IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: Oct 24, 2025
Assignment 3 — Due Date : Nov 08, 2025, **9pm**

Roll No:EE25S009
Collaborators (if any): — Name: RITABRATA MANDAL
References/sources (if any):

General Instructions

- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), and submit the resulting **single pdf file** at Gradescope by the due date. (Note: **No late submissions** will be allowed, other than one-day late submission with 10% penalty or four-day late submission with 30% penalty! You can join Gradescope using the course entry code 6K4P43. **Within Gradescope, clearly mark your answer to each question**, else we won't be able to grade it.)
- For the programming question, please submit your code (rollno.ipynb file and rollno.py file in rollno.zip) directly in moodle, but provide your results/answers (including Jupyter notebook **with output**) in the pdf file you upload to Gradescope.
- Collaboration is encouraged, but all write-ups must be done individually and independently, and mention your collaborator(s) if any. Same rules apply for codes written for any programming assignments (i.e., write your own code; we will run plagiarism and AI detection checks on codes).
- If you have referred a book or any other online material for obtaining a solution, please cite the source. Again don't copy the source *as is* - you may use the source to understand the solution, but write-up the solution in your own words.
- For all the reasons explained in class, you cannot feed these questions into LLMs (Large Language Models like ChatGPT) and cannot use the LLMs' outputs to answer this assignment. Related to this, please also complete the self-declaration statement in the end of your answer sheet pdf.
- Please be advised that *the lesser your reliance on online materials or LLMs* for answering the questions, *the more your understanding* of the concepts will be and *the more prepared you will be for the course exams*.
- Points will be awarded based on how clear, concise and rigorous your solutions are, and how correct your answer is. The weightage of this assignment is 11% towards the overall course grade.

1. **(8 points) [Logistic Regression: Theory and Practice]**

   (a) **(1 point)** Briefly explain why logistic regression is called "regression", despite it being a classification model.

   Solution

   Logistic regression is called "regression" because, it applies a logistic transformation to a liner function $w^T x + b$ to produce the class probability. As the underlying model estimates probabilities through a regression based approach, it is called logistic regression.

   (b) **(3 points)** Consider the following 2-dimensional classification dataset with 8 points:

$$X^T = \begin{bmatrix} -2 & -2 & -1 & 1 & 1 & 2 & 3 & 3 \\ -1 & 2 & 1 & 2 & 1 & 3 & 3 & 2 \end{bmatrix}$$

$$y^T = \begin{bmatrix} +1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 \end{bmatrix}$$

   Run **one iteration** of gradient descent for the logistic loss objective $L(w) = \sum_{i=1}^{8} \log(1 + \exp(-y_i w^T x_i))$ by hand. Use an initial weight vector $w^{(0)} = [0, 0]^T$ and a step size $\eta = 1$. No bias term is required. Show your calculation for the gradient $\nabla L(w^{(0)})$ and the updated weight vector $w^{(1)}$.

   Solution

   The logistic loss objective function is

$$L(w) = \sum_{i=1}^{8} \log(1 + \exp(-y_i w^T x_i))$$

   The gradient of the loss function w.r.to w

$$\begin{aligned} \nabla_w L(w) &= \sum_{i=1}^{8} \frac{1}{1 + \exp(-y_i w^T x_i)} \cdot \exp(-y_i w^T x_i) \cdot (-y_i x_i) \\ &= -\sum_{i=1}^{8} \frac{y_i x_i}{1 + \exp(y_i w^T x_i)} \end{aligned}$$

   Now, the gradient at the initial weight $w^{(0)} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$

$$\begin{aligned} \nabla_w L(w^{(0)}) &= -\sum_{i=1}^{8} \frac{y_i x_i}{1 + \exp(y_i {w^{(0)}}^T x_i)} = -\frac{1}{2} \sum_{i=1}^{8} y_i x_i \\ &= -\frac{1}{2} \left(\begin{bmatrix} -2 \\ -1 \end{bmatrix} + \begin{bmatrix} -2 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \begin{bmatrix} 3 \\ 2 \end{bmatrix} \right) \\ &= \begin{bmatrix} 11/2 \\ 3/2 \end{bmatrix} = \begin{bmatrix} 5.5 \\ 1.5 \end{bmatrix} \end{aligned}$$

   By the gradient descend update rule

$$\begin{aligned} w^{(1)} &= w^{(0)} - \eta \nabla_w (w^{(0)}) \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 5.5 \\ 1.5 \end{bmatrix} = \begin{bmatrix} -5.5 \\ -1.5 \end{bmatrix} \end{aligned}$$

(c) **(1 point)** Using the updated weight vector $w^{(1)}$ you just calculated, what is the equation of the decision boundary (where $P(y = 1|x) = 0.5$)?

Solution

The equation of the decision boundary

$$\begin{aligned} P(y = 1 \mid x) = 0.5 = \sigma({w^{(1)}}^T x) \\ \Rightarrow 0.5 = \frac{1}{1 + \exp\left(-{w^{(1)}}^T x\right)} \\ \Rightarrow 2 = 1 + \exp\left(-{w^{(1)}}^T x\right) \\ \Rightarrow {w^{(1)}}^T x = 0 \Rightarrow \begin{bmatrix} -5.5 & -1.5 \end{bmatrix} x = 0 \end{aligned}$$

(d) **(1 point)** Using $w^{(1)}$, what is the probability $P(y = 1|x)$ for a new test point $x_{new} = [2, 2]^T$? (Recall $P(y = 1|x) = \sigma(w^T x)$, where $\sigma(z) = 1/(1 + e^{-z})$).

