Homework #4

RELEASE DATE: 11/25/2021

DUE DATE: 12/09/2021, BEFORE 13:00 on Gradescope

QUESTIONS ARE WELCOMED ON THE NTU COOL FORUM.

You will use Gradescope to upload your choices and your scanned/printed solutions. For problems marked with (*), please follow the guidelines on the course website and upload your source code to Gradescope as well. Any programming language/platform is allowed.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

> This homework set comes with 16 problems and a total of 400 points. For each problem, there is one correct choice. If you choose the correct answer, you get 20 points; if you choose an incorrect answer, you get 0 points. For four of the secretly-selected problems, the TAs will grade your detailed solution in terms of the written explanations and/or code based on how logical/clear your solution is. Each of the four problems graded by the TAs counts as additional 20 points (in addition to the correct/incorrect choices you made). In general, each homework (except homework 0) is of a total of 400 points.

Regularization

1. Consider the augmented error

$$E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

$E_{in} = \|Xw - y\|^2$

$E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} w^T w$

with some $\lambda > 0$. Optimize the error by gradient descent with η as the learning rate. That is,

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta\nabla E_{\text{aug}}(\mathbf{w}(t))$$

The update rule above is equivalent to multiplying $\mathbf{w}(t)$ by a shrinkage factor s before updating with the negative gradient of E_{in}

$$\mathbf{w}(t+1) \leftarrow s \cdot \mathbf{w}(t) - \eta\nabla E_{\text{in}}(\mathbf{w}(t))$$

What is s? Choose the correct answer; explain your answer.

C

[a] $1 - \frac{\eta\lambda}{N}$

[b] $1 - \frac{2\lambda}{N}$

[c] $1 - \frac{2\eta\lambda}{N}$

[d] $1 - \frac{2\eta}{N}$

[e] none of the other choices

$\nabla E_{in}(w) = 2X^TXw - 2X^Ty$

$\nabla E_{aug}(w) = \nabla E_{in}(w) + 2\frac{\lambda}{N}w$

$= 2X^TXw - 2X^Ty + 2\frac{\lambda}{N}w$

$w(t+1) = w(t) - \eta \nabla aug(w(t))$

$= w(t) - \eta(\nabla E_{in}(w(t)) + 2\frac{\lambda}{N}w(t))$

$= (1 - 2\eta\frac{\lambda}{N})w(t) - \eta\nabla E_{in}(w)$

$s = 1 - 2\eta\frac{\lambda}{N}$

2. Consider N "labels" $\{y_n\}_{n=1}^N$ with each $y_n \in \mathbb{R}$. Then, solve the following one-variable regularized problem:

$$\min_{w\in\mathbb{R}} \frac{1}{N}\sum_{n=1}^{N}(w-y_n)^2 + \frac{\lambda}{N}w^2.$$

If the optimal solution to the problem above is w^*, it can be shown that w^* is also the optimal solution of

$$\min_{w\in\mathbb{R}} \frac{1}{N}\sum_{n=1}^{N}(w-y_n)^2 \text{ subject to } w^2 \le C$$

with $C = (w^*)^2$. This allows us to express the relationship between C in the constrained optimization problem and λ in the augmented optimization problem for any $\lambda > 0$. What is the relationship? Choose the correct answer; explain your answer.

b

[a] $C = \left(\frac{\sum_{n=1}^{N} y_n}{N + N\lambda}\right)^2$

[b] $C = \left(\frac{\sum_{n=1}^{N} y_n}{N + \lambda}\right)^2$

[c] $C = \left(\frac{\sum_{n=1}^{N} y_n^2}{\sum_{n=1}^{N} y_n + \lambda}\right)^2$

[d] $C = \left(\frac{\sum_{n=1}^{N} y_n^2}{N + \lambda}\right)$

[e] none of the other choices

$\frac{1}{N}\sum_{n=1}^{N}(w^2+y_n^2-2wy_n)$

$\frac{1}{N}\sum_{n=1}^{N}(y_n^2-2wy_n) + \frac{\lambda+N}{N}w^2$

$\frac{-1}{N}\cdot 2\sum_{n=1}^{N} y_n + 2\cdot\frac{\lambda+N}{N}w$

$C = \left(\frac{\sum_{n=1}^{N} y_n}{N+\lambda}\right)^2$

(*Note: All the choices hint you that a smaller λ corresponds to a bigger C.*)

