Supervised Learning [redacted] – Coursework 2

Due: 02 January 2025

PART I [20%]

Rademacher Complexity of Finite Spaces

1.1 [2 marks]

Claim: For $\overline{X} = \max_i X_i$ and any $\lambda > 0$,

$$\mathbb{E}[\overline{X}] \leq \frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}].$$

Proof:

1. We begin with a set of random variables $X_1, X_2, \ldots, X_m$ and define $\overline{X} = \max_i X_i$. Our goal is to relate the expectation $\mathbb{E}[\overline{X}]$ to $\log \mathbb{E}[e^{\lambda \overline{X}}]$.

2. Consider the function $\phi(z) = e^{\lambda z}$. This function is convex for any $\lambda > 0$. By Jensen's inequality, for a convex function ϕ and a random variable Y, we have $\phi(\mathbb{E}[Y]) \leq \mathbb{E}[\phi(Y)]$.

3. Apply Jensen's inequality with $\phi(z) = e^{\lambda z}$ and $Y = \overline{X}$:

$$e^{\lambda \mathbb{E}[\overline{X}]} \leq \mathbb{E}[e^{\lambda \overline{X}}].$$

4. Take the natural logarithm of both sides. Since the logarithm is a strictly increasing function, the inequality direction does not change:

$$\log\left(e^{\lambda \mathbb{E}[\overline{X}]}\right) \leq \log \mathbb{E}[e^{\lambda \overline{X}}].$$

5. The left-hand side simplifies to $\lambda \mathbb{E}[\overline{X}]$:

$$\lambda \mathbb{E}[\overline{X}] \leq \log \mathbb{E}[e^{\lambda \overline{X}}].$$

6. Divide both sides of the inequality by $\lambda > 0$:

$$\mathbb{E}[\overline{X}] \leq \frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}].$$

1.2 [5 marks]

Claim:

$$\frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}] \leq \frac{1}{\lambda} \log(m) + \lambda \frac{(b-a)^2}{8}.$$

Proof:

1. Begin with $\overline{X} = \max_{1 \leq i \leq m} X_i$. We want to bound $\log \mathbb{E}[e^{\lambda \overline{X}}]$ in a way that introduces a $\log(m)$ term and a quadratic term in λ.

2. Since $\overline{X}$ is the maximum of the X_i's, write:

$$e^{\lambda \overline{X}} = \max_{1 \leq i \leq m} e^{\lambda X_i}.$$

The maximum of a collection of nonnegative numbers is always less than or equal to their sum:

$$\max_{1 \leq i \leq m} e^{\lambda X_i} \leq \sum_{i=1}^{m} e^{\lambda X_i}.$$

3. Take expectations on both sides:

$$\mathbb{E}[e^{\lambda \overline{X}}] \leq \mathbb{E}\left[\sum_{i=1}^{m} e^{\lambda X_i}\right].$$

By linearity of expectation:

$$\mathbb{E}[e^{\lambda \overline{X}}] \leq \sum_{i=1}^{m} \mathbb{E}[e^{\lambda X_i}].$$

4. Now we need an upper bound for each $\mathbb{E}[e^{\lambda X_i}]$. The random variables X_i are centered ($\mathbb{E}[X_i] = 0$) and bounded in $[a, b]$. This is precisely the scenario where Hoeffding's lemma applies.
Hoeffding's Lemma: If Y is a random variable with $\mathbb{E}[Y] = 0$ and $Y \in [A, B]$ almost surely, then:

$$\mathbb{E}[e^{\lambda Y}] \leq \exp\left(\frac{\lambda^2 (B-A)^2}{8}\right) \quad \text{for all } \lambda.$$

In our case, each X_i satisfies $\mathbb{E}[X_i] = 0$ and $X_i \in [a, b]$. Setting $Y = X_i$, $A = a$, and $B = b$, we directly obtain:

$$\mathbb{E}[e^{\lambda X_i}] \leq \exp\left(\frac{\lambda^2 (b-a)^2}{8}\right).$$

5. Since this upper bound does not depend on i, it holds for every $i = 1, \ldots, m$. Therefore:

$$\mathbb{E}[e^{\lambda \overline{X}}] \leq \sum_{i=1}^{m} \mathbb{E}[e^{\lambda X_i}] \leq \sum_{i=1}^{m} \exp\left(\frac{\lambda^2 (b-a)^2}{8}\right).$$

6. Each term in the sum is identical, so the sum of m identical terms is just m times that term:

$$\mathbb{E}[e^{\lambda \overline{X}}] \leq m \cdot \exp\left(\frac{\lambda^2 (b-a)^2}{8}\right).$$

7. Take the natural logarithm:

$$\log \mathbb{E}[e^{\lambda \overline{X}}] \leq \log(m) + \frac{\lambda^2 (b-a)^2}{8}.$$

8. Finally, divide by $\lambda > 0$:

$$\frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}] \leq \frac{\log(m)}{\lambda} + \lambda \frac{(b-a)^2}{8}.$$

—

1.3 [3 marks]

Claim: By choosing λ optimally, we obtain:

$$\mathbb{E}\left[\max_{i=1,\ldots,m} X_i\right] \leq \frac{b-a}{2} \sqrt{2 \log(m)}.$$

Proof:
1. From the previous result, we have:

$$\frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}] \leq \frac{\log(m)}{\lambda} + \lambda \frac{(b-a)^2}{8},$$

where $\overline{X} = \max_{1 \leq i \leq m} X_i$.
2. From Section 1.1, we know:

$$\mathbb{E}[\overline{X}] \leq \frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}].$$

Substituting the previous upper bound for $\frac{1}{\lambda} \log \mathbb{E}[e^{\lambda \overline{X}}]$:

$$\mathbb{E}[\overline{X}] \leq \frac{\log(m)}{\lambda} + \lambda \frac{(b-a)^2}{8}.$$

Our task now is to pick $\lambda > 0$ to minimize the right-hand side (RHS).

3. Define:

$$f(\lambda) = \frac{\log(m)}{\lambda} + \lambda\frac{(b-a)^2}{8}.$$

Notice the opposing behaviors: - When λ is very small, $\frac{\log(m)}{\lambda}$ is large. - When λ is very large, $\lambda\frac{(b-a)^2}{8}$ is large. Thus, there must be a balance point where $f(\lambda)$ attains its minimum.
4. *Differentiation step:* To find the optimal λ, differentiate $f(\lambda)$ with respect to λ:

$$f'(\lambda) = -\frac{\log(m)}{\lambda^2} + \frac{(b-a)^2}{8}.$$

- For the term $\frac{\log(m)}{\lambda}$: Treat $\frac{\log(m)}{\lambda}$ as $\log(m)\cdot\lambda^{-1}$. Differentiating with respect to λ:

$$\frac{d}{d\lambda}\left(\frac{\log(m)}{\lambda}\right) = \log(m)\cdot(-\lambda^{-2}) = -\frac{\log(m)}{\lambda^2}.$$

- For the term $\lambda\frac{(b-a)^2}{8}$: Since $\frac{(b-a)^2}{8}$ is a constant, and the term is linear in λ, its derivative is simply:

$$\frac{d}{d\lambda}\left(\lambda\frac{(b-a)^2}{8}\right) = \frac{(b-a)^2}{8}.$$

Putting these results together:

$$f'(\lambda) = -\frac{\log(m)}{\lambda^2} + \frac{(b-a)^2}{8}.$$

Setting $f'(\lambda) = 0$:

$$-\frac{\log(m)}{\lambda^2} + \frac{(b-a)^2}{8} = 0.$$

Rearrange to solve for λ^2:

$$\frac{(b-a)^2}{8} = \frac{\log(m)}{\lambda^2} \implies \lambda^2 = \frac{8\log(m)}{(b-a)^2}.$$

Taking the positive root (since $\lambda > 0$):

$$\lambda = \frac{\sqrt{8\log(m)}}{b-a}.$$

5. Substitute $\lambda = \frac{\sqrt{8\log(m)}}{b-a}$ back into the inequality:

$$\mathbb{E}[\overline{X}] \leq \frac{\log(m)}{\frac{\sqrt{8\log(m)}}{b-a}} + \frac{\sqrt{8\log(m)}}{b-a}\cdot\frac{(b-a)^2}{8}.$$

Let's simplify each term carefully, paying attention to the square roots and their manipulation:
- *First term:*

$$\frac{\log(m)}{\frac{\sqrt{8\log(m)}}{b-a}} = \log(m)\cdot\frac{b-a}{\sqrt{8\log(m)}}.$$

Here, $\sqrt{8\log(m)} = \sqrt{8}\sqrt{\log(m)}$. Also, $\log(m)$ can be written as $\sqrt{\log(m)}\cdot\sqrt{\log(m)}$. Thus:

$$\log(m)\cdot\frac{b-a}{\sqrt{8\log(m)}} = (b-a)\frac{\sqrt{\log(m)}\cdot\sqrt{\log(m)}}{\sqrt{8}\sqrt{\log(m)}}.$$

Cancel one $\sqrt{\log(m)}$:

$$= (b-a)\frac{\sqrt{\log(m)}}{\sqrt{8}} = (b-a)\sqrt{\frac{\log(m)}{8}}.$$

- *Second term:*

$$\frac{\sqrt{8\log(m)}}{b-a}\cdot\frac{(b-a)^2}{8} = \frac{(b-a)^2}{8}\cdot\frac{\sqrt{8\log(m)}}{b-a}.$$

Cancel one $(b-a)$:

$$= (b-a)\frac{\sqrt{8\log(m)}}{8}.$$

Since $\sqrt{8\log(m)} = \sqrt{8}\sqrt{\log(m)}$, we have:

$$(b-a)\frac{\sqrt{8\log(m)}}{8} = (b-a)\frac{\sqrt{\log(m)}}{\sqrt{8}} = (b-a)\sqrt{\frac{\log(m)}{8}}.$$

6. Add the two identical terms:

$$\mathbb{E}[\overline{X}] \leq (b-a)\sqrt{\frac{\log(m)}{8}} + (b-a)\sqrt{\frac{\log(m)}{8}} = 2(b-a)\sqrt{\frac{\log(m)}{8}}.$$

7. *Further root simplification:* We have:

$$2(b-a)\sqrt{\frac{\log(m)}{8}}.$$

Note that $\sqrt{\frac{1}{8}} = \frac{1}{\sqrt{8}} = \frac{1}{2\sqrt{2}}$. Therefore:

$$2(b-a)\sqrt{\frac{\log(m)}{8}} = 2(b-a)\frac{\sqrt{\log(m)}}{2\sqrt{2}} = (b-a)\frac{\sqrt{\log(m)}}{\sqrt{2}}.$$

This can be written as:

$$(b-a)\sqrt{\frac{\log(m)}{2}}.$$

To show the final desired form $\frac{b-a}{2}\sqrt{2\log(m)}$, notice that:

$$\sqrt{\frac{\log(m)}{2}} = \frac{\sqrt{2\log(m)}}{2}.$$

Substitute this back:

$$\mathbb{E}[\overline{X}] \leq (b-a)\frac{\sqrt{2\log(m)}}{2} = \frac{b-a}{2}\sqrt{2\log(m)}.$$

8. We have arrived at the claimed bound. By choosing $\lambda = \frac{\sqrt{8\log(m)}}{b-a}$, we found the optimal balance that yields the neat final expression $\frac{b-a}{2}\sqrt{2\log(m)}$. This shows how the expected maximum scales with m.