Solution

The probability

$$P(y = 1 \mid x_{new}) = \sigma\left({w^{(1)}}^T x_{new}\right) = \sigma(-14) = 1/(1 + e^{14}) = 8.315 \times 10^{-7} \approx 0$$

(e) **(1 point)** If you were to continue training this model and found it was overfitting, you might add L2 regularization. Explain what effect adding strong L2 regularization (a large λ) would have on the magnitude of the final, converged weight vector $\hat{w}$.

Solution

If we add strong L2 regularization then the model minimizes the norm of the weight vector, leading to a smaller magnitude final weight vector $\hat{w}$, which in turn leads to underfitting of the data.

2. **(8 points) [THE KERNEL TRICK AND ITS ALTERNATIVES]**
Consider a 1D dataset that is not linearly separable:

- **Class +1:** $x = -2$, $x = 2$
- **Class -1:** $x = -1$, $x = 1$

We want to make this dataset linearly separable by mapping it to a higher-dimensional space using a feature map $\phi(x)$.

(a) **(1 point)** In your own words, what is the primary purpose of the kernel trick in algorithms like SVM? Why is it useful?

Solution

As SVM is an algorithm that maximizes the linear decision boundary in the dataset, if the dataset is not linearly separable, then SVM will fail to perform classification. However, the kernel trick maps the datapoints to higher-order feature dimensions, where they can be linearly separable, allowing SVM to work.

(b) **(2 points)** Propose a simple feature mapping $\phi(x) : \mathbb{R} \to \mathbb{R}^2$ that transforms the given 1D data into a 2D space where it becomes linearly separable. Plot or list the transformed coordinates of the four data points to show they are now separable.

Solution

Let the feature mapping be

$$\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

Figure 1 shows the full feature mapping and decision boundary.

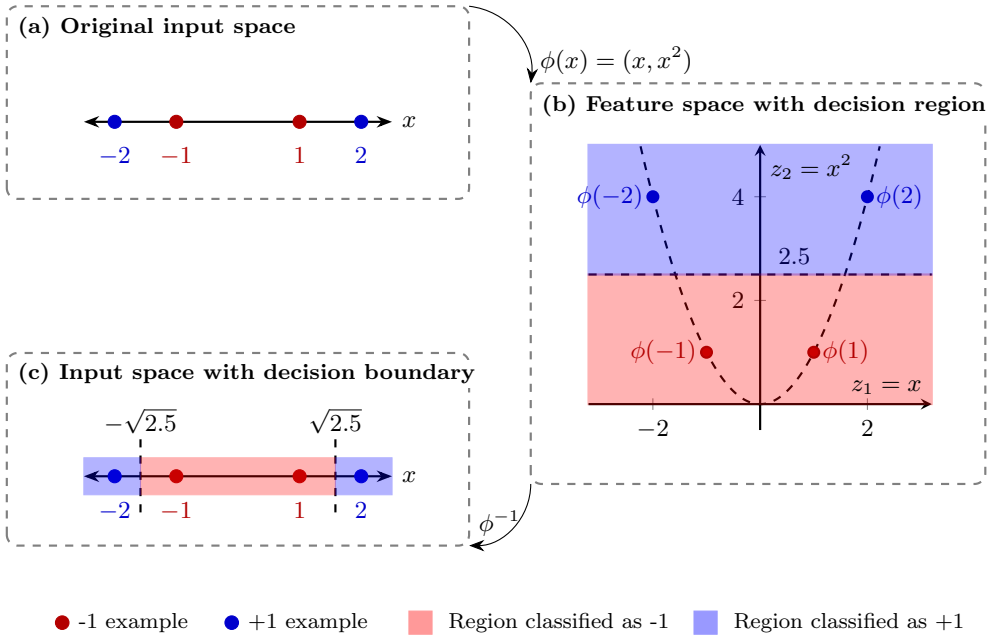

Figure 1: Feature mapping visualization

(c) **(2 points)** For the feature mapping $\phi(x)$ you defined in part (b), derive the corresponding kernel function $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. Show your work.

Solution

For the feature map

$$\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

The kernel function

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \begin{bmatrix} x_i & x_i^2 \end{bmatrix} \begin{bmatrix} x_j \\ x_j^2 \end{bmatrix} = x_i x_j + x_i^2 x_j^2$$

Since this kernel is explicitly defined as an inner product in the feature space, it is automatically a valid kernel.

(d) **(3 points)** Now, suppose you did not want to use a kernel (i.e., you stay in the original 1D space and build a linear SVM model). Could this 1D dataset be solved

using a **hard-margin SVM**? Why or why not?

Could a **soft-margin SVM** be used? If yes, justify your answer by describing what the resulting linear classifier would do (i.e., where would its 1D decision boundary likely be, and which points would it misclassify? consider two values of the model complexity hyperparameter C: 1 and 1000).

Solution

As we know, the hard-margin SVM will not tolerate any misclassification; therefore, there is no such decision boundary in 1D, hence the hard-margin SVM cannot solve the problem.

