The Chinese University of Hong Kong, Shenzhen
School of Data Science

DDA 3020 · Homework 2
Due: 23:59, March 24th, 2024

Instructions:

- This assignment accounts for 15/100 of the final score.
- You must independently complete each assignment.
- Late submission will get discounted score: 20 percent discount on $(0, 24]$ hours late; 50 percent discount on $(24, 120]$ hours late; no score on late submission of more than 120 hours.

---

1 Written Problems (50 pts.)

1.1 Problem: Overfitting and Regularized Logisitc Regression (5 pts.)

1) (2 pts.) Plot the sigmoid function $1/(1+\exp^{-wX})$ for increasing weights $w \in \{1, 5, 100\}$ with $X \in \mathbb{R}$. A qualitative sketch will suffice. Utilize these plots to explain why large weights can lead to overfitting in logistic regression.
2) (3 pts.) To mitigate overfitting, it is preferable to have smaller weights. To accomplish this, rather than utilizing maximum conditional likelihood estimation M(C)LE for logistic regression:

$$\max_{w_0,\cdots,w_d} \prod_{i=1}^{n} P(Y_i|X_i, w_0, \cdots, w_d), \tag{1}$$

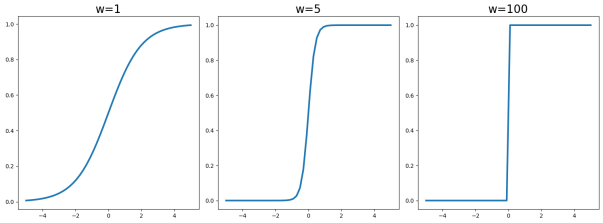
we can consider maximum conditional a posterior M(C)AP estimation:

$$\max_{w_0,\cdots,w_d} \prod_{i=1}^{n} P(Y_i|X_i, w_0, \cdots, w_d)P(w_0, \cdots, w_d), \tag{2}$$

where $P(w_0, \cdots, w_d)$ is a prior on the weights.
Given a standard Gaussian prior $\mathcal{N}(0, \mathbf{I})$ for the weight vector w, please derive the **gradient ascent** update rules for the weights and explain why M(C)AP can address overfitting issue.

Solution 1.1

1) Plot the sigmoid function $1/(1+e^{-w^\top x})$ for increasing weight $w \in \{1, 5, 100\}$.



With large weights, small changes in input can lead to a large change in probability of class, leading to easy flipping of the predicted output. This is the intuition behind why it overfits.

2) Akin to maximum conditional likelihood, we consider to maximize the log conditional posterior:

$$\mathcal{L}(\mathbf{w}) = \log(P(\mathbf{w}) \prod_{i=1}^{n} P(y^i|x^i;\mathbf{w})),$$
$$\text{where } P(\mathbf{w}) = \prod_{j=0}^{d} \frac{1}{\sqrt{2\pi}} \exp(-\frac{w_j^2}{2}).$$

The negative log conditional posterior is:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \log P(\mathbf{w}) + \log \prod_{i=1}^{n} P(y^i|x^i;\mathbf{w}) \\ &= \log \prod_{j=0}^{d} \frac{1}{\sqrt{2\pi}} \exp(-\frac{w_j^2}{2}) + \log \prod_{i=1}^{n} P(y^i|x^i;\mathbf{w}) \\ &= -\sum_j \frac{w_j^2}{2} + \sum_i \log P(y^i|x^i;\mathbf{w}).\end{aligned}$$

The second term is same as derived in class for unregularized case. Therefore, the gradient of the negative log conditional posterior is:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_j} = -w_j + \sum_{i=1}^{n} x^i(y^i - P(y^i|x^i;\mathbf{w})).$$

The final update rule is

$$w_j \leftarrow w_j + \alpha(-w_j + \sum_{i=1}^{n} x^i \Big(y^i - P(y^i|x^i;\mathbf{w}) \Big).$$

1.2 Problem: Multi-class Logisitc Regression (16 pts.)

In this question, we will derive the "multi-class logistic regression" algorithm, assuming the dataset $\mathcal{D}$ is d-dimensional (with d features) and contains n entries.
Given a training set $\{(x^i, y^i)|i = 1, \cdots, n\}$ where $x^i \in \mathbb{R}^{d+1}$ is a feature vector and $y^i \in \mathbb{R}^k$ is a one-hot encoded binary vector with k entries representing classes. In a one-hot vector, the corresponding class label is 1, and all other entries are 0s. For example, if the label of x^i is 3, then the corresponding y^i should be $[0, 0, 1, \cdots, 0] \in \mathbb{R}^k$.
We aim to determine the parameters $\hat{w} \in \mathbb{R}^{k \times (d+1)}$ (representing one weight vector for each class)

that maximize the likelihood for the training set, given a parametric model in the following form:

$$p(y_c^i = 1|x^i; w) = \frac{\exp(w_c^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)}. \tag{3}$$