1.4 [3 marks]

Claim: Let $S \subset \mathbb{R}^n$ with $|S| = m$. Then:

$$\mathcal{R}(S) = \mathbb{E}_\sigma\left[\max_{x\in S}\frac{1}{n}\sum_{j=1}^{n}\sigma_j x_j\right] \leq \max_{x\in S}\|x\|_2\frac{\sqrt{2\log(m)}}{n}.$$

Proof:

1. By definition:

$$\mathcal{R}(S) = \mathbb{E}_\sigma\left[\max_{x\in S}\frac{1}{n}\sum_{j=1}^{n}\sigma_j x_j\right],$$

where $\sigma_1, \ldots, \sigma_n$ are independent Rademacher random variables (each σ_j takes values +1 or -1 with probability 1/2).

2. For each fixed vector $x \in S$, define a random variable:

$$Z_x := \frac{1}{n}\sum_{j=1}^{n}\sigma_j x_j.$$

This isolates the random behavior (due to σ_j) from the choice of x. Notice that $\mathbb{E}[Z_x] = 0$ since $\mathbb{E}[\sigma_j] = 0$.

3. Consider the magnitude of Z_x. Since $|\sigma_j x_j| \leq |x_j|$,

$$|Z_x| = \left| \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j \right| \leq \frac{1}{n} \sum_{j=1}^{n} |x_j| \leq \max_j |x_j|.$$

Thus, for each fixed x, the random variable Z_x is bounded by $\pm M_x$, where $M_x = \max_j |x_j|$. This means $Z_x \in [-M_x, M_x]$, so $b - a = 2M_x$.

4. We are taking the maximum over all $x \in S$, which creates a collection of m random variables $\{Z_x : x \in S\}$. Among all vectors in S, the largest component magnitude is $\max_{x \in S, j} |x_j|$. Thus, we can uniformly bound Z_x for all x by taking $b - a \leq 2 \max_{x \in S, j} |x_j|$.

5. From subsection 1.3, we know that for m zero-mean random variables each bounded in an interval of length $b - a$:

$$\mathbb{E}\left[\max_i X_i\right] \leq \frac{b-a}{2} \sqrt{2 \log(m)}.$$

Replacing the index i with $x \in S$ and applying this result to our collection $\{Z_x : x \in S\}$:

$$\mathbb{E}_\sigma \left[\max_{x \in S} Z_x\right] \leq \frac{2 \max_{x,j} |x_j|}{2} \sqrt{2 \log(m)}.$$

The factor of 2 in the numerator and denominator cancels out:

$$\mathbb{E}_\sigma \left[\max_{x \in S} Z_x\right] \leq \max_{x \in S, j} |x_j| \sqrt{2 \log(m)}.$$

6. Since $\max_j |x_j| \leq \|x\|_2$ for any vector x,

$$\mathbb{E}_\sigma \left[\max_{x \in S} Z_x\right] \leq \max_{x \in S} \|x\|_2 \sqrt{2 \log(m)}.$$

7. Recall that $\mathcal{R}(S) = \mathbb{E}_\sigma[\max_{x \in S} Z_x]$ by definition, and Z_x already includes the factor $\frac{1}{n}$. Thus, the final result is:

$$\mathcal{R}(S) \leq \max_{x \in S} \|x\|_2 \frac{\sqrt{2 \log(m)}}{n}.$$

—

1.5 [7 marks]

Claim: For a finite hypothesis class $\mathcal{H}$ with $|\mathcal{H}| = m$, the empirical Rademacher complexity $\mathcal{R}_S(\mathcal{H})$ depends only logarithmically on m. In particular, there exists a bound of the form

$$\mathcal{R}_S(\mathcal{H}) \leq B \sqrt{\frac{2 \log(|\mathcal{H}|)}{n}}$$

for some $B \geq 0$.

Proof:

1. Consider a finite hypothesis class $\mathcal{H}$, $|\mathcal{H}| = m$, and a sample $S = (x_1, \ldots, x_n) \subset \mathcal{X}$.

The empirical Rademacher complexity of $\mathcal{H}$ with respect to S is defined as:

$$\mathcal{R}_S(\mathcal{H}) = \mathbb{E}_\sigma \left[\max_{f \in \mathcal{H}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j f(x_j)\right],$$

where $\sigma_1, \ldots, \sigma_n$ are independent Rademacher variables, each taking values ± 1 with probability 1/2.

2. For each hypothesis $f \in \mathcal{H}$, define a vector in $\mathbb{R}^n$:

$$x_f := (f(x_1), f(x_2), \ldots, f(x_n)).$$

Collecting all such vectors into a set:

$$T := \{x_f : f \in \mathcal{H}\} = \{(f(x_1), \dots, f(x_n)) : f \in \mathcal{H}\}.$$

Since there is a one-to-one correspondence between each $f \in \mathcal{H}$ and a vector $x_f \in T$, we have $|T| = |\mathcal{H}| = m$.
3. Notice that:

$$\max_{f \in \mathcal{H}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j f(x_j) = \max_{x_f \in T} \frac{1}{n} \sum_{j=1}^{n} \sigma_j (x_f)_j.$$

Renaming x_f simply as x for brevity:

$$\max_{f \in \mathcal{H}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j f(x_j) = \max_{x \in T} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j.$$

Therefore:

$$\mathcal{R}_S(\mathcal{H}) = \mathbb{E}_\sigma \left[\max_{x \in T} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j \right] = \mathcal{R}(T).$$

Here, $\mathcal{R}(T)$ is the Rademacher complexity defined for the finite set $T \subset \mathbb{R}^n$.
4. Assume there is a constant B such that:

$$|f(x)| \leq B \quad \text{for all } f \in \mathcal{H} \text{ and all } x \in \mathcal{X}.$$

Given $x = (f(x_1), \dots, f(x_n)) \in T$:

$$\|x\|_2 = \sqrt{\sum_{j=1}^{n} f(x_j)^2} \leq \sqrt{\sum_{j=1}^{n} B^2} = \sqrt{n}B.$$

Thus, every vector in T is bounded in norm by $\sqrt{n}B$.
5. From Subsection 1.4, we know that for a finite set $T \subset \mathbb{R}^n$ with $|T| = m$:

$$\mathcal{R}(T) \leq \max_{x \in T} \|x\|_2 \frac{\sqrt{2\log(m)}}{n}.$$

Substitute $\max_{x \in T} \|x\|_2 \leq \sqrt{n}B$:

$$\mathcal{R}(T) \leq (\sqrt{n}B) \frac{\sqrt{2\log(m)}}{n}.$$

Simplifying:

$$(\sqrt{n}B) \frac{\sqrt{2\log(m)}}{n} = B \frac{\sqrt{2\log(m)}}{\sqrt{n}}.$$

Therefore:

$$\mathcal{R}(T) \leq B \sqrt{\frac{2\log(m)}{n}}.$$

6. Recall that $m = |\mathcal{H}|$ and that $\mathcal{R}_S(\mathcal{H}) = \mathcal{R}(T)$, so:

$$\mathcal{R}_S(\mathcal{H}) \leq B \sqrt{\frac{2\log(|\mathcal{H}|)}{n}}.$$

This shows that the empirical Rademacher complexity depends on $\sqrt{\log(|\mathcal{H}|)}$, hence only logarithmically on the cardinality of $\mathcal{H}$.
—

PART II [40%]

Bayes Decision Rule and Surrogate Approaches

In (binary) classification problems, the classification or "decision" rule is a binary-valued function $c : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{Y} = \{1, -1\}$. The quality of a classification rule can be measured by the misclassification error

$$R(c) = \mathbb{P}_{(x,y)\sim\rho} \left(c(x) \neq y \right)$$

assuming to sample an input-output pair (x, y) according to a distribution ρ on $\mathcal{X} \times \mathcal{Y}$.

2.1 [4 marks]

Let $1_{y\neq y'}$ be the 0-1 loss such that $1_{y\neq y'} = 1$ if $y \neq y'$ and 0 otherwise. Show that the misclassification error corresponds to the expected risk of the 0-1 loss, namely

$$R(c) = \int_{\mathcal{X}\times\mathcal{Y}} 1_{c(x)\neq y} d\rho(x,y),$$

Answer:

Misclassification Error Definition

The misclassification error is defined as the probability that the classifier $c(x)$ assigns an incorrect label to an input x. Formally, it is given by:

$$R(c) = \mathbb{P}_{(x,y)\sim\rho}\left(c(x) \neq y\right).$$

This represents the expectation of the event where the predicted label $c(x)$ does not match the true label y.

Representing Misclassification Error as an Integral

In probability theory, the probability of an event A can be expressed as the expectation of its indicator function $\mathbf{1}_A$:

$$\mathbb{P}(A) = \mathbb{E}[\mathbf{1}_A],$$

where

$$\mathbf{1}_A = \begin{cases} 1, & \text{if } A \text{ occurs,} \\ 0, & \text{otherwise.} \end{cases}$$

Applying this to the event $\{c(x) \neq y\}$, we have:

$$\mathbb{P}_{(x,y)\sim\rho}(c(x) \neq y) = \mathbb{E}_{(x,y)\sim\rho}\left[\mathbf{1}(c(x) \neq y)\right].$$

By the definition of expectation with respect to the distribution ρ, this can be written as an integral over the joint space $\mathcal{X}\times\mathcal{Y}$:

$$R(c) = \int_{\mathcal{X}\times\mathcal{Y}} \mathbf{1}(c(x) \neq y)\, d\rho(x,y).$$

Substituting the 0-1 Loss

The 0-1 loss function is defined as:

$$1_{c(x)\neq y} = \begin{cases} 1, & \text{if } c(x) \neq y, \\ 0, & \text{if } c(x) = y. \end{cases}$$

Notice that the indicator function $\mathbf{1}(c(x) \neq y)$ is equivalent to the 0-1 loss $1_{c(x)\neq y}$. Substituting this into the integral, we obtain:

$$R(c) = \int_{\mathcal{X}\times\mathcal{Y}} 1_{c(x)\neq y}\, d\rho(x,y).$$

Thus, the misclassification error $R(c)$ is exactly the expected risk of the 0-1 loss over the distribution ρ.