3. **(8 points) [Constructing Valid Kernels]**
Let K_1 and K_2 be a valid kernel functions, with feature mapping $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{d_1}$ and $\varphi_2 : \mathbb{R}^d \to \mathbb{R}^{d_2}$.

   (i) **(2 points)** Show that $K_3 = K_1 + K_2$ is also a valid kernel. Give the feature mapping φ_3 corresponding to K_3 in terms of φ_1 and φ_2.

Solution

As K_1 and K_2 are valid kernel functions with corresponding feature mapping $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{d_1}$ and $\varphi_2 : \mathbb{R}^d \to \mathbb{R}^{d_2}$, we can write

$$K_1 = \langle \varphi_1(x_i), \varphi_1(x_j) \rangle = \varphi_1(x_i)^T \varphi_1(x_j)$$
$$K_2 = \langle \varphi_2(x_i), \varphi_2(x_j) \rangle = \varphi_2(x_i)^T \varphi_2(x_j)$$

where $x_i, x_j \in \mathbb{R}^d$. Now, define

$$\begin{aligned} K_3 = K_1 + K_2 &= \langle \varphi_1(x_i), \varphi_1(x_j) \rangle + \langle \varphi_2(x_i), \varphi_2(x_j) \rangle \\ &= \varphi_1(x_i)^T \varphi_1(x_j) + \varphi_2(x_i)^T \varphi_2(x_j) \\ &= \begin{bmatrix} \varphi_1(x_i)^T & \varphi_2(x_i^T) \end{bmatrix} \begin{bmatrix} \varphi_1(x_j) \\ \varphi_2(x_j) \end{bmatrix} \\ &= \left\langle \begin{bmatrix} \varphi_1(x_i) \\ \varphi_2(x_i) \end{bmatrix}, \begin{bmatrix} \varphi_1(x_j) \\ \varphi_2(x_j) \end{bmatrix} \right\rangle \end{aligned}$$

Hence, K_3 is a valid kernel corresponding to the feature mapping

$$\varphi_3 = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} : \mathbb{R}^d \to \mathbb{R}^{d_1 + d_2}$$

   (ii) **(2 points)** Show that $K_4 = K_1 \cdot K_2$ is also a valid kernel. Give the feature mapping φ_4 corresponding to K_4 in terms of φ_1 and φ_2.

Solution

As K_1 and K_2 are valid kernel functions with corresponding feature mapping $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{d_1}$ and $\varphi_2 : \mathbb{R}^d \to \mathbb{R}^{d_2}$, we can write

$$K_1 = \langle \varphi_1(x_i), \varphi_1(x_j) \rangle = \varphi_1(x_i)^T \varphi_1(x_j)$$
$$K_2 = \langle \varphi_2(x_i), \varphi_2(x_j) \rangle = \varphi_2(x_i)^T \varphi_2(x_j)$$

where $x_i, x_j \in \mathbb{R}^d$. Now, define

$$\begin{aligned}
K_4 = K_1 \cdot K_2 &= \langle \varphi_1(x_i), \varphi_1(x_j) \rangle \cdot \langle \varphi_2(x_i), \varphi_2(x_j) \rangle \\
&= \left\langle \begin{bmatrix} \varphi_{11}(x_i) \\ \varphi_{12}(x_i) \\ \vdots \\ \varphi_{1d_1}(x_i) \end{bmatrix}, \begin{bmatrix} \varphi_{11}(x_j) \\ \varphi_{12}(x_j) \\ \vdots \\ \varphi_{1d_1}(x_j) \end{bmatrix} \right\rangle \cdot \left\langle \begin{bmatrix} \varphi_{21}(x_i) \\ \varphi_{22}(x_i) \\ \vdots \\ \varphi_{2d_2}(x_i) \end{bmatrix}, \begin{bmatrix} \varphi_{21}(x_j) \\ \varphi_{22}(x_j) \\ \vdots \\ \varphi_{2d_2}(x_j) \end{bmatrix} \right\rangle \\
&= \left[\sum_{k=1}^{d_1} \varphi_{1k}(x_i)\varphi_{1k}(x_j) \right] \left[\sum_{l=1}^{d_2} \varphi_{2l}(x_i)\varphi_{2l}(x_j) \right] \\
&= \sum_{l=1}^{d_2} \sum_{k=1}^{d_1} \varphi_{1k}(x_i)\varphi_{2l}(x_i)\varphi_{1k}(x_j)\varphi_{2l}(x_j) \\
&= \left\langle \begin{bmatrix} \varphi_{11}(x_i)\varphi_{21}(x_i) \\ \vdots \\ \varphi_{1d_1}(x_i)\varphi_{2d_2}(x_i) \end{bmatrix}, \begin{bmatrix} \varphi_{11}(x_j)\varphi_{21}(x_j) \\ \vdots \\ \varphi_{1d_1}(x_j)\varphi_{2d_2}(x_j) \end{bmatrix} \right\rangle \\
&= \langle \varphi_1(x_i) \otimes \varphi_2(x_i), \varphi_1(x_j) \otimes \varphi_2(x_j) \rangle
\end{aligned}$$

Hence, K_4 is a valid kernel corresponding to the feature mapping

$$\varphi_4 = \varphi_1 \otimes \varphi_2 : \mathbb{R}^d \to \mathbb{R}^{d_1 d_2}$$

where $\otimes$ is Kronecker product.

