CS 189 Introduction to Machine Learning
Spring 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 **Friday, February 16 at 10pm.**

2 MLE of Multivariate Gaussian

In lecture, we discussed uses of the multivariate Gaussian distribution. We just assumed that we knew the parameters of the distribution (the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$). In practice, though, we will often want to estimate μ and Σ from data. (This will come up even beyond regression-type problems: for example, when we want to use Gaussian models for classification problems.) This problem asks you to derive the Maximum Likelihood Estimate for the mean and variance of a multivariate Gaussian distribution.

(a) Let $\mathbf{X}$ have a multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. **Write the log likelihood of drawing the n i.i.d. samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ from X given Σ and μ.**

(b) **Find MLE of μ and Σ.** For taking derivatives with respect to matrices, you may use any formula in "The Matrix Cookbook" without proof. This is a reasonably involved problem part with lots of steps to get to the answer. We recommend students first do the one-dimensional case and then the two-dimensional case to warm up.

Note: Conventions for gradient and derivative in "The Matrix Cookbook" may vary from the conventions we saw in the discussion.

(c) Use the following code to sample from a two-dimensional Multivariate Gaussian and plot the samples:

```
import numpy as np
import matplotlib.pyplot as plt
mu = [15, 5]
sigma = [[20, 0], [0, 10]]
samples = np.random.multivariate_normal(mu, sigma, size=100)
plt.scatter(samples[:, 0], samples[:, 1])
plt.show()
```

Try the following three values of Σ:

$$\Sigma = \begin{bmatrix} 20 & 0 \\ 0 & 10 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 20 & 14 \\ 14 & 10 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 20 & -14 \\ -14 & 10 \end{bmatrix}.$$

Calculate the mean and covariance matrix of these distributions from the samples (that is, implement part (b)). Report your results. Include your code in your write-up. Note: you are allowed to use numpy.

3 Tikhonov Regularization and Weighted Least Squares

In lecture, you have seen this worked out in one way. In homework 2 we introduced Tikhonov regularization as a generalization of ridge regression. In this problem, we look at Tikhonov regularization from a probabilistic standpoint.

The main goal is to deepen your understanding of how priors and thus the right regularization affect the MAP estimator. First, you will work out how introducing a certain probabilistic prior before maximizing the posterior is equivalent to adding the Tikhonov regularization term: by adding the Tikhonov regularization term, we effectively constrain our optimization space. Similarly, using a probabilistic prior drives our optimization towards solutions that have a high (prior) probability of occurring. In the second half of the problem you will then do some simulations to see how different priors influence the estimator explicitly, as well as how this effect changes as the number of samples grows.

(a) Let $\mathbf{x} \in \mathbb{R}^d$ be a d-dimensional vector and $Y \in \mathbb{R}$ be a one-dimensional random variable. Assume a linear model: $Y = \mathbf{x}^\top \mathbf{w} + Z$ where $Z \in \mathbb{R}$ is a standard Gaussian random variable $Z \sim N(0, 1)$ and $\mathbf{w} \in \mathbb{R}^d$ is a d-dimensional Gaussian random vector $\mathbf{w} \sim N(0, \mathbf{\Sigma})$. $\mathbf{\Sigma}$ is a known symmetric positive definite covariance matrix. Note that $\mathbf{w}$ is independent of the observation noise. **What is the conditional distribution of Y given $\mathbf{x}$ and $\mathbf{w}$?**

(b) (Tikhonov regularization) Let us assume that we are given n training data points $\{(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2), \cdots, (\mathbf{x}_n, Y_n)\}$ which we know are generated i.i.d. according to the model of $(\mathbf{x}, Y)$ in the previous part, i.e. we draw one $\mathbf{w}$ and use this to generate all Y_i given distinct but arbitrary $\{\mathbf{x}_i\}_{i=1}^n$, but the observation noise Z_i varies across the different training points. **Derive the posterior distribution of $\mathbf{w}$ given the training data. Based on your result, what is the MAP estimate of $\mathbf{w}$? Comment on how Tikhonov regularization is a generalization of ridge regression from a probabilistic perspective.**

Note: $\mathbf{w}$ and $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)$ are jointly Gaussian in this problem given $\{\mathbf{x}_i\}_{i=1}^n$.