(Surrogate Approaches) Since the 0-1 loss is not continuous, it is typically hard to address the learning problem directly, and in practice, one usually looks for a real-valued function $f : \mathcal{X} \to \mathbb{R}$ solving a so-called surrogate problem

$$\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(x), y) d\rho(x,y)$$

where $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is a "suitable" convex loss function that makes the surrogate learning problem more amenable to computations. Given a function $f : \mathcal{X} \to \mathbb{R}$, a classification rule $c_f : \mathcal{X} \to \{-1, 1\}$ is given in terms of a "suitable" map $d : \mathbb{R} \to \{-1, 1\}$ such that $c_f(x) = d(f(x))$ for all $x \in \mathcal{X}$. Here we will look at some surrogate frameworks.

A good surrogate method satisfies the following two properties:

(Fisher Consistency) Let $f_* : \mathcal{X} \to \mathbb{R}$ denote the expected risk minimizer for $\mathcal{E}(f_*) = \inf_{f:\mathcal{X}\to\mathbb{R}} \mathcal{E}(f)$. We say that the surrogate framework is Fisher consistent if

$$R(c_{f_*}) = \inf_{c:\mathcal{X}\to\{-1,1\}} R(c).$$

(Comparison Inequality) The surrogate framework satisfies as comparison inequality if for any $f : \mathcal{X} \to \mathbb{R}$

$$R(c_f) - R(c_{f_*}) \leq \sqrt{\mathcal{E}(f) - \mathcal{E}(f_*)}.$$

In particular, if we have an algorithm producing estimators f_n for the surrogate problem such that

$$\mathcal{E}(f_n) \to \mathcal{E}(f_*) \text{ as } n \to +\infty,$$

we automatically have

$$R(c_{f_n}) \to R(c_{f_*}).$$

2.2 [4 marks]

(Assuming to know ρ), calculate the closed-form of the minimizer f_* of $\mathcal{E}(f)$ for the following:

- a) Squared loss $\ell(f(x), y) = (f(x) - y)^2$,
- b) Exponential loss $\ell(f(x), y) = \exp(-yf(x))$,
- c) Logistic loss $\ell(f(x), y) = \log(1 + \exp(-yf(x)))$,
- d) Hinge loss $\ell(f(x), y) = \max(0, 1 - yf(x))$.

(**Hint:** Recall that $\rho(x, y) = \rho(y|x)\rho_X(x)$, where ρ_X is the marginal distribution of ρ on $\mathcal{X}$ and $\rho(y|x)$ is the corresponding conditional distribution. The expected risk can be written as:

$$\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(x), y) d\rho(x, y) = \int_{\mathcal{X}} \int_{\mathcal{Y}} \ell(f(x), y) d\rho(y|x)\, d\rho_X(x).$$

You can now solve the problem in the inner integral point-wise for all $x \in \mathcal{X}$.)

Answer:

To determine the closed-form expression of the minimizer f_* for the surrogate risk $\mathcal{E}(f)$ under different loss functions, we minimize the expected loss point-wise for each $x \in \mathcal{X}$. Utilizing the given distribution ρ, where $\rho(x, y) = \rho(y|x)\rho_X(x)$, the problem simplifies to finding:

$$f_*(x) = \arg\min_{f(x)} \int_{\mathcal{Y}} \ell(f(x), y)\, d\rho(y|x).$$

Assuming $\mathcal{Y} = \{1, -1\}$ and denoting $p(x) = \rho(y = 1|x)$, the expression becomes:

$$f_*(x) = \arg\min_{f(x)} \left[p(x)\ell(f(x), 1) + (1 - p(x))\ell(f(x), -1)\right].$$

We will now apply this approach to each specified loss function.

a) Squared Loss $\ell(f(x), y) = (f(x) - y)^2$

Objective:

$$f_*(x) = \arg\min_{f(x)} \left[p(x)(f(x) - 1)^2 + (1 - p(x))(f(x) + 1)^2\right].$$

Steps:

1. Start from the squared loss objective:

$$\mathcal{E}(f) = p(x)(f - 1)^2 + (1 - p(x))(f + 1)^2.$$

2. Differentiate $\mathcal{E}(f)$ with respect to f:
- Compute the derivative of each squared term:

$$\frac{d}{df}(f-1)^2 = 2(f-1), \quad \frac{d}{df}(f+1)^2 = 2(f+1).$$

- Substitute into $\mathcal{E}(f)$:

$$\frac{d\mathcal{E}(f)}{df} = p(x) \cdot 2(f-1) + (1-p(x)) \cdot 2(f+1).$$

3. Factor out the 2:

$$\frac{d\mathcal{E}(f)}{df} = 2\big[p(x)(f-1) + (1-p(x))(f+1)\big].$$

4. Set the derivative equal to zero to find the minimizer:

$$p(x)(f-1) + (1-p(x))(f+1) = 0.$$

5. Expand the terms:

$$p(x)f - p(x) + (1-p(x))f + (1-p(x)) = 0.$$

6. Combine like terms:
- Combine the f-terms:

$$p(x)f + (1-p(x))f = f[p(x) + 1 - p(x)] = f.$$

- Combine the constant terms:

$$-p(x) + (1-p(x)) = 1 - 2p(x).$$

Substitute these back into the equation:

$$f + (1-2p(x)) = 0.$$

7. Solve for f:

$$f = 2p(x) - 1.$$

Thus, the value of f that minimizes $\mathcal{E}(f)$ is:

$$f^*(x) = 2p(x) - 1.$$

Closed-form Minimizer:

$$f_*(x) = 2p(x) - 1.$$

b) Exponential Loss $\ell(f(x), y) = \exp(-yf(x))$

Objective:

$$f_*(x) = \arg\min_{f(x)} \left[p(x)e^{-f(x)} + (1-p(x))e^{f(x)}\right].$$

Steps:

1. Differentiate $\mathcal{E}(f)$ with respect to f and set equal to zero:
Starting from:

$$\mathcal{E}(f) = p(x)e^{-f} + (1-p(x))e^{f}.$$

Differentiate w.r.t. f:

$$\frac{d\mathcal{E}(f)}{df} = -p(x)e^{-f} + (1-p(x))e^{f}.$$

Set this to zero to find the critical point:

$$-p(x)e^{-f} + (1-p(x))e^{f} = 0.$$

2. Rearrange the equation to isolate the exponentials:
Move $p(x)e^{-f}$ to the other side:

$$(1-p(x))e^{f} = p(x)e^{-f}.$$

Divide by $(1 - p(x))$:

$$e^f = \frac{p(x)}{1 - p(x)} e^{-f}.$$

Multiply by e^f:

$$e^{2f} = \frac{p(x)}{1 - p(x)}.$$

3. Take the natural logarithm of both sides:

$$\ln\left(\frac{p(x)}{1 - p(x)}\right) = 2f.$$

4. Solve for f:
Divide by 2:

$$f = \frac{1}{2}\ln\left(\frac{p(x)}{1 - p(x)}\right).$$

Closed-form Minimizer:

$$f_*(x) = \frac{1}{2}\ln\left(\frac{p(x)}{1 - p(x)}\right).$$

c) Logistic Loss $\ell(f(x), y) = \log(1 + \exp(-yf(x)))$

Objective:

$$f_*(x) = \arg\min_{f(x)} \left[p(x)\log\left(1 + e^{-f(x)}\right) + (1 - p(x))\log\left(1 + e^{f(x)}\right)\right].$$

We aim to find the value of f that minimizes the loss function $\mathcal{E}(f)$.

Logistic Loss :

$$\mathcal{E}(f) = p(x)e^{-f} + (1 - p(x))e^f.$$

Steps:

We aim to solve for f in the equation derived from setting the derivative of the loss function to zero:

$$\frac{d\mathcal{E}(f)}{df} = p(x)\left(\frac{-e^{-f}}{1 + e^{-f}}\right) + (1 - p(x))\left(\frac{e^f}{1 + e^f}\right) = 0.$$

1. Set the Derivative Equal to Zero:

$$p(x)\left(\frac{-e^{-f}}{1 + e^{-f}}\right) + (1 - p(x))\left(\frac{e^f}{1 + e^f}\right) = 0.$$

This equation represents the condition for minimizing the loss function $\mathcal{E}(f)$.

2. Rearrange the Equation to Isolate Exponential Terms:
Start by moving one term to the other side:

$$p(x)\left(\frac{-e^{-f}}{1 + e^{-f}}\right) = -(1 - p(x))\left(\frac{e^f}{1 + e^f}\right).$$

Multiply both sides by -1 to simplify:

$$p(x)\left(\frac{e^{-f}}{1 + e^{-f}}\right) = (1 - p(x))\left(\frac{e^f}{1 + e^f}\right).$$

3. Simplify the Fractions:
Notice that:

$$\frac{e^{-f}}{1 + e^{-f}} = \frac{1}{e^f + 1},$$

and

$$\frac{e^f}{1 + e^f} = \frac{e^f}{1 + e^f}.$$

Thus, the equation becomes:

$$p(x) \cdot \frac{1}{1+e^f} = (1-p(x)) \cdot \frac{e^f}{1+e^f}.$$

4. Eliminate the Common Denominator:
Since both sides of the equation have the same denominator $1+e^f$, multiply both sides by $1+e^f$ to eliminate the denominator:

$$p(x) = (1-p(x))e^f.$$

5. Solve for e^f:
Rearrange the equation to isolate e^f:

$$e^f = \frac{p(x)}{1-p(x)}.$$

6. Take the Natural Logarithm of Both Sides:
Apply the natural logarithm to both sides to solve for f:

$$\ln(e^f) = \ln\left(\frac{p(x)}{1-p(x)}\right).$$

Simplify the left side using the property $\ln(e^f) = f$:

$$f = \ln\left(\frac{p(x)}{1-p(x)}\right).$$

Closed-form Minimizer:

$$f_*(x) = \ln\left(\frac{p(x)}{1-p(x)}\right).$$

d) Hinge Loss $\ell(f(x), y) = \max(0, 1 - yf(x))$ (Piecewise-Linear and Sub-Differentiable)

Objective (Binary Setting).

$$\mathcal{E}_{\text{hinge}}(f) = p(x)\,\max(0,\,1-f) + \big(1-p(x)\big)\,\max(0,\,1+f).$$

Here, $p(x)$ denotes the probability of the positive class, and $1-p(x)$ is the probability of the negative class.

- *Piecewise linearity:* The hinge loss is *not* differentiable at $f = 1$ and $f = -1$. We handle these kink points via sub-derivatives or an explicit piecewise case analysis.