Note that $\frac{\exp(w_c^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)}$ is always between 0 and 1, and $\sum_c p(y_c^i = 1|x^i; w)$ is always 1, which are desired properties of a probability distribution. Therefore, $\frac{\exp(w_c^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)}$ is also known as the softmax function.
Since we know the probability sums to 1, we don't care about predicting the probability of the last (k^{th}) class, since we can calculate $p(y_k^i = 1|x^i; w)$ by:

$$p(y_k^i = 1|x^i; w) = 1 - \frac{\sum_{c'=1}^{k-1} \exp(w_{c'}^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)}. \tag{4}$$

1) (4 pts.) Show the equivalence between the Eq. (5) and Eq. (6). Provide a short justification for why each line follows from the previous one in your derivation. (This is how we store less weights by making use of the fact that the probabilities sum to 1).

$$p(y_c^i = 1|x^i; w) = \frac{\exp(w_c^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)} \tag{5}$$

$$= \begin{cases} \dfrac{\exp(w_c^\top x^i)}{1 + \sum_{c'=1}^{k-1} \exp(w_{c'}^\top x^i)}, & \text{if } c < k \\ \dfrac{1}{1 + \sum_{c'=1}^{k-1} \exp(w_{c'}^\top x^i)}, & \text{if } c = k \end{cases} \tag{6}$$

2) (4 pts.) Derive the conditional log likelihood for softmax regression. For the sake of simplicity, we only consider Eq. (5) as $p(y_c^j|x^j, w)$. Show the equivalence between the Eq. (7) and Eq. (8). Provide a short justification for why each line follows from the previous one in your derivation.

$$\mathcal{L}(w) \equiv \ln \prod_{j=1}^{n} p(y_c^j|x^j, w) \text{ [here } c \text{ is the true class of } x^j\text{]} \tag{7}$$

$$= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j (w_c^\top x^j) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) \right]. \tag{8}$$

3) (4 pts.) Next, we will derive the gradient of the previous expression with respect to the c^{th} class of the weight matrix w_c, i.e., $\frac{\partial \mathcal{L}(w)}{\partial w_c}$, where $\mathcal{L}(w)$ denotes the log likelihood from Eq. (8). We will perform a few steps of the derivation, and then ask you to do one step at the end. If we task the derivative of Eq. (8) with respect to w_c, we get the following experssion:

$$\nabla_{w_c} \mathcal{L}(w) = \nabla_{w_c} \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j (w_c^\top x^j) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) \right]. \tag{9}$$

The blue expression is linear in w_c, so it can be simplified to $\sum_{j=1}^{n} y_c^j x^j$. For the red expression,first we consider a fixed $j \in [1, n]$. Use the chain rule to verify that

$$\nabla_{w_c} \sum_{c=1}^{k} y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) \tag{10}$$

$$= \frac{\exp(w_c^\top x^j)}{\sum_{c'} \exp(w_{c'}^\top x^j)} x^j. \tag{11}$$

4) (2 pts.) Now use 11 (and the previous discussion) to show that overall, Eq. (9), i.e., $\nabla_{w_c} \mathcal{L}(w)$, is equal to

$$\nabla_{w_c} \mathcal{L}(w) = \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 | x^j; w)). \tag{12}$$

5) (2 pts.) Since the log likelihood is concave, it is easy to optimize using gradient ascent. Derive the update rule for **gradient ascent** with respect to learning rate for w_c w.r.t. η, y^j, x^j, and $p(y^j = 1 | x^j; w^{(t)})$. Feel free to index into the vectors using subscripts.