(iii) **(2 points)** Show that $K_5 = f(u)K_1(u, v)f(v)$ is also a valid kernel for any function $f : \mathbb{R}^d \to \mathbb{R}$. Give the feature mapping φ_5 corresponding to K_5 in terms of φ_1 and f.

Solution

As K_1 is a valid kernel functions with feature mapping $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{d_1}$, we can write

$$K_1(u, v) = \varphi_1(u)^T \varphi_1(v)$$

where $u, v \in \mathbb{R}^d$. Now define

$$\begin{aligned}
K_5(u, v) &= f(u)K_1(u, v)f(v) \\
&= f(u)\varphi_1(u)^T \varphi_1(v)f(v) \\
&= \langle f(u)\varphi_1(u), f(v)\varphi_1(v) \rangle
\end{aligned}$$

where $f : \mathbb{R}^d \to \mathbb{R}$.
Hence, K_5 is a valid kernel corresponding to the feature mapping

$$\varphi_5 = \varphi_1 f : \mathbb{R}^d \to \mathbb{R}^{d_1}$$

(iv) **(2 points)** Show that a Kernel given by $K(u, v) = \exp(2u^T v)$ is a valid kernel. [Hint: Use the results above on a polynomial expansion of $\exp(t)$.]

Solution

By Taylor series we have

$$K(u,v) = \exp(2u^Tv) = \sum_{k=0}^{\infty} \frac{(2u^Tv)^k}{k!} = \sum_{k=0}^{\infty} \frac{2^k}{k!}(u^Tv)^k = \sum_{k=0}^{\infty} K_k(u,v)$$

where $K_k(u,v) = \frac{2^k}{k!}(u^Tv)^k$.
Equivalently we can write

$$K_k(u,v) = f_k(u)K_k^{'}(u,v)f_k(v)$$

where $f_k = \sqrt{\frac{2^k}{k!}}$ and $K_k^{'} = (u^Tv)^k$.
Now

$$K_k^{'}(u,v) = (u^Tv)^k = \langle u,v\rangle^k = \left\langle u^{\otimes k}, v^{\otimes k}\right\rangle = \left\langle \varphi_k^{'}(u), \varphi_k^{'}(v)\right\rangle$$

where $\varphi_k^{'}(\cdot) = (\cdot)^{\otimes k}$ denotes the k^{th} tensor power. Hence, $K_k^{'}$ is a valid kernel corresponding to the feature map $\varphi_k^{'}$.
Moreover, using the result from the previous problem (that $f(u)K_1(u,v)f(v)$ is a valid kernel), we have

$$\begin{aligned} K_k(u,v) &= f_k(u)\left\langle \varphi_k'(u), \varphi_k'(v)\right\rangle f_k(v) \\ &= \left\langle f_k(u)\varphi_k'(u), f_k(v)\varphi_k'(v)\right\rangle. \end{aligned}$$

This shows that each $K_k(u,v)$ is a valid kernel with feature map

$$\varphi_k(u) = f_k(u)\varphi_k'(u) = \sqrt{\frac{2^k}{k!}}u^{\otimes k}.$$

Finally,

$$K(u,v) = \sum_{k=0}^{\infty} K_k(u,v) = \sum_{k=0}^{\infty} \langle \varphi_k(u), \varphi_k(v)\rangle = \left\langle \begin{bmatrix} \varphi_0(u) \\ \varphi_1(u) \\ \vdots \end{bmatrix}, \begin{bmatrix} \varphi_0(v) \\ \varphi_1(v) \\ \vdots \end{bmatrix} \right\rangle$$

Hence, K is a valid kernel corresponding to the feature mapping

$$\varphi(u) = \begin{bmatrix} \varphi_0(u) \\ \varphi_1(u) \\ \varphi_2(u) \\ \vdots \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{2^0}{0!}}u^{\otimes 0} \\ \sqrt{\frac{2^1}{1!}}u^{\otimes 1} \\ \sqrt{\frac{2^2}{2!}}u^{\otimes 2} \\ \vdots \end{bmatrix}$$

where $u^{\otimes 0} = 1$.

(v) **(0 points) (Optional)** Show that a Kernel given by $K(u,v) = \exp(-\|u-v\|^2)$ is a valid kernel. [Hint: Use the last two parts' results.]

Solution

The given kernel

$$K(u,v) = e^{-\|u-v\|_2^2} = e^{-\|u\|_2^2 + 2u^T v - \|v\|_2^2} = e^{-\|u\|_2^2} \cdot e^{2u^T v} \cdot e^{-\|v\|_2^2}.$$

Using the results from previous problems, we can write

$$K(u,v) = \langle f(u)\varphi(u), f(v)\varphi(v)\rangle = \langle \tilde{\varphi}(u), \tilde{\varphi}(v)\rangle$$

where

$$f(u) = \exp(-\|u\|_2^2), \quad \varphi(u) = \begin{bmatrix} \sqrt{\frac{2^0}{0!}}u^{\otimes 0} \\ \sqrt{\frac{2^1}{1!}}u^{\otimes 1} \\ \sqrt{\frac{2^2}{2!}}u^{\otimes 2} \\ \vdots \end{bmatrix}$$