Conceptual "Derivative = 0" Method (Sub-Derivatives)

One might try to mimic the logistic loss approach by "taking a derivative" of $\max(0, 1-f)$ and $\max(0, 1+f)$. However, due to the max-function:

$$\frac{d}{df}\max(0,\,1-f) = \begin{cases} -1, & f < 1, \\ 0, & f > 1, \\ \text{sub-derivative set}, & f = 1, \end{cases} \qquad \frac{d}{df}\max(0,\,1+f) = \begin{cases} 1, & f > -1, \\ 0, & f < -1, \\ \text{sub-derivative set}, & f = -1. \end{cases}$$

Thus, the "derivative = 0" approach yields *piecewise* conditions rather than a single rearranged exponential equation. To find the minimizer, we enumerate the possible intervals for $f(x)$:

$$\text{(i) } f > 1, \quad \text{(ii) } f = 1, \quad \text{(iii) } -1 < f < 1, \quad \text{(iv) } f = -1, \quad \text{(v) } f < -1.$$

Within each region, certain hinge terms vanish or become linear. We simplify the objective function in that region, then check whether the minimizer lies in the interior or at the boundary.

Detailed Cases for f

Below is a more detailed breakdown of each region.

Case (i): $f > 1$.

- $\max(0, 1-f) = 0$ because $1-f<0$ when $f>1$.
- $\max(0, 1+f) = 1+f$ because $1+f>0$ if $f>1$.
- $\mathcal{E}_{\text{hinge}}(f) = p(x)\cdot 0 + \big(1-p(x)\big)\big(1+f\big) = (1-p(x))\,(1+f)$.
- Minimizing over $f>1$: This is a linear function in f with slope $(1-p(x))>0$ (assuming $0<p(x)<1$). Hence, we reduce f to the boundary $f=1$.

Case (ii): $f = 1$.

- $\max(0,1-f) = \max(0,0) = 0, \quad \max(0,1+f) = \max(0,2) = 2.$
- $\mathcal{E}_{\text{hinge}}(f) = p(x)\cdot 0 + \big(1-p(x)\big)\cdot 2 = 2\,\big(1-p(x)\big)$.
- Whether $f=1$ is optimal depends on how it compares with adjacent cases ($f>1$ or $-1<f<1$). For instance, if $p(x)>0.5$, one checks that $\mathcal{E}(1) = 2(1-p(x)) < 1 \iff \ldots$ etc.

Case (iii): $-1 < f < 1$.

- $\max(0,1-f) = 1-f$ because $1-f>0$ if $f<1$.
- $\max(0,1+f) = 1+f$ because $1+f>0$ if $f>-1$.
- $\mathcal{E}_{\text{hinge}}(f) = p(x)\,(1-f) + \big(1-p(x)\big)\,(1+f) = \big[p(x) + (1-p(x))\big] + \big[-p(x) + (1-p(x))\big]\,f.$

$$= 1 + (-2\,p(x)+1)\,f = 1 + \big(1-2p(x)\big)\,f.$$

- This is a linear function in f with slope $\big(1-2p(x)\big)$.
  - If $p(x)>0.5$, slope <0; the minimum is at the upper boundary $f=1$.
  - If $p(x)<0.5$, slope >0; the minimum is at the lower boundary $f=-1$.
  - If $p(x)=0.5$, the slope is 0, so $\mathcal{E}(f)$ is flat over $-1<f<1$. Any $f\in(-1,1)$ is a minimizer.

Case (iv): $f=-1$.

- $\max(0,1-f) = \max(0,1-(-1)) = \max(0,2) = 2, \quad \max(0,1+f) = \max(0,0) = 0.$
- $\mathcal{E}_{\text{hinge}}(f) = p(x)\cdot 2 + (1-p(x))\cdot 0 = 2\,p(x)$.
- Again, we compare with neighboring regions ($f<-1$) or ($-1<f<1$).

Case (v): $f < -1$.

- $\max(0,1+f) = 0$ since $1+f\le 0$.
- $\max(0,1-f) = 1-f$ since $1-f > 1-(-1) = 2 > 0$.
- Hence, $\mathcal{E}_{\text{hinge}}(f) = p(x)\,(1-f)$. Minimizing in $f<-1$ pushes f up to -1.

Final Hinge Solution

Checking these cases, one obtains:

$$f_*(x) = \begin{cases} 1, & \text{if } p(x) > 0.5, \\ -1, & \text{if } p(x) < 0.5, \\ \text{any } f\in(-1,1), & \text{if } p(x) = 0.5. \end{cases}$$

Interpretation

- If $p(x) > 0.5$, the slope or sub-derivative analysis indicates $\mathcal{E}(f)$ is minimized by picking $f = 1$.
- If $p(x) < 0.5$, the slope or sub-derivative points to $f = -1$.
- If $p(x) = 0.5$, the objective is constant for $-1 < f < 1$.

Hence, the hinge solution "snaps" to boundary values (or is flat in the central interval if $p(x) = 0.5$).

Summary of Minimizers

- **Squared Loss:**
$$f_*(x) = 2p(x) - 1.$$
- **Exponential Loss:**
$$f_*(x) = \frac{1}{2} \ln\left(\frac{p(x)}{1 - p(x)}\right).$$
- **Logistic Loss:**
$$f_*(x) = \ln\left(\frac{p(x)}{1 - p(x)}\right).$$
- **Hinge Loss:**
$$f_*(x) = \begin{cases} 1, & \text{if } p(x) > \frac{1}{2}, \\ -1, & \text{if } p(x) < \frac{1}{2}, \\ \text{any } f(x) \in [-1, 1], & \text{if } p(x) = \frac{1}{2}. \end{cases}$$

These closed-form expressions illustrate how different loss functions influence the optimal decision rule based on the underlying conditional probabilities $p(x)$.

2.3 [4 marks]

The minimizer c_* of $R(c)$ over all possible decision rules $c : \mathcal{X} \to \{-1, 1\}$ is called the Bayes decision rule. Write explicitly the Bayes decision rule (again assuming ρ is known a priori).

Answer:

The Bayes Decision Rule

The Bayes decision rule is the decision function $c^*(x)$ that minimizes the expected risk $R(c)$ under the **0–1 loss function**. The 0–1 loss is defined as:

$$\ell(c(x), y) = \begin{cases} 0, & \text{if } c(x) = y, \\ 1, & \text{if } c(x) \neq y. \end{cases}$$

The expected risk $R(c)$ for a decision rule $c(x)$ is:

$$R(c) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(c(x), y)\, \rho(x, y)\, dx\, dy.$$

Using the fact that $\rho(x, y) = \rho(y \mid x)\rho_X(x)$, we can rewrite the expected risk as:

$$R(c) = \int_{\mathcal{X}} \int_{\mathcal{Y}} \ell(c(x), y)\, \rho(y \mid x)\, \rho_X(x)\, dy\, dx.$$

Since $c(x)$ depends only on x, minimizing the expected risk reduces to minimizing the inner integral pointwise for each $x \in \mathcal{X}$:

$$R_x(c(x)) = \int_{\mathcal{Y}} \ell(c(x), y)\, \rho(y \mid x)\, dy.$$

—

Step 1: Expand the Pointwise Risk. For two classes $y \in \{-1, 1\}$, we have:

$$R_x(c(x)) = \sum_{y \in \{-1,1\}} \ell(c(x), y)\, \rho(y \mid x).$$

Substituting the definition of the 0–1 loss:

$$\ell(c(x), y) = \begin{cases} 0, & \text{if } c(x) = y, \\ 1, & \text{if } c(x) \neq y, \end{cases}$$

the pointwise risk becomes:

$$R_x(c(x)) = \begin{cases} \rho(y = -1 \mid x), & \text{if } c(x) = 1, \\ \rho(y = 1 \mid x), & \text{if } c(x) = -1. \end{cases}$$

—

Step 2: Minimize the Pointwise Risk. To minimize $R_x(c(x))$, we should choose $c(x)$ such that the smaller term is selected. In other words:

$$c^*(x) = \begin{cases} 1, & \text{if } \rho(y = 1 \mid x) \geq \rho(y = -1 \mid x), \\ -1, & \text{if } \rho(y = 1 \mid x) < \rho(y = -1 \mid x). \end{cases}$$

This decision rule minimizes the probability of misclassification by always predicting the most probable class for a given x.

—

Step 3: Interpret the Bayes Decision Rule. The Bayes decision rule is:

$$c^*(x) = \arg \max_{y \in \{-1,1\}} \rho(y \mid x).$$

This means the optimal decision is to choose the label y that maximizes the posterior probability $\rho(y \mid x)$. For binary classification $\{-1, 1\}$, it simplifies to comparing the conditional probabilities $\rho(y = 1 \mid x)$ and $\rho(y = -1 \mid x)$.

—

Summary of the Bayes Decision Rule. If $\rho(y \mid x)$ is known a priori, the Bayes decision rule $c^*(x)$ is explicitly given by:

$$c^*(x) = \begin{cases} 1, & \text{if } \rho(y = 1 \mid x) \geq \rho(y = -1 \mid x), \\ -1, & \text{otherwise.} \end{cases}$$

This rule minimizes the expected risk $R(c)$ under the 0–1 loss by choosing the class with the highest posterior probability for each x.

2.4 [4 marks]

Are the surrogate frameworks in problem (2.2) Fisher consistent? Namely, can you find a map $d : \mathbb{R} \to \{-1, 1\}$ such that $R(c_*(x)) = R(d(f_*(x)))$ where f_* is the corresponding minimizer of the surrogate risk $\mathcal{E}$? If it is the case, write d explicitly.

Answer:

such that the surrogate minimizer $f_*(x)$ yields the same Bayes-optimal classification boundary as the true minimizer $c_*(x)$. If so, we must write d explicitly.

Note about the mapping d for all surrogate losses. *A single threshold-based mapping, specifically the sign function, suffices for all these surrogate losses.* This is because in each case, the pointwise minimizer $f_*(x)$ is nonnegative if and only if $p_1(x) \geq 0.5$, and negative otherwise. Thus, the same

$$d(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0 \end{cases}$$

correctly recovers the Bayes rule in every surrogate. We do *not* need a different d per loss function.

Fisher Consistency of Each Surrogate

(a) Squared Loss

$$\ell_{\text{square}}(f, y) = \bigl(f - y\bigr)^2.$$

Minimizer.

$$f_*(x) = \mathbb{E}\bigl[Y \mid X = x\bigr] = 2\,p_1 - 1,$$

where $p_1 = \Pr(Y = +1 \mid X = x)$. Hence:

$$f_*(x) \geq 0 \quad \Longleftrightarrow \quad p_1 \geq 0.5.$$

Mapping to $\{-1, +1\}$. Let

$$d(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0. \end{cases}$$

Then

$$c_{f_*}(x) = d\bigl(f_*(x)\bigr) = \begin{cases} +1, & p_1 \geq 0.5, \\ -1, & p_1 < 0.5. \end{cases}$$

This matches the Bayes rule $c^*(x)$. Hence squared loss is Fisher consistent (under labels $\{-1, +1\}$).