Solution 1.2

1) Note that adding the same value to all the weights does not affect the value of the equation. Therefore, we shift the k^{th} weight to 0.

$$\begin{aligned}
p(y_c^i = 1 | x^i; w) &= \frac{\exp(w_c^\top x^i)}{\sum_{c'} \exp(w_{c'}^\top x^i)} \\
&= \frac{\exp((w_c - w_k)^\top x^i)}{\sum_{c'} \exp((w_{c'} - w_k)^\top x^i)} \\
&= \frac{\exp((w_c - w_k)^\top x^i)}{1 + \sum_{c'}^{k-1} \exp((w_{c'} - w_k)^\top x^i)} \\
&= \begin{cases} \dfrac{\exp(w_c^\top x^i)}{1 + \sum_{c'=1}^{k-1} \exp(w_c^\top x^i)}, & \text{if } c < k \\ \dfrac{1}{1 + \sum_{c'=1}^{k-1} \exp(w_c^\top x^i)}, & \text{if } c = k \end{cases}
\end{aligned}$$

This shows that one can store only $k - 1$ weight vectors by subtracting the k^{th} vector from all other ones (5 implies 6). Equivalently, in logistic regression, it is acceptable to set one of the weight vectors to 0, and so you get 1 after exponentiating it (6 implies 5).

2)

$$\begin{aligned}\mathcal{L}(w) &\equiv \ln \prod_{j=1}^{n} p(y_c^j | x^j, w) \text{ [here } c \text{ is the true class of } x^j\text{]} \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} y_c^i \ln(p(y_c^j | x^j, w)) \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j \ln \frac{\exp(w_c^\top x^j)}{\sum_{c'} \exp(w_{c'}^\top x^j} \right] \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j (w_c^\top x^j) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j \right) \right].\end{aligned}$$

3) First note that the c in ∇_{w_c} is not the same c in $\sum_{c=1}^{k} y_c^j \ln(\sum_{c'} \exp(w_{c'}^\top xj))$. The later c iterates over 1 to k, and so the latter term is independent of c.

$$\begin{aligned}\nabla_{w_c} \sum_{c=1}^{k} y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) &= \nabla_{w_c} \sum_{j=1}^{n} (\sum_{c=1}^{k} y_c^j) \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) \\ &= \nabla_{w_c} \sum_{j=1}^{n} \ln \left(\sum_{c'} \exp(w_{c'}^\top x^j) \right) \\ &= \frac{\exp(w_c^\top x^j)}{\sum_{c'} \exp(w_{c'}^\top x^j)} x^j \\ &= p(y_c^j = 1 | x^j; w) x^j.\end{aligned}$$

4)
Note that Eq. (11) equals $p(y_c^j = 1|x^j; w)x^j$.
Then we have:

$$\sum_{j=1}^{n} \frac{\exp(w_c^\top x^j)}{\sum_{c'} \exp(w_{c'}^\top x^j)} x^j = \sum_{j=1}^{n} p(y_c^j = 1 | x^j; w) x^j.$$

Plugging into Eq. (9), we get

$$\nabla_{w_c} \mathcal{L}(w) = \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 | x^j; w)).$$

5)

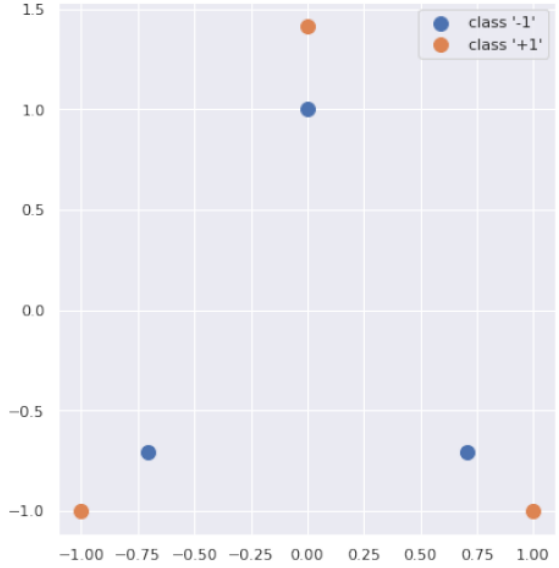
$$w_c \leftarrow w_c + \eta \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 | x^j; w)).$$

1.3 Problem: Support Vector Machine 1 (6 pts.)

Given a binary data set: Class -1: $\{(0,1),(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}}),(-\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}})\}$. Class +1: $\{(0,\sqrt{2}),(1,-1),(-1,-1)\}$.
1) (3 pts.) Could you find an SVM classifier (without slack variable) for this dataset? Please explain your reasoning, possibly with the help of a plot sketch.
2) (3 pts.) Use SVM by expanding the original feature vector $x=[x_1,x_2]$ to $x=[x_1^2,x_2^2]$, find the svm of this given data set and predict the label of $(-\frac{1}{2},\sqrt{2})$.