3. Scaling can affect regularization. Consider a data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Define $\Phi(\mathbf{x}) = \mathrm{V}\mathbf{x}$ where V is a diagonal matrix with the i-th diagonal component storing a *positive* value to scale the i-th feature. Now, conduct L2-regularized linear regression with the transformed data $\{(\Phi(\mathbf{x}_n), y_n)\}_{n=1}^N$.

$$\min_{\tilde{\mathbf{w}}\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_n) - y_n)^2 + \frac{\lambda}{N}(\tilde{\mathbf{w}}^T\tilde{\mathbf{w}})$$

The problem is equivalent to the following regularized linear regression problem on the original data with a different regularizer.

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 + \frac{\lambda}{N}(\mathbf{w}^T\mathrm{U}\mathbf{w})$$

d

What is U? Choose the correct answer; explain your answer.

[a] V

[b] V^2

[c] V^{-1}

[d] $(\mathrm{V}^{-1})^2$

[e] none of the other choices

∵ $\phi(x) = Vx$

$\frac{1}{N}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\frac{\lambda}{N}(w^TUw)$

$w^Tx_n = w^TV^{-1}\underline{Vx_n} = w^TV^{-1}\phi(x)$

$\tilde{w} = V^{-1}w$

V is diagonal matrix, therefore $(V^{-1})^T = V^{-1}$

$U = V^{-1}\cdot V^{-1} = (V^{-1})^2$

4. Noise might seem to be an absolute evil at first glance. Can we add noise to combat overfitting, much like how we get vaccinated? Consider a *noisy transform* $\Phi(\mathbf{x}) = \mathbf{x} + \boldsymbol{\epsilon}$ that adds a noise vector $\boldsymbol{\epsilon} = (\epsilon_0, \cdots, \epsilon_d)$ to $\mathbf{x}$. Assume that the noise vector is generated i.i.d. from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \sigma^2 I)$. Consider the following optimization problem that minimizes the expected E_{in} of the transformed data:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\Phi(\mathbf{x}_n) - y_n)^2\right]$$

The problem is actually equivalent to L2-regularized linear regression with some λ.

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 + \frac{\lambda}{N}||\mathbf{w}||_2^2$$

What is λ? Choose the correct answer; explain your answer.

a

[a] $N\sigma^2$

[b] $2N\sigma^2$

[c] $\frac{N}{2}\sigma^2$

[d] $\frac{N}{\sigma^2}$

[e] none of the other choices

$$\frac{1}{N}\sum_{n=1}^{N}(w^T(x_n+\epsilon)-y_n)^2 + \frac{\lambda}{N}||w||_2^2$$

every element in $\epsilon \sim N(0,\sigma^2 I)$

$w_{reg} = (X^TX+\lambda I)^{-1}X^Ty$

$w = (X^TX)^{-1}X^Ty$

we get $\lambda = \sigma^2 N$

5. Additive smoothing (`https://en.wikipedia.org/wiki/Additive_smoothing`) is a simple yet useful technique in estimating discrete probabilities. Consider the technique for estimating the head probability of a coin. Let $y_1, y_2, \ldots, y_N$ denotes the flip results from a coin, with $y_n = 1$ represents a head and $y_n = 0$ represents a tail. Additive smoothing adds (αk) "virtual flips", with α of them being head and the other $(k-1)\alpha$ being tail. Then, the head probability is estimated by

$$\frac{(\sum_{n=1}^{N} y_n) + \alpha}{N + \alpha k}$$

The estimate can be viewed as the optimal solution of

$$\min_{y\in\mathbb{R}} \frac{1}{N}\sum_{n=1}^{N}(y - y_n)^2 + \frac{\alpha k}{N}\Omega(y),$$

where $\Omega(y)$ is a "regularizer" to this estimation problem. What is $\Omega(y)$? Choose the correct answer; explain your answer.

d

$$\frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2 + \frac{\alpha k}{N}\Omega(y), \quad y = \frac{(\sum_{n=1}^{N} y_n)+\alpha}{N+\alpha k}$$

let $\Omega(y) = (y+z)^2$

[a] $(y+k)^2$

[b] $(y+\frac{1}{k})^2$

[c] $(y-k)^2$

[d] $(y-\frac{1}{k})^2$

[e] none of the other choices

derivation:

6. When performing L2-regularization on any twice-differentiable $E_{\text{in}}(\mathbf{w})$ for some linear model of interest, the optimization problem can be written as:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \quad E_{\text{aug}}(\mathbf{w})$$
$$\text{subject to} \quad E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}||\mathbf{w}||_2^2$$

Suppose $\mathbf{w}^*$ is the minimizer of $E_{\text{in}}(\mathbf{w})$. That is, $\nabla E_{\text{in}}(\mathbf{w}^*) = \mathbf{0}$. Take the second-order Taylor's expansion of $E_{\text{in}}(\mathbf{w})$ around $\mathbf{w}^*$, we can approximate $E_{\text{in}}(\mathbf{w})$ by

$$\tilde{E}_{\text{in}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}^*) + \underbrace{(\mathbf{w}-\mathbf{w}^*)^T\nabla E_{\text{in}}(\mathbf{w}^*)}_{0} + \frac{1}{2}(\mathbf{w}-\mathbf{w}^*)^T \ \mathrm{H} \ (\mathbf{w}-\mathbf{w}^*)$$

where $\mathrm{H} \in \mathbb{R}^{(d+1)\times(d+1)}$ is some Hessian matrix. Then, $E_{\text{aug}}(\mathbf{w})$ can be approximated by

$$\tilde{E}_{\text{aug}}(\mathbf{w}) = \tilde{E}_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\|\mathbf{w}\|^2.$$

b

Which of the following is the minimizer of $\tilde{E}_{\text{aug}}(\mathbf{w})$? Choose the correct answer; explain your answer.

the $\tilde{E}_{aug}(w) = \tilde{E}_{in}(w) + \frac{\lambda}{N}||w||^2$

$= \underset{||Xw^*-y||^2}{\underline{E_{in}(w^*)}} + \frac{1}{2}(w-w^*)^T H (w-w^*) + \frac{\lambda}{N}||w||^2$

$H\tilde{E}_{aug}(w) + \frac{2\lambda}{N}\tilde{E}_{aug}(w) = Hw^*$

so $\tilde{E}_{aug}(w) = (H + \frac{2\lambda}{N} I)^{-1} H w^*$

[a] $(\mathrm{H} + \lambda\mathrm{I})^{-1}\mathrm{H}\mathbf{w}^*$

[b] $(\mathrm{H} + \frac{2\lambda}{N}\mathrm{I})^{-1}\mathrm{H}\mathbf{w}^*$

[c] $(\mathrm{H}^T\mathrm{H} + \lambda\mathrm{I})^{-1}\mathrm{H}^T\mathrm{H}\mathbf{w}^*$

[d] $(\mathrm{H} + \frac{\lambda}{N}2\mathrm{I})^{-1}\mathrm{H}\mathbf{w}^*$

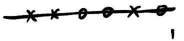
[e] none of the other choices.

(*Note: The result demonstrates the difference between the minimizers of E_{in} and $\tilde{E}_{\text{aug}}$*)

Validation

7. Consider a binary classification algorithm $\mathcal{A}_{\text{minority}}$, which returns a constant classifier that always predicts the minority class (i.e., the class with fewer instances in the data set that it sees). As you can imagine, the returned classifier is the worst-E_{in} one among all constant classifiers. For a binary classification data set with N positive examples and N negative examples, what is $E_{\text{loocv}}(\mathcal{A}_{\text{minority}})$? Choose the correct answer; explain your answer.

$E_{loocv}(A_{minority})$



the worst $1-\frac{1}{N}$

d

$E_{loocv}(A_{minority})$

N positive

N negative

$A_{minority}$ for binary classification

N=1, we get 0 so $\frac{N-1}{N}$ is correct in special cases

[a] 0

[b] $1/N$

[c] $1/2$

[d] $(N-1)/N$

[e] 1

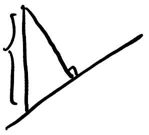
8. You are given three data points: $(x_1, y_1) = (2, 0), (x_2, y_2) = (\rho, 2), (x_3, y_3) = (-2, 0)$ with $\rho \geq 0$, and a choice between two models: constant (all hypotheses are of the form $h(x) = w_0$) and linear (all hypotheses are of the form $h(x) = w_0 + w_1 x$). For which value of ρ would the two models be tied using leave-one-out cross-validation with the squared error measure? Choose the closest answer; explain your answer.