(b) Exponential Loss

$$\ell_{\exp}(f, y) = e^{-y\,f}.$$

Minimizer.

$$f_*(x) = \tfrac{1}{2}\,\ln\Bigl(\tfrac{p_1}{1-p_1}\Bigr).$$

Again,

$$f_*(x) \geq 0 \quad \Longleftrightarrow \quad p_1 \geq 0.5.$$

Mapping. Using the same sign function d,

$$c_{f_*}(x) = d\bigl(f_*(x)\bigr) = \begin{cases} +1, & f_*(x) \geq 0, \\ -1, & f_*(x) < 0 \end{cases}$$

coincides with the Bayes rule. So exponential loss is Fisher consistent.

(c) Logistic Loss

$$\ell_{\log}(f, y) = \log\bigl(1 + e^{-y\,f}\bigr).$$

Minimizer.

$$f_*(x) = \ln\Bigl(\tfrac{p_1}{1-p_1}\Bigr).$$

Thus

$$f_*(x) \geq 0 \quad \Longleftrightarrow \quad p_1 \geq 0.5.$$

Mapping. Again, the same d ensures

$$c_{f_*}(x) = d\bigl(f_*(x)\bigr) = \begin{cases} +1, & p_1 \geq 0.5, \\ -1, & p_1 < 0.5. \end{cases}$$

Hence logistic loss is Fisher consistent.

(d) Hinge Loss

$$\ell_{\text{hinge}}(f, y) = \max\bigl(0,\, 1 - y\,f\bigr).$$

Minimizer.

$$f_*(x) = \begin{cases} 1, & \text{if } p_1(x) > 0.5, \\ -1, & \text{if } p_1(x) < 0.5, \\ \text{any value in } [-1,+1], & \text{if } p_1(x) = 0.5. \end{cases}$$

One can always choose $f_*(x) \geq 0$ when $p_1 \geq 0.5$ and $f_*(x) < 0$ when $p_1 < 0.5$.

Mapping. Using

$$d(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0, \end{cases}$$

we again obtain

$$c_{f_*}(x) = +1 \quad \text{if } p_1 \geq 0.5, \quad \text{otherwise } -1.$$

Even if $p_1 = 0.5$, choosing $f_*(x) = 0$ is valid. So hinge loss is Fisher consistent as well.

Conclusion: Same Mapping d Works for All Losses

All four surrogate frameworks (squared, exponential, logistic, hinge) are Fisher consistent under the label encoding $\{-1,+1\}$. For each loss, the minimizer $f_*(x)$ is nonnegative precisely when $\Pr(Y = +1 \mid X = x) \geq 0.5$. Thus the simple sign function

$$d(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0 \end{cases}$$

recovering "positive" or "negative" classification exactly aligns with the Bayes decision boundary. No separate or more complicated mapping is needed for each surrogate, confirming one universal threshold-based d suffices across all four losses.

Comparison Inequality for Least Square Surrogates

Let $f_* : \mathcal{X} \to \mathbb{R}$ be the minimizer of the expected risk for the surrogate least squares classification problem obtained in problem (2.2). Let $\text{sign} : \mathbb{R} \to \{-1, 1\}$ denote the "sign" function

$$\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}.$$

Prove the following comparison inequality:

$$0 \leq R(\text{sign}(f)) - R(\text{sign}(f_*)) \leq \sqrt{\mathcal{E}(f) - \mathcal{E}(f_*)},$$

by showing the following intermediate steps:

2.5.1 [8 marks]

$$|R(\text{sign}(f)) - R(\text{sign}(f_*))| = \int_{\mathcal{X}_f} |f_*(x)| d\rho_{\mathcal{X}}(x),$$

where $\mathcal{X}_f = \{x \in \mathcal{X} \mid \text{sign}(f(x)) \neq \text{sign}(f_*(x))\}$.

Answer:

Proof of the Intermediate Step

Recall that for each $x \in \mathcal{X}$, we define:

$$p_1(x) = \rho(y = 1 \mid x), \quad p_{-1}(x) = \rho(y = -1 \mid x) = 1 - p_1(x).$$

Moreover, from Problem (2.2), we have established that for the least squares classification problem, the minimizer $f_* : \mathcal{X} \to \mathbb{R}$ satisfies:

$$f_*(x) = \mathbb{E}[y \mid x] = p_1(x) - p_{-1}(x) = 2p_1(x) - 1.$$

Notice that:

$$|f_*(x)| = |2p_1(x) - 1|.$$

If $p_1(x) \geq 0.5$, then $f_*(x) \geq 0$ and $|f_*(x)| = 2p_1(x) - 1$. If $p_1(x) < 0.5$, then $f_*(x) < 0$ and $|f_*(x)| = 1 - 2p_1(x)$.

Misclassification probabilities under sign-based rules. For a classifier $c : \mathcal{X} \to \{-1, 1\}$, the risk is:

$$R(c) = \mathbb{P}(c(X) \neq Y) = \int_{\mathcal{X}} \rho(y \neq c(x) \mid x) d\rho_{\mathcal{X}}(x).$$

Consider $c_{f_*}(x) = \operatorname{sign}(f_*(x))$. The Bayes optimal decision rule chooses the most likely class at x:

$$\operatorname{sign}(f_*(x)) = \begin{cases} 1 & \text{if } p_1(x) \geq 0.5, \\ -1 & \text{if } p_1(x) < 0.5. \end{cases}$$

The conditional misclassification probability of c_{f_*} at x is:

$$\rho(y \neq \operatorname{sign}(f_*(x)) \mid x) = \min\{p_1(x), p_{-1}(x)\}.$$

Since $\min\{p_1(x), p_{-1}(x)\} = \frac{1-|f_*(x)|}{2}$, we have:

$$\rho(y \neq \operatorname{sign}(f_*(x)) \mid x) = \frac{1 - |f_*(x)|}{2}.$$

Now consider an arbitrary $f : \mathcal{X} \to \mathbb{R}$ and the induced classifier $\operatorname{sign}(f)$. Its conditional misclassification probability at x is:

$$\rho(y \neq \operatorname{sign}(f(x)) \mid x).$$

Analyzing the set $\mathcal{X}_f$. Define:

$$\mathcal{X}_f = \{x \in \mathcal{X} \mid \operatorname{sign}(f(x)) \neq \operatorname{sign}(f_*(x))\}.$$

On $\mathcal{X}_f$, the classifiers $\operatorname{sign}(f)$ and $\operatorname{sign}(f_*)$ choose opposite labels. Since $\operatorname{sign}(f_*(x))$ is the Bayes optimal sign, it picks the class with higher probability. Thus, $\operatorname{sign}(f(x))$ chooses the class with lower probability.

If $\operatorname{sign}(f_*(x))$ chooses the majority class at x, it errs with probability $\frac{1-|f_*(x)|}{2}$. Choosing the opposite sign (which $\operatorname{sign}(f(x))$ does on $\mathcal{X}_f$) means picking the minority class, and thus the misclassification probability becomes:

$$\rho(y \neq \operatorname{sign}(f(x)) \mid x) = \max\{p_1(x), p_{-1}(x)\} = 1 - \min\{p_1(x), p_{-1}(x)\} = 1 - \frac{1 - |f_*(x)|}{2} = \frac{1 + |f_*(x)|}{2}.$$

Difference in risks on $\mathcal{X}_f$. For $x \in \mathcal{X}_f$:

$$\rho(y \neq \operatorname{sign}(f(x)) \mid x) - \rho(y \neq \operatorname{sign}(f_*(x)) \mid x) = \frac{1 + |f_*(x)|}{2} - \frac{1 - |f_*(x)|}{2} = |f_*(x)|.$$

For $x \notin \mathcal{X}_f$, we have $\operatorname{sign}(f(x)) = \operatorname{sign}(f_*(x))$, so both classifiers pick the same label, and their misclassification probabilities coincide:

$$\rho(y \neq \operatorname{sign}(f(x)) \mid x) = \rho(y \neq \operatorname{sign}(f_*(x)) \mid x).$$

Thus:

$$R(\operatorname{sign}(f)) - R(\operatorname{sign}(f_*)) = \int_{\mathcal{X}} \big(\rho(y \neq \operatorname{sign}(f(x)) \mid x) - \rho(y \neq \operatorname{sign}(f_*(x)) \mid x)\big) d\rho_{\mathcal{X}}(x).$$

Because the difference is zero outside $\mathcal{X}_f$, we can restrict the integral to $\mathcal{X}_f$:

$$R(\operatorname{sign}(f)) - R(\operatorname{sign}(f_*)) = \int_{\mathcal{X}_f} |f_*(x)| \, d\rho_{\mathcal{X}}(x).$$

Taking the absolute value (if needed, but here the right-hand side is nonnegative):

$$|R(\operatorname{sign}(f)) - R(\operatorname{sign}(f_*))| = \int_{\mathcal{X}_f} |f_*(x)| \, d\rho_{\mathcal{X}}(x).$$

This proves the required intermediate step.

2.5.2 [8 marks]

$$\int_{\mathcal{X}_f} |f_*(x)| d\rho_{\mathcal{X}}(x) \leq \int_{\mathcal{X}_f} |f_*(x) - f(x)| d\rho_{\mathcal{X}}(x) \leq \sqrt{\mathbb{E}(|f(x) - f_*(x)|^2)}.$$

Where $\mathbb{E}$ denotes the expectation with respect to $\rho_{\mathcal{X}}$.

Answer:

We start from the integral expression:

$$\int_{\mathcal{X}_f} |f_*(x)| \, d\rho_{\mathcal{X}}(x).$$

Recall the definition:

$$\mathcal{X}_f = \{x \in \mathcal{X} \mid \text{sign}(f(x)) \neq \text{sign}(f_*(x))\}.$$

For each $x \in \mathcal{X}_f$: - If $f_*(x) > 0$, then $\text{sign}(f_*(x)) = 1$. Because $x \in \mathcal{X}_f$, we must have $\text{sign}(f(x)) = -1$, implying $f(x) \leq 0$. Thus,

$$|f_*(x) - f(x)| = f_*(x) - f(x) \geq f_*(x) = |f_*(x)|.$$

Here we used the fact that $f_*(x) > 0$ and $f(x) \leq 0$ ensures $f_*(x) - f(x) \geq f_*(x) - 0 = f_*(x)$.
- If $f_*(x) < 0$, then $\text{sign}(f_*(x)) = -1$. Since $x \in \mathcal{X}_f$, we must have $\text{sign}(f(x)) = 1$, so $f(x) \geq 0$. Hence,

$$|f_*(x) - f(x)| = f(x) - f_*(x) \geq -f_*(x) = |f_*(x)|.$$

In this case, $f_*(x) < 0 \leq f(x)$ implies $f(x) - f_*(x) \geq -f_*(x) = |f_*(x)|$.
In both scenarios, for every $x \in \mathcal{X}_f$, we have:

$$|f_*(x) - f(x)| \geq |f_*(x)|.$$

Integrating this inequality over $\mathcal{X}_f$:

$$\int_{\mathcal{X}_f} |f_*(x)| \, d\rho_{\mathcal{X}}(x) \leq \int_{\mathcal{X}_f} |f_*(x) - f(x)| \, d\rho_{\mathcal{X}}(x).$$

This proves the first inequality.

