CS 274A Homework 5

Probabilistic Learning: Theory and Algorithms, CS 274A, Winter 2023

Due: 23:59, Tuesday March 7th, submit via Gradescope

Instructions and Guidelines for Homeworks

- Please answer all of the questions and submit your solutions to Gradescope (either hand-written or typed are fine as long as the writing is legible).
- All problems are worth equal points (10 points) unless otherwise stated. All homeworks will get equal weight in computation of the final grade for the class (with lowest-scoring homework being dropped).
- The homeworks are intended to help you better understand the concepts we discuss in class. It is important that you solve the problems yourself to help you learn and reinforce the material from class. If you don't do the homeworks you will likely have difficulty in the exams later in the quarter.
- In problems that ask you to derive or prove a result you should submit a complete mathematical proof (i.e., each line must follow logically from the preceding one, without "hand-waving"). Be as clear as possible in explaining your notation and in stating your reasoning as you go from line to line.
- If you can't solve a problem, you can discuss the high-level concepts *verbally* with another student (e.g., what concepts from the lectures or notes or text are relevant to a problem). However, you should not discuss any of the details of a solution with another student. In particular note that you are not allowed to view (or show to any other student) *any written material* directly related to the homeworks, including other students' solutions or drafts of solutions, solutions from previous versions of this class, etc. The work you hand in should be your own original work.
- If you need to you can look up standard results/definition/identities from textbooks, class notes, textbooks, other reference material (e.g., from the Web). If you base any part of your solution on material that we did not discuss in class, or is not in the class notes, or is not a standard known result, then you may want to rovide a reference in terms of where the result is from, e.g., "based on material in Section 2.2 in ....." or a URL.

Problem 1: Bias-Variance Trade Off in Linear Regression

In this problem, we are interested in the trade-off between bias and variance in linear regression model on synthetic data. Assume we are going to estimate the true function $E[y|x] = \sin(x)$ with $x \in [-2\pi, 2\pi]$ from noisy IID data, using linear regression. Note that $\sin(x)$ is non-linear in x, but it can be written as the sum of polynomials according to its Taylor expansion around zero

$$\sin(x) = \sum_{k \geq 0} \frac{(-1)^k}{(2k+1)!} x^{2k+1}.$$

Suppose we wish to use higher degree polynomials of x as features to learn an approximation $f(x)$ to $\sin(x)$; we denote degree-d polynomial features of x as vector $\phi_d(x) = [x, x^2, x^3, ..., x^d]$. To generate a dataset of size n, we will sample x_i for $i = 1, ..., n$ uniformly at random for $x \in [-2\pi, 2\pi]$ and set $y_i = \sin(x_i) + z$, where $z \sim \mathcal{N}(0, 0.16)$ to create a dataset $D_n = \big\{(x_i, y_i)\big\}_{i=1}^n$.

Below we will refer to a trained model as $\hat{f}(x)$, the true target function as $f = \sin(x)$, and predictions for any x as $\hat{y} = \hat{f}(x)$. Note that we only consider x inside the range $[-2\pi, 2\pi]$ for both training and evaluation. To measure the quality of a trained model $\hat{f}$, we will compute MSE between f and $\hat{f}$ on 1000 equidistant generated points from the interval containing -2π and 2π .

Note. If you are using Numpy and Python, please set random seed to 1 or insert **np.seed(1)** in your code for this problem.

1. Generate a dataset of length 16, i.e., D_{16} as training data, and consider two different ways to fit models: (a) linear regression without regularization (ordinary LR) and (b) linear regression with L_2 penalty and regularization coefficient $\lambda = 5 \times 10^{-3}$ (ridge LR). Use degree-d polynomials ϕ_d with various degrees $d \in \big\{1, 3, 5, 6, 9, 11, 13, 15\big\}$, and train the models with each d. For obtaining weights in each ordinary LR or ridge LR, you can use any method for solving the optimization problem (normal equation, gradient descent, stochastic gradient descent, . . .) as long as it converges.

   Plot the target function f, training data points D_{16}, estimated training data points $\hat{y}_i, i = 1, \ldots, 16$, and trained model $\hat{f}$ inside one figure for each scenario. That is, provide 8 figures for ordinary LR and 8 figures for ridge LR with different d values (see figure 1 for one such plot when $d = 3$ and ordinary LR was used). Organize each set of 8 plots to 4×2 layout with the top left plot corresponding to $d = 1$.