$(x_1, y_1) = (2, 0)$

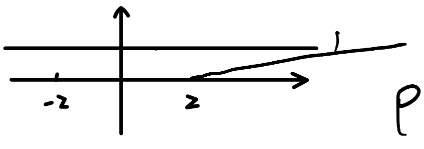
$(x_2, y_2) = (\rho, 2)$

$(x_3, y_3) = (-2, 0)$ $\quad h(x) = w_0 + w_1 x$

this is the calculated distance



c



$\rho \approx 8.5$

[a] 6.5

[b] 7.5

[c] 8.5

[d] 9.5

[e] 10.5

9. For N labels $y_1, y_2, \ldots, y_N$ generated i.i.d. from some distribution of mean 0 and variance σ^2. Partition the first $N-K$ labels to be the training data, and the last K labels to be the validation data. If we "estimate" the mean by the "target" of 0, the expected validation error

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)^2\right),$$

where the expectation is taken on the process of generating N labels, is simply σ^2 by the independence of the y_n's. Now, assume that we take the average on the first $N-K$ examples to estimate the mean instead. That is,

$$\bar{y} = \frac{1}{N-K}\sum_{n=1}^{N-K} y_n$$

d

What is the expected validation error

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\bar{y})^2\right)?$$

Choose the correct answer; explain your answer.

N-K train
K validation

$$\frac{1}{K}+\frac{1}{N-K}=\frac{N}{K(N-K)}$$

[a] σ^2

[b] $\sigma^2 + \frac{1}{N-K}\sigma^2$

[c] $\frac{1}{K(N-K)}\sigma^2$ $\left[1+\left(\frac{1}{K}+\frac{1}{N-K}\right)\frac{1}{N}\right]\sigma^2$

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\bar{y})^2\right)=\sigma^2\left[1+\left(\frac{1}{K}+\frac{1}{N-K}\right)\frac{1}{N}\right]$$

[d] $\sigma^2 + \frac{1}{K(N-K)}\sigma^2$

[e] none of the other choices

Learning Principles

10. In Lecture 9, we talked about the probability to fit data perfectly when the labels are random. For instance, page 5 of Lecture 9 shows that the probability of fitting the data perfectly with positive rays is $\frac{N+1}{2^N}$. Consider 4 different points on the unit circle in $\mathbb{R}^2$ as input vectors $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$, and a 2D perceptron model that minimizes $E_{\text{in}}(\mathbf{w})$ (0/1 error) to the lowest possible value. One way to measure the power of the model is to consider four random labels y_1, y_2, y_3, y_4, each in ±1 and generated by i.i.d. fair coin flips, and then compute

$$\mathbb{E}_{y_1,y_2,y_3,y_4}\left(\min_{\mathbf{w}\in\mathbb{R}^{2+1}} E_{\text{in}}(\mathbf{w})\right).$$

For a perfect fitting, $\min E_{\text{in}}(\mathbf{w})$ will be 0; for a less perfect fitting (when the data is not linearly separable), $\min E_{\text{in}}(\mathbf{w})$ will be some non-zero value. The expectation above averages over all 16 possible combinations of y_1, y_2, y_3, y_4. What is the value of the expectation? Choose the correct answer; explain your answer.

a

when it is impartial

4 points have 2^4 groups

the expectation : $\frac{2}{2^4} \times \underline{\frac{1}{4}} = \frac{1}{32}$

error rate

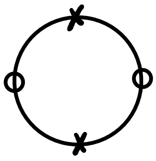
[a] 1/32

[b] 2/32

[c] 3/32

[d] 5/32

[e] 8/32



case 1, exchange ⇒ case 2

(*Note: It can be shown that* 1 *minus twice the expected value above is the same as the so-called empirical Rademacher complexity of 2D perceptrons. Rademacher complexity, similar to the VC dimension, is another tool to measure the complexity of a hypothesis set. If a hypothesis set shatters some data points, zero* E_{in} *can always be achieved and thus Rademacher complexity is* 1*; if a hypothesis set cannot shatter some data points, Rademacher complexity provides a soft measure of how "perfect" the hypothesis set is.*)

11. Consider a binary classifier g such that

$$\begin{aligned} P(g(\mathbf{x}) = -1 | y = +1) &= \epsilon_+ \\ P(g(\mathbf{x}) = +1 | y = -1) &= \epsilon_-. \end{aligned}$$