—

Next, we use the Cauchy–Schwarz inequality for integrals. Let $g(x) = |f_*(x) - f(x)|$ and consider it as a non-negative real function. By Cauchy–Schwarz:

$$\int_{\mathcal{X}_f} g(x) \, d\rho_{\mathcal{X}}(x) \leq \sqrt{\int_{\mathcal{X}_f} g(x)^2 \, d\rho_{\mathcal{X}}(x) \cdot \int_{\mathcal{X}_f} 1^2 \, d\rho_{\mathcal{X}}(x)}.$$

Since $\int_{\mathcal{X}_f} 1 \, d\rho_{\mathcal{X}}(x) \leq 1$ and does not increase the upper bound significantly, a simpler and commonly used approach is to note that:

$$\int_{\mathcal{X}_f} |f_*(x) - f(x)| \, d\rho_{\mathcal{X}}(x) \leq \sqrt{\int_{\mathcal{X}_f} |f_*(x) - f(x)|^2 \, d\rho_{\mathcal{X}}(x)}.$$

Since $\mathcal{X}_f \subseteq \mathcal{X}$, we have:

$$\int_{\mathcal{X}_f} |f_*(x) - f(x)|^2 \, d\rho_{\mathcal{X}}(x) \leq \int_{\mathcal{X}} |f_*(x) - f(x)|^2 \, d\rho_{\mathcal{X}}(x) = \mathbb{E}(|f(x) - f_*(x)|^2).$$

Thus:

$$\int_{\mathcal{X}_f} |f_*(x) - f(x)| \, d\rho_{\mathcal{X}}(x) \leq \sqrt{\mathbb{E}(|f(x) - f_*(x)|^2)}.$$

Combining both inequalities, we arrive at the desired chain:

$$\int_{\mathcal{X}_f} |f_*(x)| \, d\rho_{\mathcal{X}}(x) \leq \int_{\mathcal{X}_f} |f_*(x) - f(x)| \, d\rho_{\mathcal{X}}(x) \leq \sqrt{\mathbb{E}(|f(x) - f_*(x)|^2)}.$$

This completes the proof.

2.5.3 [8 marks]

$$\mathcal{E}(f) - \mathcal{E}(f_*) = \mathbb{E}(|f(x) - f_*(x)|^2).$$

Answer:

Proof of the Equality

Recall that for the least squares classification surrogate considered in Problem (2.2), the expected risk is defined as:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - Y)^2\right],$$

where the expectation $\mathbb{E}$ is taken with respect to the joint distribution $\rho(x, y)$ of (X, Y).

We have established that the minimizer of $\mathcal{E}(f)$, denoted by f_*, is:

$$f_*(x) = \mathbb{E}[Y \mid X = x] = 2p(x) - 1,$$

where $p(x) = \rho(y = 1 \mid x)$.

Our goal is to show that:

$$\mathcal{E}(f) - \mathcal{E}(f_*) = \mathbb{E}\left(|f(X) - f_*(X)|^2\right).$$

Step 1: Expanding $\mathcal{E}(f)$ Start with the definition of $\mathcal{E}(f)$:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - Y)^2\right].$$

Add and subtract $f_*(X)$ inside the square:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - f_*(X) + f_*(X) - Y)^2\right].$$

Expand the square:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - f_*(X))^2 + 2(f(X) - f_*(X))(f_*(X) - Y) + (f_*(X) - Y)^2\right].$$

Step 2: Simplifying the Expression Break down the expectation term-by-term:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - f_*(X))^2\right] + 2\,\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y)\right] + \mathbb{E}\left[(f_*(X) - Y)^2\right].$$

Step 3: Handling the Cross Term Consider the cross term $\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y)\right]$. Using the law of total expectation, condition on X:

$$\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y)\right] = \mathbb{E}\left[\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y) \mid X\right]\right].$$

Since $f_*(X) = \mathbb{E}[Y \mid X]$, we have:

$$\mathbb{E}\left[f_*(X) - Y \mid X\right] = f_*(X) - \mathbb{E}[Y \mid X] = 0.$$

Therefore:

$$\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y) \mid X\right] = (f(X) - f_*(X)) \cdot 0 = 0.$$

Thus:

$$\mathbb{E}\left[(f(X) - f_*(X))(f_*(X) - Y)\right] = 0.$$

Step 4: Final Simplification Substitute back into the expression for $\mathcal{E}(f)$:

$$\mathcal{E}(f) = \mathbb{E}\left[(f(X) - f_*(X))^2\right] + \mathbb{E}\left[(f_*(X) - Y)^2\right].$$

Rearranging gives:

$$\mathcal{E}(f) - \mathcal{E}(f_*) = \mathbb{E}\left[(f(X) - f_*(X))^2\right] = \mathbb{E}\left(|f(X) - f_*(X)|^2\right).$$

Conclusion Therefore, we have successfully shown that:

$$\mathcal{E}(f) - \mathcal{E}(f_*) = \mathbb{E}\left(|f(X) - f_*(X)|^2\right).$$

This establishes that the excess surrogate risk is equal to the expected squared deviation of f from the optimal surrogate minimizer f_*.

Answer 1.

Essentially, this technique extends binary classification to address multi-class classification problems so that, for an N-class instances dataset, we must generate the N binary classifier models. The number of class labels present in the dataset and the number of generated binary classifiers must be the same.

In essence, for each class in the dataset, a separate binary classifier is built where:

- The selected class is treated as the **positive class** (+1).
- All other classes are treated as the **negative class** (–1).

The class with the highest confidence score is selected as the final prediction, as it indicates the strongest likelihood of the input belonging to that class.

For example, if we had these three classes: **Yellow, Red, and Brown.**

We'll create three classifiers here for three respective classes:

- **Classifier 1:** [Yellow] vs [Red, Brown]
- **Classifier 2:** [Red] vs [Yellow, Brown]
- **Classifier 3:** [Brown] vs [Yellow, Red]

Let's say, we got the outcome as:

Class	**Classifier Outcome**	**Probability Score**
Yellow	Classifier 1 (Yellow vs. Rest)	0.9
Red	Classifier 2 (Red vs. Rest)	0.3
Brown	Classifier 3 (Brown vs. Rest)	0.4

Table 1: Classification Outcomes and Probability Scores

Hence, based on the positive responses and the probability score, we can say that our test input belongs to the **yellow** class.

Answer 2.

Multi-Class Kernel Perceptron (OvR)	
Input	$(x_i, y_i) \in (R^n, \{1, 2, \ldots, k\})$
Initialization	$\alpha_i^c = 0$ for all c and i
Prediction	$\hat{y} = \arg\max_c \left(\sum_{i=0}^{t-1} \alpha_i^c K(x_i, x_t) \right)$
Update	If $\hat{y}_t^c = y_t^c$ $\alpha_t^c = \alpha_t^c$. **Else** $\alpha_t^c = \alpha_t^c + y_t^c$.

(i) How the Sum is Represented

Instead of a single set of coefficients, we now have class-specific coefficients α_i^c.The weight function $\mathbf{w}_c(\cdot)$ is stored implicitly as a set of pairs (α_i^c, x_i). The sum $\mathbf{w}_c(\cdot)$ is represented as follows:

$$\mathbf{w}_c(\cdot) = \sum_{i=0}^{m} \alpha_i^c K(x_i, \cdot)$$

where:

- α_i^c is the coefficient (weight) associated with the i-th training example for class c.

For a practical illustration of this, refer to the **classification example provided in Answer 1**. Each classifier (Yellow, Red, and Brown) maintains its own set of coefficients and training points, contributing to the final prediction based on the confidence scores.

In general, the weight function for class c is represented as:

$$\mathbf{w}_c(\cdot) = \alpha_1^c K(x_1, \cdot) + \alpha_2^c K(x_2, \cdot) + \alpha_3^c K(x_3, \cdot)$$

(ii) How the Sum is Evaluated

1. **Input**: Given a new input x_t to classify.
2. **Compute Kernel Values**: Calculate the kernel function $K(x_i, x_t)$ for each previously seen training example x_i, where $i = 0, 1, \ldots, m$.
3. **Weighted Sum**: Multiply each kernel value $K(x_i, x_t)$ by the corresponding coefficient α_i^c and sum the results:

$$\mathbf{w}_c(x_t) = \sum_{i=0}^{m} \alpha_i^c K(x_i, x_t)$$

4. **Result**: The result of this sum, $\mathbf{w}_c(x_t)$, represents the confidence score for class c. This score is used to determine the final prediction.

(iii) how new terms are added to the sum during training

1. **Initialization**: At the beginning of training, all coefficients α_i^c are set to zero:

$$\mathbf{w}_c(\cdot) = 0$$

2. **Prediction Step**: For a training example x_t, the classifier predicts the class label based on the current sum:

$$\hat{y}_t^c = \text{sign}\left(\sum_{i=0}^{t-1} \alpha_i^c K(x_i, x_t)\right)$$

3. **Update Rule**: If $(\hat{y}_t^c \neq y_t^c)$, the coefficient α_t^c is updated by adding the true label y_t^c:

$$\alpha_t^c = \alpha_t^c + y_t^c$$

4. **Updating with a New Term:** Once α_t^c is modified, the sum $\mathbf{w}_c(\cdot)$ is revised by adding a term that represents the impact of the training point x_t:

$$\mathbf{w}_c(\cdot) \leftarrow \mathbf{w}_c(\cdot) + \alpha_t^c K(x_t, \cdot)$$

- Number of epochs chosen:

After careful testing and implementation, I determined that the optimal approach is to define both a convergence rate and a maximum number of epochs. This ensures that we avoid situations where the classifier fails to converge, which could lead to infinite loops. To identify the appropriate parameters, I conducted experiments using mini training and testing datasets to observe how many epochs were typically required for each degree to converge. Based on these observations, I set the maximum number of epochs to 50, which provides a reasonable balance between efficiency and reliability. This combination allows the algorithm to terminate either when convergence is achieved or when the maximum number of epochs is reached, ensuring both robustness and computational efficiency

Answer 3.