Solution 1.3

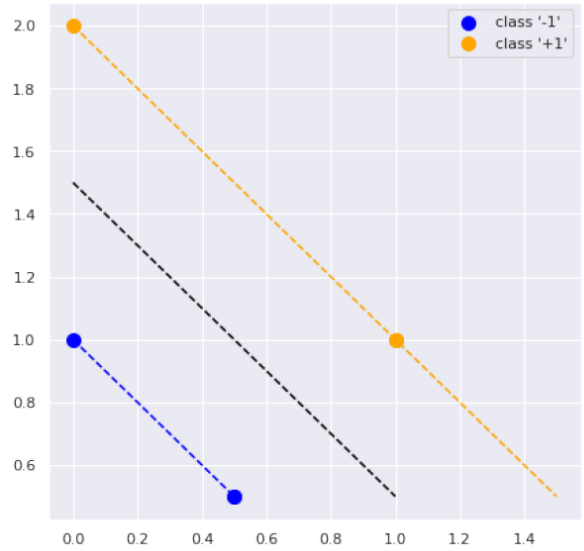
1)



Since the data is not linearly separable, we cannot find an SVM classifier without slack variable for this dataset
2) After expanding the original feature vector, we get the new labels: Class -1: $\{(0,1),(\frac{1}{2},\frac{1}{2}),(\frac{1}{2},\frac{1}{2})\}$. Class +1: $\{(0,2),(1,1),(1,1)\}$.
By plotting the data, we can easily find that all the data are support vectors for the SVM.
And the margin of the SVM is $\frac{\sqrt{2}}{4}$ (Half of the distance between the two black lines).
Since $w_1x_1+w_2x_2+b=0$ when $x_1+x_2=\frac{3}{2}$:

$$w_1x_1+(-x_1+\frac{3}{2})w_2+b=0$$
$$(w_1-w_2)x_1+(b+\frac{3}{2}w_w)=0$$

Then we get $w_1=w_2$ and $b=-\frac{3}{2}w_2$.
Thus the margin is $\frac{1}{||w||}=\frac{1}{\sqrt{w_1^2+w_2^2}}=\frac{\sqrt{2}}{4}$. We get $w_1=w_2=2$ and $b=-3$.
Therefore, decision boundary for the SVM is $f(\phi(x))=(2,2)\phi(x)-3$. We will predict class +1 when $f(\phi(x))>0$ and class -1 otherwise.



Predicting for $(-\frac{1}{2}, \sqrt{2})$:

$$f(\phi((-\frac{1}{2}, \sqrt{2}))) = (2,2)\phi((-\frac{1}{2}, \sqrt{2})) - 3$$
$$= (2,2)(\frac{1}{4}, 2)^\top - 3 = \frac{1}{2} + 4 - 3 > 0.$$

Thus $(-\frac{1}{2}, \sqrt{2})$ belongs to class $+1$.

1.4 Problem: Support Vector Machine 2 (18 pts.)

In this question you have to derive the dual form of SV regression (SVR). Your training data is $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$.
Since the hinge loss that we used in class is only designed for classification we cannot use that for regression. A frequently used loss function for regression is the epsilon sensitive loss:

$$\mathcal{L}_\epsilon(x, y, f) = |y - f(x)|_\epsilon = \max(0, |y - f(x)| - \epsilon). \tag{13}$$

Here x is the input, y is the output, and f is the function used for predicting the label. Using this notation, the SVR cost function is defined as

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \mathcal{L}_\epsilon(x_i, y_i, f), \tag{14}$$

where $f(x) = w^\top x$, and C, $\epsilon > 0$ are parameters.
1) (3 pts.) Introduce appropriate slack variables, and rewrite this problem as a quadratic problem (i.e. quadratic objective with linear constraints). This form is called the Primal form of support vector regression.
2) (2 pts.) Write down the Lagrangian function for the above primal form.
3) (3 pts.) Using the Karush Kunh Tucker conditions derive the dual form.

4) (1 pts.) Can we use quadratic optimization solvers to solve the dual problem?
5) (2 pts.) How would you define support vectors in this problem?
6) (2 pts.) Write down the equation that can be used for predicting the label of unseen sample X.
7) (1 pts.) Is it possible to kernelize this algorithm?
8) (2 pts.) What happens if we change ϵ?
9) (2 pts.) What happens if we change C?