   - Comment on what you see in the plots.

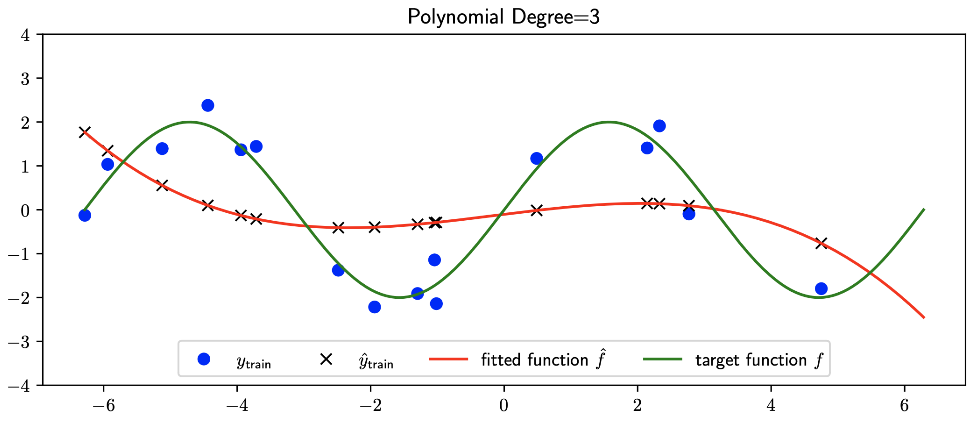

Figure 1: Sample plot for problem 1.1.

2. Perform the previous part $K = 100$ times (generate new dataset each time) and compute train and test error of each model measured by MSE. Plot the mean of train and test MSE with error bars as one standard deviation across K runs as a function of d for each model separately. Provide two figures; one for ordinary LR and one for ridge LR.

   - Explain the relation between the mean and standard deviation of train and test error of each model with d?
   - What is the connection between these plots and the bias-variance trade off?

3. To see the effect of increasing dataset size, generate D_n for $n \in \{32, 64, 128, 256, 512\}$ and again use (a) ordinary LR and (b) ridge LR with $\lambda = 5 \times 10^{-3}$ to fit models. Use degree-d polynomials ϕ_d for $d \in \{3, 9, 15\}$ to train the models. Perform this procedure $K = 100$ times and compute test MSE each time. Similar to the previous part, plot the mean of test MSE with error bars as one standard deviation across K runs as a function of dataset size n for each d and model separately. Again provide two figures; one for ordinary LR and one for ridge LR.

   - Explain the relation between the mean and standard deviation of train and test error of each model with n?

Hint. You might need to standardize the data after extracting polynomial features, for avoiding numerical issues. Also, make sure to include a bias term in each of your models.

Problem 2: LASSO

As we saw in the homework 4, assuming a Laplace prior for weights in linear regression leads to adding L_1 regularization. Linear regression with the L_1 penalty or LASSO is well-known for obtaining sparse weights i.e., explaining response variables with a small number of explanatory variables. This property is particularly useful when some of the features are independent of the target variable, which is the case in many machine learning applications.

In this problem, we want to investigate the sparsity properties of LASSO on a synthetic dataset. To begin with, we have generated a random weight vector $w \in \mathbb{R}^d$ and zeroed out the components with even indices. Then, the response variable is obtained by computing $y = w^T x + z$ where z is a Gaussian noise. The training dataset, validation dataset, and weight vector are provided for you in CSV format under *data* directory.

"P2_X_train.csv" and "P2_X_valid.csv" contains a data matrix of shape $n \times d$ and "P2_y_train.csv" and "P2_y_valid.csv" include target variables for training and validation, respectively. Also, "P2_w.csv" contains ground weights, which have generated target values.

1. Using the provided training dataset, train three models: (a) linear regression without regularization (ordinary), (b) linear regression with L_2 regularization (ridge), and (c) linear regression with L_1 regularization (lasso). For (b) and (c), use the validation dataset to select a regularization coefficient by using grid-search over the range $(0, 5)$ with the precision of 10^{-2}. For obtaining weights in each model, you can use any optimization method you prefer as long as it converges.

