(e) 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

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right). \tag{7}$$

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. **Comment on the effect of** σ**.**

(f) For polynomial ridge regression, **which of your implementation is more efficient, the kernelized one or the non-kernelized one?** For RBF kernel, **explain whether it is possible to implement it in the non-kernelized ridge regression. Summarize when you prefer the kernelized to the non-kernelized ridge regression.**

4 Ridge regression vs. PCA

Assume we are given n training data points $(\mathbf{x}_i, y_i)$. We collect the target values into $\mathbf{y} \in \mathbb{R}^n$, and the inputs into the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ where the rows are the $d-$dimensional feature vectors $\mathbf{x}_i^\top$ corresponding to each training point. Furthermore, assume that $\frac{1}{n}\sum_{i=1}^n \mathbf{x_i} = \mathbf{0}$, $n > d$ and $\mathbf{X}$ has rank d.

In this problem we want to compare two procedures: The first is ridge regression with hyperparameter λ, while the second is applying ordinary least squares after using PCA to reduce the feature dimension from d to k (we give this latter approach the short-hand name k-PCA-OLS where k is the hyperparameter).

Notation: The singular value decomposition of $\mathbf{X}$ reads $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times d}$ and $\mathbf{V} \in \mathbb{R}^{d \times d}$. We denote by $\mathbf{u}_i$ the n-dimensional column vectors of $\mathbf{U}$ and by $\mathbf{v}_i$ the $d-$dimensional column vectors of $\mathbf{V}$. Furthermore the diagonal entries $\sigma_i = \Sigma_{i,i}$ of $\mathbf{\Sigma}$ satisfy $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$. For notational convenience, assume that $\sigma_i = 0$ for $i > d$.

(a) It turns out that the ridge regression optimizer (with $\lambda > 0$) in the $\mathbf{V}$-transformed coordinates

$$\widehat{\mathbf{w}}_{\text{ridge}} = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{V}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

has the following expression:

$$\widehat{\mathbf{w}}_{\text{ridge}} = \text{diag}\left(\frac{\sigma_i}{\lambda + \sigma_i^2}\right)\mathbf{U}^\top\mathbf{y}. \tag{8}$$

Use $\widehat{y}_{test} = \mathbf{x}_{test}^\top \mathbf{V}\widehat{\mathbf{w}}_{\text{ridge}}$ to denote the resulting prediction for a hypothetical $\mathbf{x}_{test}$. Using (8) and the appropriate scalar $\{\beta_i\}$, this can be written as:

$$\widehat{y}_{test} = \mathbf{x}_{test}^\top \sum_{i=1}^{d} \mathbf{v}_i \beta_i \mathbf{u}_i^\top \mathbf{y}. \tag{9}$$

What are the $\beta_i \in \mathbb{R}$ for this to correspond to (8) from ridge regression?

(b) Suppose that we do k-PCA-OLS — i.e. ordinary least squares on the reduced k-dimensional feature space obtained by projecting the raw feature vectors onto the $k < d$ principal components of the covariance matrix $\mathbf{X}^\top \mathbf{X}$. Use $\widehat{y}_{test}$ to denote the resulting prediction for a hypothetical $\mathbf{x}_{test}$,

It turns out that the learned k-PCA-OLS predictor can be written as:

$$\widehat{y}_{test} = \mathbf{x}_{test}^\top \sum_{i=1}^{d} \mathbf{v}_i \beta_i \mathbf{u}_i^\top \mathbf{y}. \tag{10}$$

Give the $\beta_i \in \mathbb{R}$ coefficients for k-PCA-OLS. Show work.

Hint 1: some of these β_i will be zero. Also, if you want to use the compact form of the SVD, feel free to do so if that speeds up your derivation.

Hint 2: some inspiration may be possible by looking at the next part for an implicit clue as to what the answer might be.