When deploying the classifier to a test distribution of $P(y = +1) = P(y = -1) = 1/2$, we get $E_{\text{out}}(g) = \frac{1}{2}\epsilon_+ + \frac{1}{2}\epsilon_-$. Now, if we deploy the classifier to another test distribution $P(y = +1) = p$ instead of $1/2$, the $E_{\text{out}}(g)$ under this test distribution will then change to a different value. Note that under this test distribution, a constant classifier g_- that always predicts -1 will suffer from $E_{\text{out}}(g_-) = p$ as it errors on all the positive examples. At what p, if its value is between $[0, 1]$, will our binary classifier g be as good as (or as bad as) the constant classifier g_- in terms of E_{out}? Choose the correct answer; explain your answer.

b

[a] $p = \frac{\epsilon_+}{\epsilon_- - \epsilon_+ + 1}$

[b] $p = \frac{\epsilon_-}{\epsilon_- - \epsilon_+ + 1}$

[c] $p = \frac{1-\epsilon_+}{\epsilon_- - \epsilon_+ + 1}$

[d] $p = \frac{1-\epsilon_-}{\epsilon_- - \epsilon_+ + 1}$

[e] none of the other choices

$E_{out}(g) = \frac{\epsilon_- + \epsilon_+}{2}$

$E_{out}(g^-) = p$

$(p-1)\frac{\epsilon_-}{2} = \frac{1-\epsilon_+}{2}$

$p\epsilon_- = \epsilon_- - \epsilon_+ + 1$

so $p = \frac{\epsilon_-}{\epsilon_- - \epsilon_+ + 1}$

Experiments with Regularized Logistic Regression

Consider L2-regularized logistic regression with third-order polynomial transformation.

$$\mathbf{w}_\lambda = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{\lambda}{N}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{n=1}^{N} \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{\Phi}_3(\mathbf{x}_n))),$$

Here $\mathbf{\Phi}_3$ is the third-order polynomial transformation introduced in Lecture 6 (with $Q = 3$), defined as

$$\mathbf{\Phi}_3(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1^2, x_1x_2, \ldots, x_1x_d, x_2^2, x_2x_3, \ldots, x_2x_d, \ldots, x_d^2, x_1^3, x_1^2x_2, \ldots, x_d^3)$$

Next, we will take the following file as our training data set $\mathcal{D}$:

`http://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw4/hw4_train.dat`

and the following file as our test data set for evaluating E_{out}:

`http://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw4/hw4_test.dat`

We call the algorithm for solving the problem above as $\mathcal{A}_\lambda$. The problem guides you to use LIBLINEAR (`https://www.csie.ntu.edu.tw/~cjlin/liblinear/`), a machine learning packaged developed in our university, to solve this problem. In addition to using the default options, what you need to do when running LIBLINEAR are

- set option `-s 0`, which corresponds to solving regularized logistic regression
- set option `-c C`, with a parameter value of `C` calculated from the λ that you want to use; read `README` of the software package to figure out how `C` and your λ should relate to each other
- set option `-e 0.000001`, which corresponds to getting a solution that is really really close to the optimal solution

LIBLINEAR can be called from the command line or from major programming languages like python. If you run LIBLINEAR in the command line, please include screenshots of the commands/results; if you run LIBLINEAR from any programming language, please include screenshots of your code.

We will consider the data set as a *binary classification problem* and take the "regression for classification" approach with regularized logistic regression (see Page 6 of Lecture 10). So please evaluate all errors below with the 0/1 error.

12. Select the best λ^* *in a cheating manner* as

$$\underset{\log_{10}\lambda\in\{-4,-2,0,2,4\}}{\operatorname{argmin}} E_{\text{out}}(\mathbf{w}_\lambda).$$

Break the tie, if any, by selecting the largest λ. What is $\log_{10}(\lambda^*)$? Choose the closest answer; provide your command/code.

b

[a] -4

[b] -2

[c] 0 base on code

[d] 2

[e] 4

13. Select the best λ^* as

$$\underset{\log_{10}\lambda\in\{-4,-2,0,2,4\}}{\operatorname{argmin}} E_{\text{in}}(\mathbf{w}_\lambda).$$

