CS 189 Introduction to Machine Learning
Fall 2018 HW3

# 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 to assignment on Gradescope, "HW3 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.

(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 **Friday, September 21st at 10pm.**

# 2 Properties of kernels

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 $\ell$ 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 $\ell$ 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.

For a function $k$ 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 $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d,\ k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. The map $\Phi$ is called a *feature map* of the kernel $k$.
2. For every sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in R^d$, the Gram matrix

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & k(\mathbf{x}_i, \mathbf{x}_j) & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

   is positive semidefinite.

(a) 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 $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi_{\mathcal{X}}(\mathbf{x}_i), \Phi_{\mathcal{X}}(\mathbf{x}_j) \rangle$.

(b) 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.

(c) 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 $\Phi_1$ and $\Phi_2$, i.e. construct $\Phi_3$ such that for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ we have

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

# 3 Understanding specific kernels

(a) (Polynomial Regression from a kernelized view) In this part, we will show that polynomial regression with a particular 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

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

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 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.

(Hint: you should consider a regularization term of the form $\|Mw\|_2^2$ for some matrix $M$. Such regularization is called *Tikhonov regularization*.)

(b) In general, for any polynomial regression with $p$th order polynomial on $\mathbb{R}^\ell$ with an appropriately specified Tikhonov regularization, 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 the 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.

(c) Show that the function $k$ defined by $k(x_1, x_2) = \exp\left(\frac{x_1 x_2}{\gamma^2}\right)$ for all $x_1, x_2 \in \mathbb{R}$ is a valid kernel. Comment on the relation between polynomial regression and kernel regression with the kernel $k$. What effect does $\gamma$ have on the importance of different features in the resulting regression model? (Hint: consider the Taylor series expansion of the function $e^z$.)

(d) Consider the function $k\colon (0,1) \times (0,1) \to \mathbb{R}$ defined by $k(x_1, x_2) = \min\{x_1, x_2\}$. Prove that $k$ is a valid kernel (Hint: write $k$ as the integral of a product of two simple functions and then prove that its Gram matrices are positive semi-definite).

Now, consider a training set $\{(x_i, y_i)\}_{i=1,\ldots,n}$ with $y_i \in \mathbb{R}$ and distinct points $x_i$ in $(0,1)$. Show that if we ran kernel regression without regularization on this data set, we would obtain zero training error. More precisely, find explicit coefficients $\alpha_j$, in terms of the training data, such that for all points $(x_i, y_i)$ in the training set we have

$$\sum_{j=1}^{n} \alpha_j \min\{x_j, x_i\} = y_i.$$

(e) Let $k$ be a valid kernel with positive definite Gram matrices and let us consider a training set $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for all $i$, and with all vectors $\mathbf{x}_i$ distinct. What training error should we expect if we ran kernel regression without regularization on this data set? Justify your answer.

# 4 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.

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 that you should already have implemented in your previous homework) to fit the datasets `circle.npz`, `asymmetric.npz`, 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 homework 2.

(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) With increasing of amount of data, the gains from regularization diminish. 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. 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 $\lambda$. You can use log plot on $x$ axis for clarity and you need to resample the data multiple times for the given $p$, $\lambda$, 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

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

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.