Hint: (You may or may not find this useful) If the probability density function of a random variable is of the form

$$f(\mathbf{v}) = C \cdot \exp\{-\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v}\},$$

where C is some constant to make $f(\mathbf{v})$ integrates to 1, then the mean of $\mathbf{v}$ is $\mathbf{A}^{-1}\mathbf{b}$. This can be used to help complete squares if you choose to go that way.

(c) We have so far assumed that the observation noise has a standard normal distribution. While this assumption is nice to work with, we would like to be able to handle a more general noise model. In particular, we would like to extend our result from the previous part to the case where the observation noise variables Z_i are no longer independent across samples, i.e. $\mathbf{Z}$ is no longer $N(\mathbf{0}, \mathbb{I}_n)$ but instead distributed as $N(\boldsymbol{\mu}_z, \mathbf{\Sigma}_z)$ for some mean μ_z and some covariance $\mathbf{\Sigma}_z$ (still independent of the parameter $\mathbf{w}$). **Derive the posterior distribution of $\mathbf{w}$ by appropriately changing coordinates.** We make the reasonable assumption that the $\mathbf{\Sigma}_z$ is invertible, since otherwise, there would be some dimension in which there is no noise.

Hint: Write $\mathbf{Z}$ as a function of a standard normal Gaussian vector $\mathbf{V} \sim N(\mathbf{0}, \mathbb{I}_n)$ and use the result in (b) for an equivalent model of the form $\widetilde{\mathbf{Y}} = \widetilde{\mathbf{X}}\mathbf{w} + \mathbf{V}$.

(d) (Compare the effect of different priors) In this part, you will generate plots that show how different priors on $\mathbf{w}$ affect our prediction of the true $\mathbf{w}$ which generated the data points. Pay attention to how the amount of data used and the choice of prior relative to the true $\mathbf{w}$ we use are related to the final prediction.

Do the following for $\mathbf{\Sigma} = \mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3, \mathbf{\Sigma}_4, \mathbf{\Sigma}_5, \mathbf{\Sigma}_6$ respectively, where

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{\Sigma}_2 = \begin{bmatrix} 1 & 0.25 \\ 0.25 & 1 \end{bmatrix}; \quad \mathbf{\Sigma}_3 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix};$$

$$\mathbf{\Sigma}_4 = \begin{bmatrix} 1 & -0.25 \\ -0.25 & 1 \end{bmatrix}; \quad \mathbf{\Sigma}_5 = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix}; \quad \mathbf{\Sigma}_6 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Under the priors above, the coordinates of the (random) vector $\mathbf{w}$ are: (1) independent with large variance, (2) mildly positively correlated, (3) strongly positively correlated, (4) mildly negatively correlated, (5) strongly negatively correlated, and (6) independent with small variances respectively.

Using the starter code, generate $n \in \{5, 6, \ldots, 500\}$ data points $Y = x_1 + x_2 + Z$ with $x_1, x_2 \sim N(0, 5)$ and $Z \sim N(0, 1)$ as training data (here, the true $\mathbf{w}$ is thus $\begin{bmatrix} 1 & 1 \end{bmatrix}^\top$). Note that the randomness of x_i here is only for the generation of the plot but in our probabilistic model for parameter estimation we consider them as fixed and given. The starter code helps you generate an interactive plot where you can adjust the covariance prior and the number of samples used to calculate the posterior. **Include 6 plots of the contours of the posteriors on $\mathbf{w}$ for various settings of $\mathbf{\Sigma}$ and number of data points. Write the covariance prior and number of samples for each plot. What do you observe as the number of data points increases?**

(e) (Influence of Priors) For our simulations, we will generate n training data samples from $Y = x_1 + x_2 + Z$ where again $x_1, x_2 \sim N(0, 5)$ and $Z \sim N(0, 1)$ (all of them independent of each other) as before. Notice that the true parameters $w_1 = 1, w_2 = 1$ are moderately large and positively correlated with each other. We want to quantitatively understand how the effect of the prior influences the mean square error as we get more training data. This should corroborate the qualitative results you saw in the previous part.