   Plot estimated weights of each model and the ground weights in a plot similar to 2; x-axis shows weight components, and y-axis is their value. Use unique markers for each model to distinguish them.

   - Comment on the estimated weights.

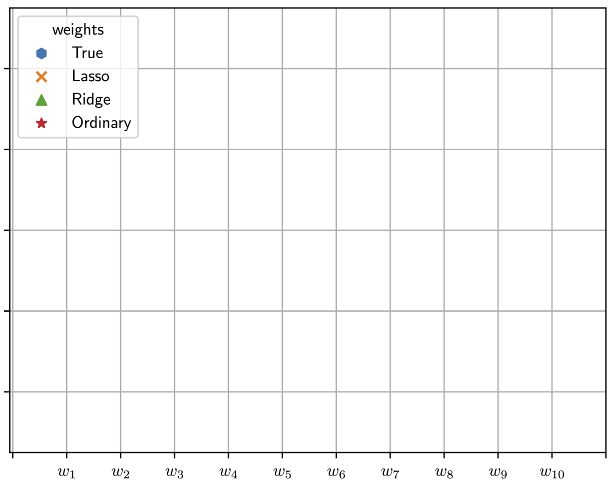

Figure 2: Plot template.

2. Train lasso with different regularization coefficients from 0 to 1 with stepsize of 10^{-2}. Define true positive rate (TPR) as the percentage of non-zero weight components estimated as non-zero correctly, and false positive rate (FPR) as the percentage of zero weight components estimated as non-zero incorrectly. Plot TPR and FPR in a single plot as a function of the regularization coefficient.

- Explain the connection between TPR and FPR with the lasso regularization coefficient.

Problem 3: Bayesian Logistic Regression

This problem is based on section 8.4 Bayesian logistic regression of the book "Machine Learning: A Probabilistic Perspective" by Kevin Murphy, 2012. You can read more about it in the book.

In Bayesian logistic regression, we want to compute the posterior predictive distribution defined as

$$\mathbb{P}(y^*|x^*, D) = \int_w \mathbb{P}(y^*|x^*, w)\mathbb{P}(w|D)dw \tag{1}$$
$$\propto \int_w \mathbb{P}(y^*|x^*, w)\mathbb{P}(D|w)\mathbb{P}(w)dw, \tag{2}$$

where, $D = \{(x_i, y_i)\}_{i=1}^n$ is the training dataset and x^* is an unseen data point that we want to classify. In binary logistic regression, $\mathbb{P}(y|x, w)$ is assumed to be a Bernoulli random variable with parameter $\sigma(w^T x)$, where σ is sigmoid function. Assuming a Gaussian prior $\mathbb{P}(w) = \mathcal{N}(0_d, \sigma_w^2 I_d)$, we can't obtain a closed form posterior distribution $\mathbb{P}(w|D)$, unlike the case of linear regression: the integral is intractable and needs to be approximated. One such approximation method is the Laplace approximation, defined as

$$\mathbb{P}(w|D) \approx \mathcal{N}(w|\hat{w}_{\text{MAP}}, H^{-1}),$$

where, $\hat{w}_{\text{MAP}}$ is the mode of $\mathbb{P}(y^*|x^*, w)$ and H is the Hessian matrix of the negative log of $\mathbb{P}(y^*|x^*, w)$ evaluated at $\hat{w}_{\text{MAP}}$ i.e.,

$$\hat{w}_{\text{MAP}} = \underset{w}{\operatorname{argmax}} \log \mathbb{P}(y^*|x^*, w)$$
$$H = -\nabla_w \nabla_w \log \mathbb{P}(y^*|x^*, w)\Big|_{w=\hat{w}_{\text{MAP}}}.$$

We can find $\hat{w}_{\text{MAP}}$ with gradient ascent and show that $H = X^T F X + \frac{2}{\sigma_d^2} I_d$, where X is the data matrix and F is a diagonal matrix with diagonal entries $F_{ii} = \sigma(\hat{w}_{\text{MAP}}^T x_i)\big(1 - \sigma(\hat{w}_{\text{MAP}}^T x_i)\big)$.