Hence, K is a valid kernel corresponding to the feature mapping

$$\tilde{\varphi}(u) = e^{-\|u\|_2^2} \begin{bmatrix} \sqrt{\frac{2^0}{0!}}u^{\otimes 0} \\ \sqrt{\frac{2^1}{1!}}u^{\otimes 1} \\ \sqrt{\frac{2^2}{2!}}u^{\otimes 2} \\ \vdots \end{bmatrix}$$

4. **(8 points) [Primal, Dual, and the Representer Theorem]**
Consider the primal objective function for Ridge Regression in a feature space (with features $\phi(x)$):

$$\min_{w \in \mathbb{R}^{d'}} \left(L(w) := \frac{1}{2}||\Phi w - y||^2 + \frac{1}{2}||w||^2 \right)$$

where Φ is the $n \times d'$ design matrix with rows $\phi(x_i)^T$, y is the $n \times 1$ target vector, w is the $d' \times 1$ weight vector, and $\lambda > 0$ is the regularization parameter.

(a) **(1 point)** Using the Representer Theorem, get the dual version of the given primal problem.

Solution

Let, $u + v$ where $u \in \text{null}(\Phi);\ v \in \text{col}(\Phi^T)$

The objective

$$\min_{w\in\mathbb{R}^{d'}} \frac{1}{2}\|\Phi w - y\|_2^2 + \frac{1}{2}\|w\|_2^2$$
$$\Updownarrow$$
$$\min_{\substack{u\in\text{null}(\Phi)\\ v\in\text{col}(\Phi^T)}} \frac{1}{2}\|\Phi(u+v) - y\|_2^2 + \frac{1}{2}\|u\|_2^2 + \frac{1}{2}\|v\|_2^2$$
$$\Updownarrow$$
$$\min_{v\in\text{col}(\Phi^T)} \frac{1}{2}\|\Phi v - y\|_2^2 + \frac{1}{2}\|v\|_2^2$$
$$\Updownarrow$$
$$\min_{\alpha\in\mathbb{R}^n} \frac{1}{2}\ \|\Phi\Phi^T\alpha - y\|_2^2 + \frac{1}{2}\|\Phi^T\alpha\|_2^2$$

Hence, the dual problem is

$$\min_{\alpha\in\mathbb{R}^n} R(\alpha) := \frac{1}{2}\|\Phi\Phi^T\alpha - y\|_2^2 + \frac{1}{2}\|\Phi^T\alpha\|_2^2$$

(b) **(2 points)** Find the optimal $\hat{\alpha}$ by computing the gradient of the dual objective, $\nabla R(\alpha)$, setting it to zero, and solving for α.

Solution

The gradient of $R(\alpha)$

$$\begin{aligned}\nabla R(\alpha) &= \Phi\Phi^T\left(\Phi\Phi^T\alpha - y\right) + \Phi\Phi^T\alpha \\ &= \Phi\Phi^T\left((\Phi\Phi^T + \mathbb{I})\alpha - y\right)\end{aligned}$$

Now setting $\nabla R(\alpha)$ to zero

$$\begin{aligned}&\nabla R(\alpha) = 0 \\ \Rightarrow &\Phi\Phi^T\left((\Phi\Phi^T + \mathbb{I})\alpha - y\right) = 0 \\ \Rightarrow &\hat{\alpha} = \left(\Phi\Phi^T + \mathbb{I}\right)^{-1} y\end{aligned}$$

(c) **(1 point)** Show how the learned function $f(x) = \hat{w}^T\phi(x)$ can be evaluated on a new test point x using only the dual solution $\hat{\alpha}$ and kernel function evaluations $K(x_i, x) = \phi(x_i)^T\phi(x)$.

Solution

The learned function can be written as

$$\begin{aligned}f(x) &= \hat{w}^T\phi(x) \\ &= \hat{\alpha}^T\Phi\phi(x) = \hat{\alpha}^T\begin{bmatrix}\phi(x_1)^T \\ \vdots \\ \phi(x_n)^T\end{bmatrix}\phi(x) \\ &= \sum_{i=1}^n \hat{\alpha}_i\phi(x_i)^T\phi(x) = \sum_{i=1}^n \hat{\alpha}_i K(x_i, x)\end{aligned}$$

(d) **(2 points)** The standard primal solution $\hat{w}_{primal}$ is found by solving $\nabla L(w) = 0$ for w. The dual-derived solution is $\hat{w}_{dual} = \Phi^T \hat{\alpha}$.

Argue that both are identical and use it to prove the push-through matrix identity (also related to the Sherman–Morrison–Woodbury identity):
$(\Phi^T\Phi + I)^{-1}\Phi^T = \Phi^T(\Phi\Phi^T + I)^{-1}$.

Solution

The standard primal solution is

$$\nabla L(w) = 0$$
$$\Rightarrow \hat{w}_{primal} = \left(\Phi^T\Phi + \mathbb{I}\right)^{-1}\Phi^T y$$

and the dual-derived solution is

$$\hat{w}_{dual} = \Phi^T\hat{\alpha} = \Phi^T\left(\Phi\Phi^T + \mathbb{I}\right)^{-1} y$$

Since these two are equal (because convex objective in w) for every y , the linear operators must be same

$$\left(\Phi^T\Phi + \mathbb{I}\right)^{-1}\Phi^T = \Phi^T\left(\Phi\Phi^T + \mathbb{I}\right)^{-1}$$

which is exactly the push-through matrix identity.