In this case, we could directly compute the "test error" for a given estimator $\widehat{\mathbf{w}}$ of the parameter $\mathbf{w}$ (our prediction for Y given a new data point $\mathbf{x} = (x_1, x_2)^\top$ is then $\widehat{Y} = \widehat{w}_1 x_1 + \widehat{w}_2 x_2$). Specifically, considering $\widehat{\mathbf{w}}$ now fixed, the expected error for a randomly drawn Y given the true (but unknown) parameter vector $\mathbf{w} = (1, 1)^\top$ is equal to $\mathbb{E}_{Z,\mathbf{x}}(Y - \widehat{Y})^2 = 5(\widehat{w}_1 - 1)^2 + 5(\widehat{w}_2 - 1)^2 + 1$. We call this the *theoretical average test error*. Note that here by our choice of definition, the expectation for new test samples is over $\mathbf{x}$ as well, although our estimator is not taking the randomness of $\mathbf{x}$ in the training data into account.

In practice, the expectation with respect to the true conditional distribution of Y given $\mathbf{w}$ cannot be computed since the true $\mathbf{w}$ is unknown. Instead, we are only given a finite amount of samples from the model (which we call the *test set*, which independent of the training data, but identically distributed) so that it is only possible to compute

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

which we call the *empirical average test error* (also known as MSE). Again, note that here, $\hat{Y}_i = \mathbf{x}_i^\top \widehat{\mathbf{w}}$ where $\mathbf{x}_i \in \mathbb{R}^2$ and $\widehat{\mathbf{w}}$ in your model is the solution to the least square problem with Tikhonov regularization given the training data.

Generate a test set of 500 data points $(\mathbf{x}_i, Y_i)$ from the above model. Plot the empirical and theoretical mean square error between $\widehat{Y}_i$ and Y_i over the test data with respect to the size of training data n (increase n from 5 to 200 in increments of 5).

Note: If we just plotted both the empirical and theoretical average test errors with respect to the amount of training data for one "round" or training data, the results would still look jagged. In order to give a quantitive statement about the test error with respect to the training data n with a "smoother" plot, what we really want to know is the expectation of the theoretical average test error with respect to $\mathbf{w}$ and the training samples $(\mathbf{x}_i, Y_i)$, i.e. $\mathbb{E}_{(\mathbf{x}_1,Y_1),\ldots,(\mathbf{x}_n,Y_n)}\mathbb{E}_{Z,\mathbf{x}}(Y - \widehat{Y})^2$ (note that in this term, only $\widehat{Y}$ depends on the training data $(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)$ whereas $(\mathbf{x}, Y)$ is an independent fresh test sample). Consequently, as an approximation, it is worth replicating the entire experiment a few times (say 100 times) to get an empirical estimate of this quantity. (It is also insightful to look at the spread.) **Compare what happens for different priors as the amount of training data increases. Try plotting the theoretical MSE with logarithmic x and y-axes and explain the plot. What constitutes a "good" prior and which of the given priors are "good" choices for our particular $\mathbf{w} = (1, 1)^\top$? Describe how the influence of different priors changes with the number of data points.**

4 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

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

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

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

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.

(c) **Finally, prove that the optimal** 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.

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

(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 w.) **Compare with the computational complexity of actually doing prediction as well.**

5 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 homework 2, problem 5.

(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.**

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.