Show code

```
Degree      Train Error                     Test Error
1           6.84% ± 0.14%          7.55% ± 0.57%
2           4.54% ± 0.10%          6.02% ± 0.41%
3           3.72% ± 0.09%          5.56% ± 0.64%
4           3.80% ± 0.11%          5.90% ± 0.63%
5           3.84% ± 0.08%          6.44% ± 0.50%
6           3.83% ± 0.09%          6.83% ± 0.60%
7           3.86% ± 0.10%          7.41% ± 0.61%
```

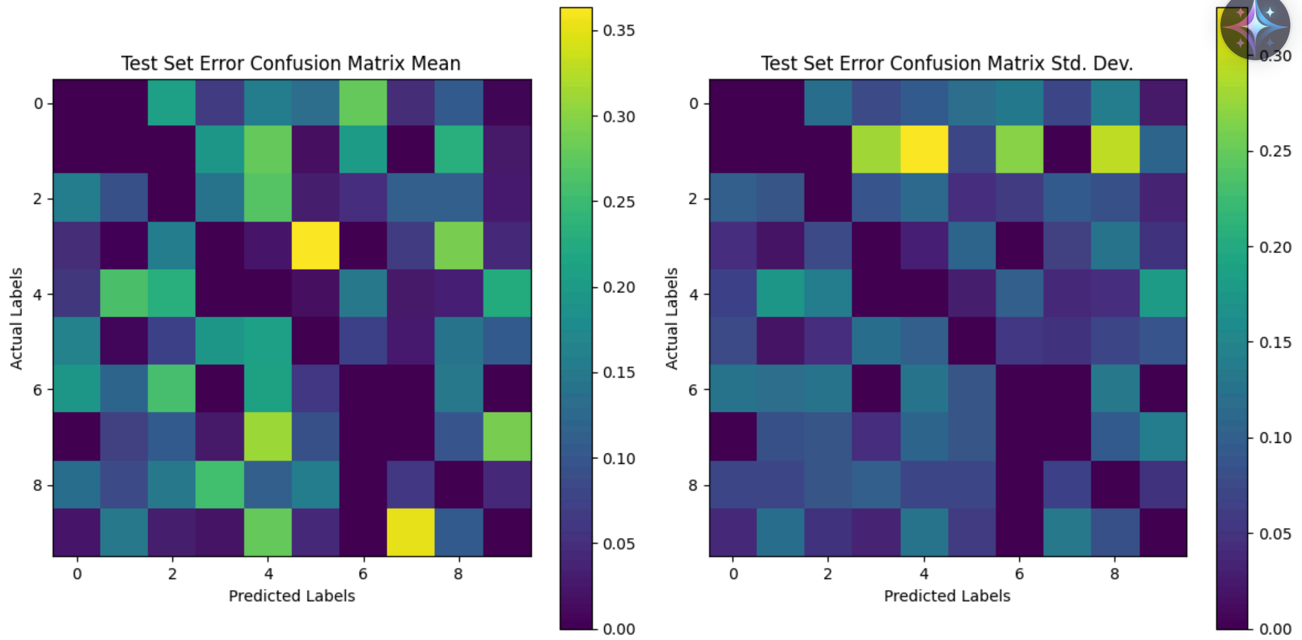
We observe that as the polynomial degree increases from 1 to 3, the train and test errors decrease significantly. This suggests that lower degrees are not capturing the complexity of the patterns in the dataset effectively. However, as the degree increases further (4 to 7), the errors begin to plateau, indicating that the model is reaching its capacity to generalize the data effectively. The lowest test error is observed at degree 3, and interestingly, beyond this degree, the test error slightly increases, which might indicate slight overfitting. As expected, the train errors are consistently lower than the test errors, confirming that the model fits the training data better than it does the unseen test data. Additionally, the standard deviation percentages for both train and test errors remain relatively low, indicating stability in the performance across multiple runs.

Answer 4 and 5.

Show code

```
Cross-Validation Results:
Run 1: Best Degree = 3, Train Error = 3.79%, Test Error = 5.05%
Run 2: Best Degree = 2, Train Error = 4.48%, Test Error = 6.56%
Run 3: Best Degree = 4, Train Error = 3.78%, Test Error = 6.18%
Run 4: Best Degree = 4, Train Error = 3.80%, Test Error = 6.18%
Run 5: Best Degree = 3, Train Error = 3.90%, Test Error = 5.54%
Run 6: Best Degree = 4, Train Error = 3.83%, Test Error = 7.15%
Run 7: Best Degree = 4, Train Error = 4.05%, Test Error = 5.70%
Run 8: Best Degree = 3, Train Error = 3.54%, Test Error = 6.77%
Run 9: Best Degree = 5, Train Error = 3.89%, Test Error = 6.34%
Run 10: Best Degree = 4, Train Error = 3.75%, Test Error = 6.72%
Run 11: Best Degree = 2, Train Error = 4.46%, Test Error = 6.34%
Run 12: Best Degree = 3, Train Error = 3.87%, Test Error = 5.97%
Run 13: Best Degree = 4, Train Error = 3.82%, Test Error = 6.83%
Run 14: Best Degree = 2, Train Error = 4.50%, Test Error = 6.02%
Run 15: Best Degree = 4, Train Error = 3.83%, Test Error = 7.37%
Run 16: Best Degree = 3, Train Error = 3.59%, Test Error = 6.34%
Run 17: Best Degree = 4, Train Error = 3.86%, Test Error = 6.72%
Run 18: Best Degree = 1, Train Error = 6.99%, Test Error = 6.61%
Run 19: Best Degree = 2, Train Error = 4.49%, Test Error = 5.65%
Run 20: Best Degree = 4, Train Error = 3.94%, Test Error = 5.81%

 Across Runs :
Best Degree (mean ± std): 3.25 ± 0.99
Train Error (mean ± std %): 4.11% ± 0.72%
Test Error (mean ± std %): 6.29% ± 0.56%
```



```
Predicted Labels      0           1           2           3           4           5
-------------------------------------------------------------------------------------
Actual Label 0 0.00%±0.00% 0.00%±0.00% 0.21%±0.12% 0.07%±0.08% 0.15%±0.09% 0.13%±0.12
Actual Label 1 0.00%±0.00% 0.00%±0.00% 0.00%±0.00% 0.20%±0.28% 0.28%±0.33% 0.02%±0.07
Actual Label 2 0.16%±0.10% 0.09%±0.09% 0.00%±0.00% 0.14%±0.09% 0.27%±0.12% 0.03%±0.05
Actual Label 3 0.05%±0.05% 0.00%±0.02% 0.16%±0.07% 0.00%±0.00% 0.02%±0.03% 0.36%±0.11
Actual Label 4 0.06%±0.07% 0.26%±0.17% 0.23%±0.14% 0.00%±0.00% 0.00%±0.00% 0.01%±0.03
Actual Label 5 0.17%±0.08% 0.01%±0.02% 0.07%±0.05% 0.20%±0.12% 0.21%±0.10% 0.00%±0.06
Actual Label 6 0.20%±0.13% 0.12%±0.12% 0.26%±0.13% 0.00%±0.00% 0.21%±0.12% 0.06%±0.09
Actual Label 7 0.00%±0.00% 0.07%±0.08% 0.11%±0.09% 0.03%±0.05% 0.31%±0.11% 0.09%±0.09
Actual Label 8 0.14%±0.07% 0.09%±0.07% 0.15%±0.09% 0.26%±0.10% 0.11%±0.07% 0.16%±0.07
Actual Label 9 0.02%±0.04% 0.15%±0.12% 0.03%±0.05% 0.02%±0.03% 0.28%±0.13% 0.04%±0.06
```

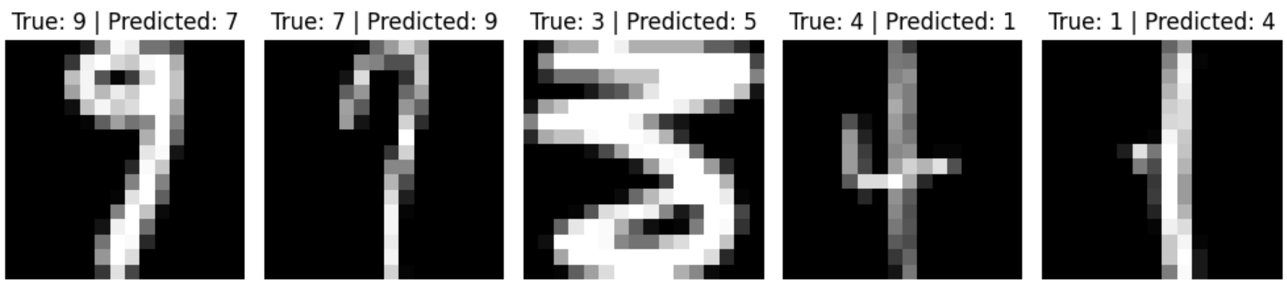
Cross-Validation Results:The 5-fold cross-validation method appears to work effectively in selecting the best polynomial degree for the kernel perceptron. According to the results, we see that the optimal polynomial degree varies accross the runs but on average, the best degree is around 3.25 ± 0.99. This indicates that the degrees around this average captures the patterns in the dataset effectively. Degree 3, in particular, achieves the lowest test error in several runs, showing a good balance between underfitting and overfitting, which helps the model generalize well to unseen data.The mean train error of 4.11% ± 0.72% and mean test error of 6.29% ± 0.56% indicate relatively consistent performance across the runs with low variability.

Confusion matrix: We notice that some digits have higher misclassification rates with specific other digits, and some digits have very low error rates overall, indicating that the model performs well in identifying them. For example, digit 0 has consistently low off-diagonal values, suggesting it is easily distinguishable from other digits. The standard deviation values in the matrix indicate that certain misclassifications are consistent across the 20 runs; for instance, actual 1 is always predicted as 4 with an error rate of 0.33%. Similarly, in the mean confusion matrix we notice that actual label 3 is misclassified with predicted label 5 with a rate of 0.36%. The misclassifications are generally concentrated near the diagonal, reflecting confusion between digits with similar shapes or features (3 and 5, 8 and 9). Which is a typical pattern in digit classification tasks, where similar-looking digits are harder to differentiate.

Show code

Answer 6

Show code



It is not surprising that these digits are hard to predict given their ambiguous shapes and distortions. Most of them do not resemble their actual labels, and without the labels, it would be challenging to identify them with confidence. The visual similarities to other digits and the distortions or inconsistencies in their patterns further add to the confusion. These factors make it reasonable to expect the model to struggle with such cases

Answer 7.