Break the tie, if any, by selecting the largest λ. What is $\log_{10}(\lambda^*)$? Choose the closest answer; provide your command/code.

b

[a] -4

[b] -2 base on code

[c] 0

[d] 2

[e] 4

14. Now split the given training examples in $\mathcal{D}$ to two sets: the first 120 examples as $\mathcal{D}_{\text{train}}$ and 80 as $\mathcal{D}_{\text{val}}$. (*Ideally, you should randomly do the* 120/80 *split. Because the given examples are already randomly permuted, however, we would use a fixed split for the purpose of this problem*). Run $\mathcal{A}_\lambda$ on *only* $\mathcal{D}_{\text{train}}$ to get $\mathbf{w}_\lambda^-$ (the weight vector within the g^- returned), and validate $\mathbf{w}_\lambda^-$ with $\mathcal{D}_{\text{val}}$ to get $E_{\text{val}}(\mathbf{w}_\lambda^-)$. Select the best λ^* as

$$\underset{\log_{10}\lambda\in\{-4,-2,0,2,4\}}{\operatorname{argmin}} E_{\text{val}}(\mathbf{w}_\lambda^-).$$

Break the tie, if any, by selecting the largest λ. Then, estimate $E_{\text{out}}(\mathbf{w}_{\lambda^*}^-)$ with the test set. What is the value of $E_{\text{out}}(\mathbf{w}_{\lambda^*}^-)$? Choose the closest answer; provide your command/code.

c

[a] 0.081

[b] 0.078

[c] 0.075 base on code

[d] 0.072

[e] 0.069

15. For the λ_* selected in the previous problem, compute $\mathbf{w}_{\lambda^*}$ by running $\mathcal{A}_{\lambda^*}$ with the full training set $\mathcal{D}$. Then, estimate $E_{\text{out}}(\mathbf{w}_{\lambda^*})$ with the test set. What is the value of $E_{\text{out}}(\mathbf{w}_{\lambda^*})$? Choose the closest answer; provide your command/code.

d

[a] 0.081

[b] 0.078

[c] 0.075 base on code

[d] 0.072

[e] 0.069

16. Now split the given training examples in $\mathcal{D}$ to five folds, the first 40 being fold 1, the next 40 being fold 2, and so on. Again, we take a fixed split because the given examples are already randomly permuted. Select the best λ^* as

$$\mathop{\mathrm{argmin}}_{\log_{10}\lambda\in\{-4,-2,0,2,4\}} E_{\mathrm{cv}}(\mathcal{A}_\lambda).$$

Break the tie, if any, by selecting the largest λ. What is the value of $E_{\mathrm{cv}}(\mathcal{A}_{\lambda^*})$ Choose the closest answer; provide your command/code.