(e) **(2 points)** Compare the computational complexity for calculating the two $\hat{w}$ solutions from part (d).

Provide a step-by-step breakdown of the time complexity (in asymptotic big-Oh notation) for computing $\hat{w}_{primal}$ and $\hat{w}_{dual}$. Based on your analysis, when is the primal solution more efficient to compute, and when is the dual solution more efficient?

Solution

For primal problem

- forming $\Phi^T\Phi$ cost : $O(d'^2 n)$
- forming $(\Phi^T\Phi + \mathbb{I})^{-1}$ cost : $O(d'^3)$
- forming $\Phi^T y$ cost : $O(d'n)$
- multiplying $(\Phi^T\Phi + \mathbb{I})^{-1}$ with $\Phi^T y$ cost : $O(d'^2)$

the total cost: $O(nd'^2 + d'^3)$.
For dual problem

- forming $\Phi\Phi^T$ cost : $O(n^2 d')$
- forming $(\Phi\Phi^T + \mathbb{I})^{-1}$ cost : $O(n^3)$
- multiplying Φ^T with $(\Phi\Phi^T + \mathbb{I})^{-1}$ cost : $O(n^2 d')$
- multiplying $\Phi^T(\Phi\Phi^T + \mathbb{I})^{-1}$ with y cost : $O(nd')$

the total cost: $O(n^2 d' + n^3)$.
Hence, when $d' \ll n$(i.e., feature space is smaller then the number of data points) solving the primal problem is computationally efficient (because

$O(nd'^2 + d'^3) \ll O(n^2d' + n^3)$) whereas if $d' \gg n$(i.e., number of data points is smaller then the feature space) solving the dual problem is computationally efficient (because $O(nd'^2 + d'^3) \gg O(n^2d' + n^3)$).

5. **(8 points) [MAXIMIZING THE MARGIN WITH SVMS]**
Consider a simple, linearly separable 2D dataset for a Support Vector Machine (SVM) classifier:

   - **Class +1:** $(2, 3)$, $(3, 3)$
   - **Class -1:** $(1, 1)$, $(2, 1)$

   The decision boundary is a hyperplane of the form $w^T x + b = 0$. For a canonical hyperplane, the margin is $2/||w||$, and for all data points (x_i, y_i), we have $y_i(w^T x_i + b) \geq 1$.

   (a) **(3 points)** By inspection or simple geometry, determine the equation of the maximal margin hyperplane that separates these two classes. Identify the weight vector w and the intercept/bias b.

Solution

As we can see from Figure 2 the decision boundary which maximize the margin is

$$x_2 = 2 \Rightarrow \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 2 = 0$$

writing it in standard hyperplane form $w^T x + b = 0$, we have $w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $b = -2$

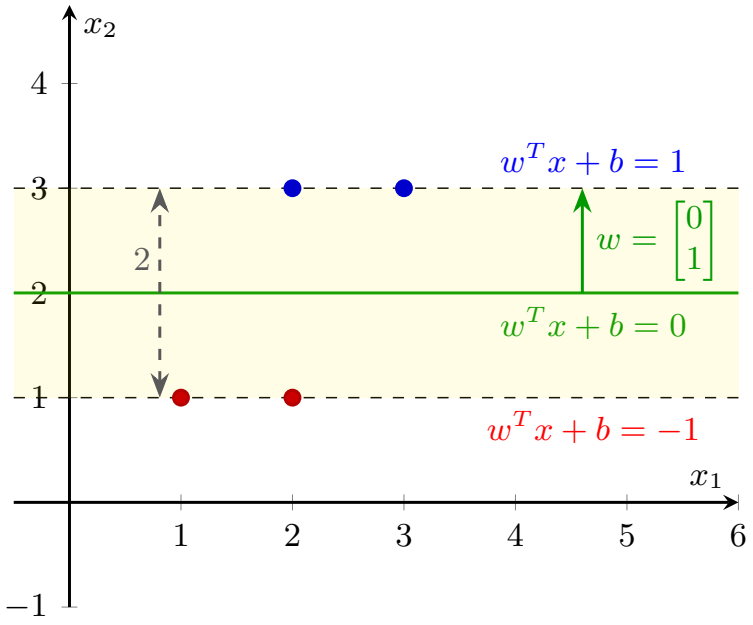

Figure 2: Data points with decision boundary

   (b) **(3 points)** Identify the support vectors from the dataset. How do these points satisfy the condition $y_i(w^T x_i + b) = 1$?

Solution

For class $+1$ the support vectors are $(2,3)$ and $(3,3)$.
For class -1 the support vectors are $(1,1)$ and $(2,1)$.
Consider the datapoints $(2,3)$ and $(3,3)$. For these point $y=+1$, so

$$y_i(w^T x_i + b) = +1\left(\begin{bmatrix}0 & 1\end{bmatrix} x_i - 2\right) = 1$$

similarly for the datapoints $(1,1)$ and $(2,1)$ $y=-1$ so, we have

$$y_i(w^T x_i + b) = -1\left(\begin{bmatrix}0 & 1\end{bmatrix} x_i - 2\right) = 1$$

(c) **(2 points)** Suppose we remove the point $(1,1)$ from the dataset. Would the decision boundary change? Now, suppose we instead remove the point $(2,3)$. Would the decision boundary change in this case? Explain your reasoning.