(c) For the following part, $d = 5$. The following $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_5)$ (written out to two significant figures) are the results of OLS (i.e. what we would get from ridge regression in the limit $\lambda \to 0$), λ-ridge-regression, and k-PCA-OLS for some $\mathbf{X}, \mathbf{y}$ (identical for each method) and $\lambda = 100, k = 3$. **For each of the three sub-parts below, determine if it is possible for the given β to be generated by k-PCA-OLS and explain your reasoning.**

We hope this helps you intuitively see the connection between these three methods.

Hint: The singular values of $\mathbf{X}$ *are* $\sigma_1 = 100$*,* $\sigma_2 = 10$*,* $\sigma_3 = 2$*,* $\sigma_4 = 0.1$*,* $\sigma_5 = 0.01$*.*

(i) $\boldsymbol{\beta} = (0.01, 0.1, 0.5, 10, 100)$

(ii) $\boldsymbol{\beta} = (0.01, 0.05, 0.02, 0, 0)$

(iii) $\boldsymbol{\beta} = (0.01, 0.1, 0.5, 0, 0)$

5 Kernel PCA

In lectures, discussion, and homework, we learned how to use PCA to do dimensionality reduction by projecting the data to a subspace that captures most of the variability. This works well for data that is roughly Gaussian shaped, but many real-world high dimensional datasets have underlying low-dimensional structure that is not well captured by linear subspaces. However, when we lift the raw data into a higher-dimensional feature space by means of a nonlinear transformation, the underlying low-dimensional structure once again can manifest as an approximate subspace. Linear dimensionality reduction can then proceed. As we have seen in class so far, kernels are an alternate way to deal with these kinds of nonlinear patterns without having to explicitly deal with the augmented feature space. This problem asks you to discover how to apply the "kernel trick" to PCA.

Let $\mathbf{X} \in \mathbb{R}^{n \times \ell}$ be the data matrix, where n is the number of samples and ℓ is the dimension of the

raw data. Namely, the data matrix contains the data points $\mathbf{x}_j \in \mathbb{R}^\ell$ as rows

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times \ell}. \tag{11}$$

(a) **Compute $\mathbf{X}\mathbf{X}^\top$ in terms of the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{V} \in \mathbb{R}^{\ell \times \ell}$.** Notice that $\mathbf{X}\mathbf{X}^\top$ is the matrix of pairwise Euclidean inner products for the data points. **How would you get $\mathbf{U}$ if you only had access to $\mathbf{X}\mathbf{X}^\top$?**

(b) Given a new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, one central use of PCA is to compute the projection of $\mathbf{x}_{test}$ onto the subspace spanned by the k top singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.

Express the scalar projection $z_j = \mathbf{v}_j^\top \mathbf{x}_{test}$ onto the j-th principal component as a function of the inner products

$$\mathbf{X}\mathbf{x}_{test} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_{test} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_{test} \rangle \end{pmatrix}. \tag{12}$$

Assume that all diagonal entries of $\mathbf{\Sigma}$ are nonzero and non-increasing, that is $\sigma_1 \geq \sigma_2 \geq \cdots > 0$.

Hint: Express $\mathbf{V}^\top$ in terms of the singular values $\mathbf{\Sigma}$, the left singular vectors $\mathbf{U}$ and the data matrix $\mathbf{X}$. If you want to use the compact form of the SVD, feel free to do so.

(c) How would you define kernelized PCA for a general kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ (to replace the Euclidean inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$)? For example, the RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\delta^2}\right)$.

Describe this in terms of a procedure which takes as inputs the training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^\ell$ and the new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, and outputs the analog of the previous part's z_j coordinate in the kernelized PCA setting. You should include how to compute U from the data, as well as how to compute the analog of $\mathrm{X}\mathbf{x}_{test}$ from the previous part.

Invoking the SVD or computing eigenvalues/eigenvectors is fine in your procedure, as long as it is clear what matrix is having its SVD or eigenvalues/eigenvectors computed. The kernel $k(\cdot, \cdot)$ can be used as a black-box function in your procedure as long as it is clear what arguments it is being given.

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn the material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.