[a] 0.07

[b] 0.08

[✓ **[c]** 0.09] base on code

[d] 0.10

[e] 0.11

HW4

程式Python語言（Jupiter notebook），並使用了LIBLINEAR，如下

```
import numpy as np
from sklearn preprocessing import PolynomialFeatures
def preprocess data
    #維度
    n  d = data
    #剝離X
    X = data      -1
    #偏置項1
    #X = np.c_[np.ones(n), X]
    #剝離y
    y = data     -1

    return X  y
#將網頁上的資料下載到桌面，存儲到同一個目錄下，打開命名
train = np genfromtxt 'hw4_train.dat.txt'
test = np genfromtxt 'hw4_test.dat.txt'
X_train1  y_train = preprocess train
X_test1  y_test = preprocess test
#X_test
#y_test
reg = PolynomialFeatures degree=3
X_test=reg fit_transform X_test1

reg = PolynomialFeatures degree=3
X_train=reg fit_transform X_train1

X_test
```

```
f = open "hw4_train.dat.txt" "r"    #设置文件对象
for i in range 120
    print f readline    strip
import liblinear
from liblinear liblinearutil import *
import scipy
#Problem 12
#cheating means use data from "Eout"
#c=1/(2*lamda)

#lamda=0.0001
m=train y_test X_test '-s 0 -c 5000 -e 0.000001'
p_labs p_acc p_vals=            y_test X_test m

#lamda=0.01
m=train y_test X_test '-s 0 -c 50 -e 0.000001'
p_labs p_acc p_vals=            y_test X_test m

#lamda=1
m=train y_test X_test '-s 0 -c 0.5 -e 0.000001'
p_labs p_acc p_vals=            y_test X_test m

#lamda=100
m=train y_test X_test '-s 0 -c 0.005 -e 0.000001'
p_labs p_acc p_vals=            y_test X_test m

#lamda=10000
m=train y_test X_test '-s 0 -c 0.00005 -e 0.000001'
p_labs p_acc p_vals=            y_test X_test m
#Problem 13

#lamda=0.0001
```

```
m=trai  (y_tr  in,X_t       '-s 0 -c 5000 -e 0.000001'
                    =

#lamda=0.01
m=trai  (y_tr  in,X_t       '-s 0 -c 50 -e 0.000001'
                    =

#lamda=1
m=trai  (y_tr  in,X_t       '-s 0 -c 0.5 -e 0.000001'
                    =

#lamda=100
m=trai  (y_tr  in,X_t       '-s 0 -c 0.005 -e 0.000001'
                    =

#lamda=10000
m=trai  (y_tr  in,X_t       '-s 0 -c 0.00005 -e 0.000001'
                    =
#Problem 14
#120 train 和 80val
np genfromtxt 'hw4_train.dat.txt' skip_header=0
skip_footer=119
     =                     'hw4_train.dat1.txt' #前120筆
                 =


#lamda=0.0001
  =                         '-s 0 -c 5000 -e 0.000001'
                    =


#lamda=0.01
  =                         '-s 0 -c 50 -e 0.000001'
                    =
#lamda=1
   =                                  '-s 0 -c 0.5 -e 0.000001'
                              =

#lamda=100
   =                                  '-s 0 -c 0.005 -e 0.000001'
                              =

#lamda=10000
   =                                  '-s 0 -c 0.00005 -e 0.000001'
                              =
#120 train 和 80val
np genfromtxt 'hw4_train.dat.txt' skip_header=120
skip_footer=0
       =                             'hw4_train.dat2.txt' #後80筆
                         =

#lamda=0.0001
   =                                  '-s 0 -c 5000 -e 0.000001'
                              =

#lamda=0.01
   =                                  '-s 0 -c 50 -e 0.000001'
                              =

#lamda=1
   =                                  '-s 0 -c 0.5 -e 0.000001'
                              =

#lamda=100
   =                                  '-s 0 -c 0.005 -e 0.000001'
                              =
#lamda=10000
  =                          '-s 0 -c 0.00005 -e 0.000001'
                        =
#lamda=0.01
  =                        '-s 0 -c 0.05 -e 0.000001'
                        =
#Problem 15
#convert from Dtrain to D

#lamda=0.0001
  =                          '-s 0 -c 5000 -e 0.000001'
                        =


#lamda=0.01
  =                          '-s 0 -c 50 -e 0.000001'
                        =


#lamda=1
  =                          '-s 0 -c 0.5 -e 0.000001'
                        =


#lamda=100
  =                          '-s 0 -c 0.005 -e 0.000001'
                        =


#lamda=10000
  =                          '-s 0 -c 0.00005 -e 0.000001'
                        =


#Problem 16
#for lamda=0.0001,0.01,1,100,10000
#40+40+40+40+40=5*40=200
     =                     'hw4_train.dat.txt' skip_header=0
```

```
skip_footer=160
                =

    =                   degree=3
       =

#train2=np.genfromtxt('hw4_train.dat.txt',skip_header=4
0, skip_footer=120)
#X_train, y_train = preprocess(train2)
#train3=np.genfromtxt('hw4_train.dat.txt',skip_header=8
0, skip_footer=80)
#X_train, y_train = preprocess(train3)
#train4=np.genfromtxt('hw4_train.dat.txt',skip_header=1
20, skip_footer=40)
#X_train, y_train = preprocess(train4)
#train5=np.genfromtxt('hw4_train.dat.txt',skip_header=1
60, skip_footer=0)
#X_train, y_train = preprocess(train5)

#lamda=0.0001
  =                   '-s 0 -c 5000 -e 0.000001'
                   =


#lamda=0.01
  =                   '-s 0 -c 50 -e 0.000001'
                   =


#lamda=1
  =                   '-s 0 -c 0.5 -e 0.000001'
                   =


#lamda=100
  =                   '-s 0 -c 0.005 -e 0.000001'
                   =
#lamda=10000
  =                          '-s 0 -c 0.00005 -e 0.000001'
                         =
```