Solution

If we remove the point $(1,1)$, the decision boundary remains unchanged. Since removing the support vector does not increase the distance between the class hulls, the previous optimal hyperplane remains optimal.
Now, if we remove $(2,3)$, the positive class is only at $(3,3)$, which will be a support vector, and the nearest point from the negative hull(i.e., convex combination of $(1,1)$ and $(2,1)$) is $(2,1)$. Moreover, the maximum margin hyperplane is the perpendicular bisector of the segment joining the points $(2,1)$ and $(3,3)$. Therefore, the perpendicular bisector has a midpoint of $(2.5,2)$ and a normal vector of $(1,2)$. Finally, the equation is

$$\begin{aligned}(x_1 - 2.5) + 2(x_2 - 2) = 0 \Rightarrow x_1 + 2x_2 - 6.5 = 0\\ \Rightarrow \begin{bmatrix}\frac{2}{5} & \frac{4}{5}\end{bmatrix} x - 2.6 = 0 \Rightarrow w^T x + b = 0\end{aligned}$$

where $w = \begin{bmatrix}0.4\\0.8\end{bmatrix}$ and $b = -2.6$.
Figure 3 shows the new decision boundary after removing the datapoint $(2,3)$.

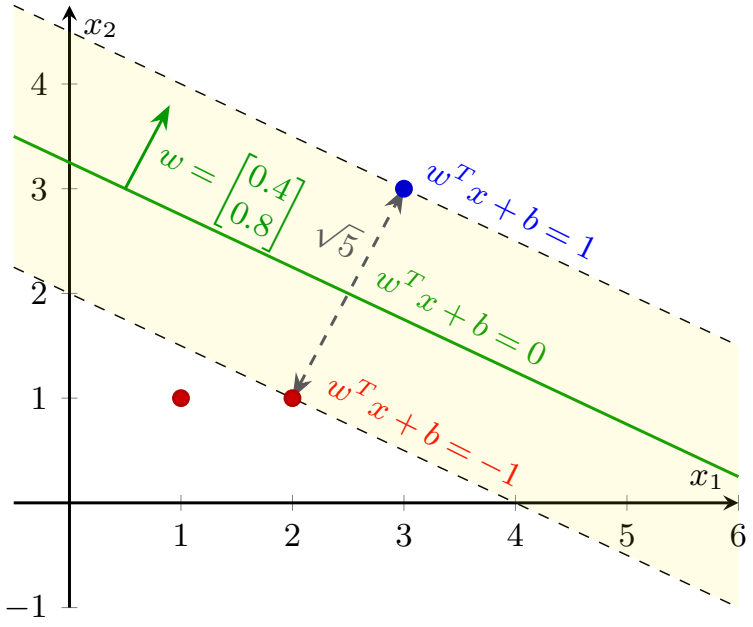

Figure 3: New decision boundary after removing datapoint $(2, 3)$

6. **(15 points) [GUESS-TIMATING BIAS AND VARIANCE]**
You are given a dataset consisting of 100 datapoints in this folder. You have to fit a polynomial ridge regression model to this data.

A model's error can be decomposed into bias, variance, and noise. A "learning curve" provides an opportunity to determine the bias and variance of machine learning models, and to identify models that suffer from high bias (underfitting) or high variance (overfitting). The "learning curve" typically shows the training error and validation/testing error on the y-axis and the model complexity on the x-axis.

(a) **(2 points)** Read the last Section (Section 4) on "Bias and Variance in practice" in this document, and summarize briefly how you will heuristically find whether your model suffers from (i) high bias, or (ii) high variance, using only the train and validation/test errors of the model.

Solution

As mentioned in the document (i) high bias leads to under fitting and (ii) high variance leads to over fitting on the training data. Therefore heuristically I can say if the model performs poorly(i.e., high error) on the training as well as validation/test dataset it has high bias(under fit) and if the model performs good(i.e., low error) in training dataset but suffers in test/validation dataset then it has high variance(over fit).

(b) **(4 points)** Start with the code for polynomial regression and add quadratic regularization functionality to the code. That is, your code should do polynomial regression with quadratic regularization that takes degree d and regularization parameter λ as input. Do not use any inbuilt functions from Python packages (except for plotting functions, functions to compute polynomial features for each data point, and functions for (pseudo)inverse of matrices).

Solution

Written code and attached below

(c) **(4 points)** Run your code on the provided dataset for degree $d = 24$ and each λ in the set:

$$\{10^{-15}, 10^{-9}, 10^{-6}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{6}, 10^{9}, 10^{15}\}$$

i. Perform 5-fold cross-validation on the 100 data points (20 data points in each fold). For each validation fold, compute both training (4-fold-based) and validation (1-fold-based) errors using the mean squared error measure.

ii. Calculate the average training and validation errors across the 5 folds.

Solution

The average training and validation errors across the 5 folds are as follows

λ	Training error	Validation error
10^{-15}	10.4015	62.6490
10^{-9}	0.9076	1.3552
10^{-6}	0.9284	1.2113
10^{-3}	1.0180	1.2334
10^{-2}	1.1219	1.2577
10^{-1}	1.3129	1.4736
1	1.6767	1.8537
10	2.4839	2.7989
10^{2}	3.6237	4.0391
10^{3}	4.3157	4.3851
10^{6}	4.4512	4.5091
10^{9}	4.4440	4.5753
10^{15}	4.4542	4.4830

(d) **(3 points)** Construct a learning curve by plotting the average training and validation errors against the model complexity ($\log_{10} \lambda$). Based on this learning curve, identify the (i) model with the highest bias, (ii) model with the highest variance?, and (iii) the model that will work best on some unseen data.