To compute the integral 1, we use the probit approximation, which is based on the fact that sigmoid function has a similar shape as the Normal CDF. That is $\sigma(t) \approx \Phi(\lambda t)$, where Φ is the Normal CDF and

$\lambda = \sqrt{\frac{\pi}{8}}$. Hence, equation 1 turns into

$$\begin{aligned}
\mathbb{P}(y^* = 1|x^*, D) &= \int_w \sigma(w^T x^*)\mathbb{P}(w|D)dw && (3)\\
&\approx \int_w \sigma(w^T x^*)\mathcal{N}(w|\hat{w}_{\text{MAP}}, H^{-1})dw && \text{(Laplace Approximation)} \quad (4)\\
&\approx \int_w \Phi(\lambda w^T x^*)\mathcal{N}(w|\hat{w}_{\text{MAP}}, H^{-1})dw && \text{(Probit Approximation)} \quad (5)\\
&= \Phi\Big(\frac{\lambda m}{(1+\lambda^2 v)^{\frac{1}{2}}}\Big) && (6)\\
&\approx \sigma\Big(\kappa(v)m\Big). && \text{(Probit Approximation)} \quad (7)
\end{aligned}$$

where, $m = \hat{w}_{\text{MAP}}^T x^*$, $v = \text{Var}[w^T x^*] = x^{*T}(\sigma_w^2 I_d)x^*$ and $\kappa(v) = (1+\pi v)^{-\frac{1}{2}}$.

Training data D is provided for you in the *data* directory: "P3_X.csv" contains 64×2 matrix representing 2d datapoints $\{x_i\}_{i=1}^n$ and "P3_y.csv" contains their corresponding labels $\{y_i\}_{i=1}^n$ where $y_i \in \{0, 1\}$. Also, set $\sigma_w^2 = 1$.

Important. For all plots asked below, bound your plot axes to $(-6, 6)$. To see sample plots, refer to section 8.4 of the book "Machine Learning: A Probabilistic Perspective" by Kevin Murphy, 2012

1. Show that conditional log-likelihood $\mathbb{P}(D_y|D_x, w)$, where D_x is the set of data points and D_y is their labels, can be expressed as

$$\log \mathbb{P}(D_y|D_x, w) = \sum_{i=1}^n w^T x_i y_i - \log(1 + \exp(w^T x_i)) \quad (8)$$

   and so, the un-normalized log posterior can be written as

$$\log \widetilde{\mathbb{P}}(D|w) = \log \mathbb{P}(D_y|D_x, w) + \log \mathcal{N}(w|0, \sigma_w^2). \quad (9)$$

2. Plot D and assign unique colors to datapoints in the same class.

3. Plot contours of 8 and 9 using D and mark their maximum value to obtain $\hat{w}_{\text{MLE}}$ and $\hat{w}_{\text{MAP}}$, respectively. Note that these plots are in w-space.

   - Is $||\underset{w}{\text{argmax}} \log \mathbb{P}(D_y|D_x, w)||$ bounded?
   - How does defining the prior affect the estimated weight in MAP compared to ML?

4. Plot the decision boundary of ML and MAP estimation along with training datapoints in the same plot using $\hat{w}_{\text{MLE}}$ and $\hat{w}_{\text{MAP}}$.

5. Prove equality 6 by showing that

$$\int_{-\infty}^{\infty} \Phi(t)\mathcal{N}(t|\mu, \sigma^2)dt = \Phi(\frac{\mu}{(1+\sigma^2)^{\frac{1}{2}}}).$$

   Hint. You don't need to integrate; think about a linear combination of Gaussian RVs.

6. ML and MAP logistic models measure the uncertainty in their predictions using $P(y^*|x^*, \hat{w}_{\text{MLE}})$ and $P(y^*|x^*, \hat{w}_{\text{MAP}})$, respectively. Note that $\hat{w}_{\text{MLE}}$ and $\hat{w}_{\text{MAP}}$ are the estimated weights in item 3. Bayesian logistic regression with Laplace and probit approximation employs 7 as its confidence for predicting x^* to $y^* = 1$.

   Plot the contours of prediction confidence for each ML, MAP, and Bayesian logistic regression as described above for classifying datapoints in the rectangle $(-6, 6) \times (-6, 6)$.

   - Which model is more certain and which is more uncertain in their predictions?
   - How does increasing σ_w affect the uncertainty of each model?
   - Intuitively explain why Bayesian is more uncertain about regions with fewer data compared to MAP and ML.