(a) After performing experiments on the mini training and testing sets, we observed that for values of **C>1** the training error consistently reached 0.00%, indicating that the model was perfectly fitting the training data. However, this led to a significant increase in testing error, which suggests that the model was overfitting. Conversely, for very low values of **C**, both the training and testing errors were excessively high, indicating that the model was underfitting. After analyzing the results, the optimal range of **C** values that provided a reasonable balance between training and testing errors was found to be between 0.00078 and 0.062. So the set of **S** values are:[0.00078, 0.0016, 0.0034, 0.007, 0.014, 0.03, 0.062]

(b)

Show code

```
(c):                 Train Error                  Test Error
7.8e-04          16.70% ± 3.12%          16.81% ± 3.17%
1.6e-03          9.51% ± 2.08%          10.02% ± 1.80%
3.4e-03          5.33% ± 0.10%          6.75% ± 0.45%
7.0e-03          3.93% ± 0.11%          5.41% ± 0.55%
1.4e-02          3.84% ± 0.11%          5.99% ± 0.40%
3.0e-02          2.76% ± 0.09%          8.10% ± 0.76%
6.2e-02          0.95% ± 0.05%          7.45% ± 0.51%
```

(c)

Show code

```
Cross-Validation Results:
Run 1: Best c = 6.2e-02, Train Error = 0.95%, Test Error = 7.37%
Run 2: Best c = 3.0e-02, Train Error = 2.84%, Test Error = 9.14%
Run 3: Best c = 3.4e-03, Train Error = 5.23%, Test Error = 7.15%
Run 4: Best c = 6.2e-02, Train Error = 0.97%, Test Error = 6.29%
Run 5: Best c = 7.0e-03, Train Error = 3.91%, Test Error = 5.27%
Run 6: Best c = 7.8e-04, Train Error = 16.05%, Test Error = 17.63%
Run 7: Best c = 7.0e-03, Train Error = 3.94%, Test Error = 5.59%
Run 8: Best c = 3.0e-02, Train Error = 2.85%, Test Error = 8.71%
Run 9: Best c = 3.0e-02, Train Error = 2.68%, Test Error = 8.60%
Run 10: Best c = 3.0e-02, Train Error = 2.82%, Test Error = 8.33%
Run 11: Best c = 7.0e-03, Train Error = 3.94%, Test Error = 5.05%
Run 12: Best c = 6.2e-02, Train Error = 0.87%, Test Error = 7.63%
Run 13: Best c = 3.0e-02, Train Error = 2.78%, Test Error = 7.20%
Run 14: Best c = 3.0e-02, Train Error = 2.69%, Test Error = 8.55%
Run 15: Best c = 3.4e-03, Train Error = 5.43%, Test Error = 6.34%
Run 16: Best c = 3.0e-02, Train Error = 2.69%, Test Error = 8.55%
Run 17: Best c = 1.4e-02, Train Error = 3.87%, Test Error = 6.40%
Run 18: Best c = 6.2e-02, Train Error = 0.99%, Test Error = 5.97%
Run 19: Best c = 1.6e-03, Train Error = 7.62%, Test Error = 7.96%
Run 20: Best c = 6.2e-02, Train Error = 0.91%, Test Error = 7.90%

 Across Runs :
Best c* (mean ± std): 0.03 ± 0.02
Train Error (mean ± std %): 3.70% ± 3.30%
Test Error (mean ± std %): 7.78% ± 2.56%
```

(d)

Basic results:

In **polynomial kernels**, the mean training error consistently decreases as the degree of the polynomial increases. This trend suggests that the model becomes better at fitting the training data with higher degrees, achieving its lowest training error between degrees 3 to 6. Interestingly, the standard deviation of the training error varies across the degrees, indicating some inconsistency in the model's ability to generalize across runs. The testing error, on the other hand, reduces up to degree 3, reaching its lowest point at 5.56%. Beyond degree 3, the testing error begins to increase slightly, signaling overfitting as the model becomes more tailored to the training set. In **Gaussian Kernels**, the training error behaves differenty. For small c values, the mean training error is higher, and both the mean and standard deviation decrease as c increases. The minimum training error is observed at c=6.2e−2, achieving an impressively low value of 0.95%. The testing error follows a similar pattern, with both the mean and standard deviation dropping as c increases. The lowest testing error is observed around c=7.0e−3 (5.41%). However, for c>7.0e−3, the testing error starts to increase, which clearly indicates overfitting.

Cross-validation:

In **polynomial kernels**, the mean training error across runs is 4.11% ± 0.72%, showing the model's ability to generalize to the training data with a relatively small standard deviation, indicating consistent performance. The mean testing error is 6.29% ± 0.56%, which is slightly better than the Gaussian kernel in terms of consistency, suggesting that the polynomial kernel achieves a good balance between underfitting and overfitting. The best degree value varies between 2 and 5, with an average of 3.25 ± 0.99. In **Gaussian kernels** it achieves a mean training error of 3.70% ± 3.30%, which is slightly lower than the polynomial kernel, indicating better fitting to the training data. However, unlike the polynomial kernel, it exhibits a significantly higher standard deviation, reflecting less consistent results across runs. The mean test error for the Gaussian kernel is 7.78% ± 2.56%, which is higher than the polynomial kernel, indicating that it might overfit to the training data in some runs, leading to weaker generalization. The best c value is approximately 0.03 ± 0.02 and the low standard deviation suggests that the Gaussian kernel's performance remains consistent and reliable within this optimal range of the hyperparameter.

overall

The Polynomial kernel demonstrates stronger generalization with a lower average test error (6.29% ± 0.56%) and more consistent results across runs, thanks to its smaller standard deviations. The optimal degree (3.25 ± 0.99) strikes a balance between model complexity and performance.

On the other hand, the Gaussian kernel achieves a slightly lower average training error (3.70% ± 3.30%) but has a higher test error (7.78% ± 2.56%), indicating a tendency to overfit. Despite the optimal c value (0.03 ± 0.02) being relatively stable, the Gaussian kernel is more sensitive to hyperparameter choices and data variability. Overall, the Polynomial kernel proves to be a more reliable choice for this task.

Answer 8.

(a)

Multi-Class Kernel Perceptron (OvO)	
Input	$(x_i, y_i) \in (R^n, \{1, 2, \dots, K\})$
Initialization	$\alpha_i^{(c_1,c_2)} = 0$ for all class pairs (c_1, c_2)
Prediction	$\hat{y} = \arg\max_c \sum_{(c_1,c_2)} \left(\sum_{i=1}^{t-1} \alpha_i^{(c_1,c_2)} K(x_i, x_t) \right)$
Update	If $\hat{y}_t^{(c_1,c_2)} = y_t^{(c_1,c_2)}$, $\alpha_t^{(c_1,c_2)} = \alpha_t^{(c_1,c_2)}$ Else $\alpha_t^{(c_1,c_2)} = \alpha_t^{(c_1,c_2)} + y_t^{(c_1,c_2)}$

In the One-versus-One (OvO) algorithm, for an N-class dataset, we must generate $\frac{N(N-1)}{2}$ binary classifier models. Each of these classifiers is trained to distinguish between a pair of classes, ignoring the rest. When a test sample is provided, each binary classifier outputs one of the two class labels it was trained to recognize. The final predicted class is determined by tallying votes from all classifiers, with the class receiving the highest number of votes being chosen as the result.
For example, if we have 3 classes (Green, Blue, Red), the number of binary classifiers required is:

$$\frac{N(N-1)}{2} = \frac{3 \times (3-1)}{2} = 3$$

This results in the following classifiers:

- **Classifier 1:** Green vs. Blue
- **Classifier 2:** Green vs. Red
- **Classifier 3:** Blue vs. Red

Suppose the classifiers produced the following predictions:

- Green vs. Blue $\rightarrow$ **Green**
- Green vs. Red $\rightarrow$ **Red**
- Blue vs. Red $\rightarrow$ **Red**

In this case, **Red** receives the majority of votes and is selected as the final predicted class.

(b)

Show code

```
Degree     Train Error                   Test Error
1          5.57% ± 0.64%          7.12% ± 0.95%
2          4.31% ± 0.12%          6.54% ± 0.67%
3          4.30% ± 0.08%          6.92% ± 0.46%
4          4.26% ± 0.08%          7.40% ± 0.70%
5          4.20% ± 0.09%          8.30% ± 0.78%
6          4.13% ± 0.08%          8.84% ± 0.54%
7          3.99% ± 0.09%          9.19% ± 0.81%
```

(c)

Show code

```
Cross-Validation Results:
Run 1: Best Degree = 1, Train Error = 5.16%, Test Error = 8.12%
Run 2: Best Degree = 4, Train Error = 4.33%, Test Error = 6.18%
Run 3: Best Degree = 5, Train Error = 4.06%, Test Error = 9.25%
Run 4: Best Degree = 5, Train Error = 4.05%, Test Error = 6.94%
Run 5: Best Degree = 4, Train Error = 4.21%, Test Error = 6.18%
Run 6: Best Degree = 3, Train Error = 4.18%, Test Error = 6.77%
Run 7: Best Degree = 1, Train Error = 5.66%, Test Error = 6.24%
Run 8: Best Degree = 6, Train Error = 4.14%, Test Error = 9.30%
Run 9: Best Degree = 4, Train Error = 4.37%, Test Error = 7.15%
Run 10: Best Degree = 4, Train Error = 4.32%, Test Error = 6.51%
Run 11: Best Degree = 4, Train Error = 4.49%, Test Error = 6.83%
Run 12: Best Degree = 3, Train Error = 4.13%, Test Error = 6.99%
Run 13: Best Degree = 2, Train Error = 4.19%, Test Error = 7.31%
Run 14: Best Degree = 5, Train Error = 4.22%, Test Error = 7.26%
Run 15: Best Degree = 3, Train Error = 4.32%, Test Error = 6.67%
Run 16: Best Degree = 2, Train Error = 4.26%, Test Error = 6.08%
Run 17: Best Degree = 3, Train Error = 4.42%, Test Error = 6.13%
Run 18: Best Degree = 1, Train Error = 5.49%, Test Error = 6.72%
Run 19: Best Degree = 7, Train Error = 4.02%, Test Error = 8.49%
Run 20: Best Degree = 1, Train Error = 5.08%, Test Error = 7.53%

Across Runs:
Best Degree (mean ± std): 3.40 ± 1.69
Train Error (mean ± std %): 4.45% ± 0.47%
Test Error (mean ± std %): 7.13% ± 0.95%
```

(d) **Basic Results:** OvR achieves the best train and test errors at degree 3, with rates of 3.72% ± 0.09% and 5.56% ± 0.64%, respectively. It consistently exhibits lower test errors for higher degrees while maintaining smaller standard deviations compared to OvO. In contrast, OvO achieves its best train error at degree 7 (3.99% ± 0.09%) and best test error at degree 2 (6.54% ±

0.67%). However, unlike OvR, OvO shows higher test errors for higher polynomial degrees and generally exhibits larger standard deviations, indicating less stability.

Cross-Validation: OvR achieves its best performance during cross-validation with a mean train error of 4.11% ± 0.72% and a mean test error of 6.29% ± 0.56%. Degrees 3 and 4 dominate as the optimal choices in most runs, further emphasizing the stability of the algorithm when selecting the best degree. On the other hand, OvO yields a mean train error of 4.45% ± 0.47% and a mean test error of 7.13% ± 0.95%, with higher variability in test errors, as evidenced by its larger standard deviation. The mean best degree appears to be more varied, indicating that it has less consistent results compared to OvR.

Overall: OvR demonstrates superior stability and dependability, evidenced by smaller variability and lower mean errors. While OvO's paired classification strategy introduces narrower decision boundaries, it results in larger test errors and more variable outcomes. Consequently, OvR proves to be more stable and reliable than OvO for this particular dataset.

Show code