Solution

From the Figure 4 we can see

(i) **Model with highest bias:** 10^{15}

(ii) **Model with highest variance:** 10^{-15}

(iii) **Model that will work best on some unseen data:** 10^{-6}

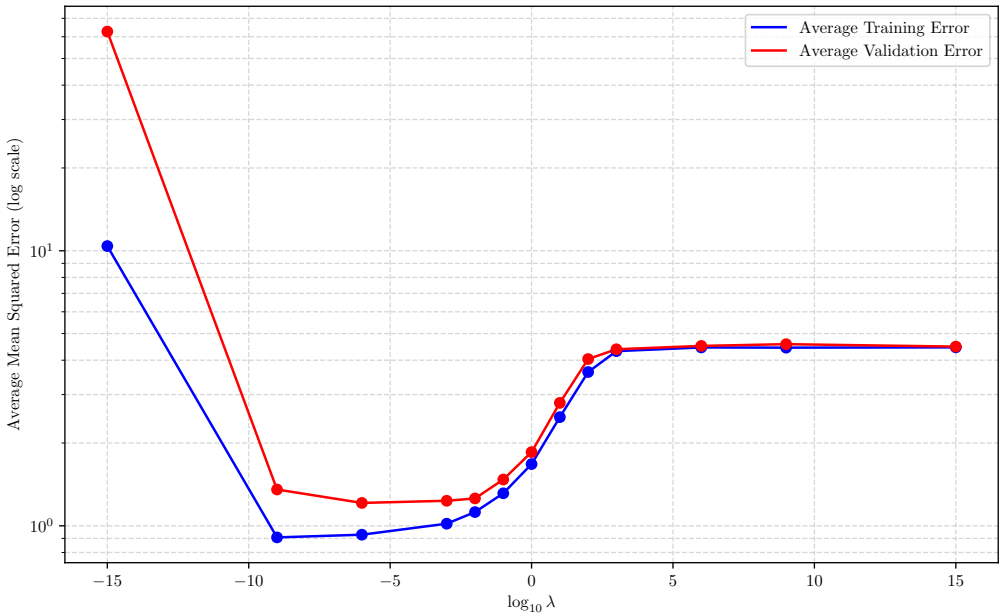

Figure 4: Learning Curve

(e) **(2 points)** Plot the fitted curve to the given data ($\hat{y}$ against x curve) for the three models reported in part (d) and superimposed with the training and validation data

points for any one-fold.

Solution

Figure 5 shows the fitted curve to the given data for models reported in part (d) with one fold training and validation data points.

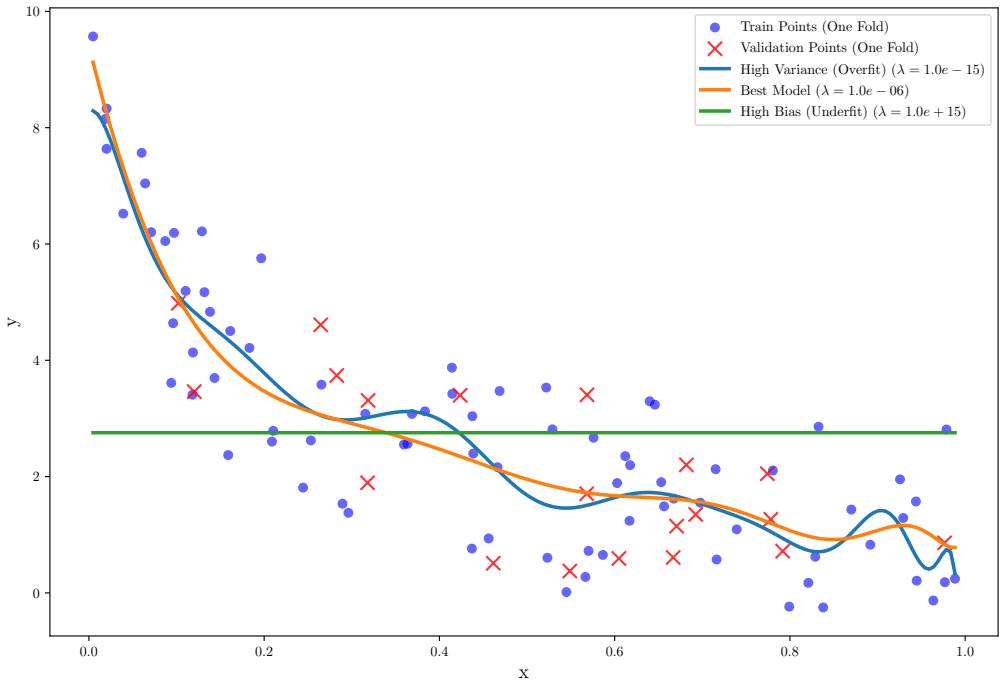

Figure 5: Fitted Polynomial (degree 24) curves

Self Declaration

I, RITABRATA MANDAL , swear on my honour that I have prepared and written the answers for this assignment and associated code by myself and have not copied it from the internet, any LLM's output, or other students.