Solution 1.4

1) In the above cost function, ϵ defines the region inside which errors are ignored. The loss function defined above is non-differentiable due to the absolute value in the loss function. We can introduce slack variables ξ and ξ^* to account for errors in points that lie outside the ϵ tube as follows. (These are similar to the slack variables used in classification.)

$$\begin{aligned} y_i - \langle w, x_i \rangle - \epsilon &\leq \xi_i \\ \langle w, x_i \rangle - y_i - \epsilon &\leq \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, \cdots, n \end{aligned}$$

Thus, we can rewrite the primal form as:

$$\min_{w \in \mathbb{R}^m, \xi \in \mathbb{R}^n, \xi^* \in \mathbb{R}^n} \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*).$$

s.t. above tree equations are satisfied.
2) Having the above constraints and objective, the Lagrangian function can be written as follows.

$$\begin{aligned} L = L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) := & \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ & - \sum_{i=1}^{n} (\beta_i \xi_i + \beta_i^* \xi_i^*) \\ & - \sum_{i=1}^{n} \alpha_i (\epsilon + \xi_i - y_i + \langle w, x_i \rangle) \\ & - \sum_{i=1}^{n} \alpha_i^* (\epsilon + \xi_i^* - y_i + \langle w, x_i \rangle), \end{aligned}$$

where the Lagrange multipliers have to satisfy the positivity constraints,

$$\alpha_i, \alpha_i^*, \beta_i, \beta_i^* \geq 0, (i = 1, \cdots, n).$$

3) We need to solve the following min-max problem:

$$(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) = \min_{w,\xi,\xi^*} \max_{\alpha,\alpha^*,\beta,\beta^*} L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*)$$
$$= \max_{\alpha,\alpha^*,\beta,\beta^*} \min_{w,\xi,\xi^*} L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*)$$

[The max and min can be switched because the so-called strong duality holds for quadratic problems.] Taking the derivative of L w.r.t the primal variables (w, ξ_i and ξ_i^*), we get

$$\partial_w L = w - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i = 0$$
$$\partial_{\xi_i} L = C - \alpha_i - \beta_i = 0$$
$$\partial_{\xi_i^*} = C - \alpha_i^* - \beta_i^* = 0$$

From the last two equations we have that

$$0 \leq \beta_i = C - \alpha_i$$
$$0 \leq \beta_i^* = C - \alpha_i^*$$

Substituting the results back into the Lagrangian $L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*)$, we get

$$L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\beta_i\xi_i + \beta_i^*\xi_i^*) - \sum_{i=1}^{n}\alpha_i(\epsilon + \xi_i - y_i + \langle w, xi\rangle) - \sum_{i=1}^{n}\alpha_i^*(\epsilon + \xi_i^* - y_i + \langle w, xi\rangle)$$
$$= \frac{1}{2}||\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i||^2 + \sum_{i=1}^{n}\xi_i\underbrace{(C - \beta_i - \alpha_i)}_{0} + \sum_{i=1}^{n}\underbrace{(C - \beta_i^* - \alpha_i^*)}_{0} - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*)$$
$$+ \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) + \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\underbrace{\langle w, xi\rangle}_{\langle\sum_{j=1}^{n}(\alpha_j - \alpha_j^*)x_j, x_i\rangle}$$
$$= -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle w, xi\rangle - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*).$$

Therefore, the dual problem is

$$\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*),$$
$$s.t.\ \alpha_i, \alpha_i^* \in [0, C].$$

[Note that if you use $\langle x_i, x_j\rangle + b$ instead of $\langle x_i, x_j\rangle$, then you have an extra constraint $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$.]
4) The problem has a quadratic objective with linear constraints, therefore it can be solved by a Quadratic Programming solver.

5) The KKT complementary slackness conditions are as follows. In the optimal solutions of min-max problem, we have that:

$$\begin{aligned}\alpha_i(\epsilon + \xi_i - y_i + \langle w, x_i\rangle) &= 0\\ \alpha_i^*(\epsilon + \xi_i^* + y_i - \langle w, x_i\rangle) &= 0\\ \beta_i\xi_i &= 0\\ \beta_i^*\xi_i^* &= 0\end{aligned}$$

for all $i = 1, \cdots, n$.
The first equation implies that if $\alpha_i > 0$, then $(\epsilon + \xi_i - y_i + \rangle w, x_i) = 0$.
Now, if $\xi_i = 0$, then it implies that x_i is on the border of the ϵ-tube, therefore x_i is a margin support vector. If $\xi_i > 0$, then it means we are outside of the ϵ-tube. These x_i vectors are the non-margin support vectors. Similar reasoning holds for ξ_i^* and α_i^*.
6) Since for prediction we use $f(x) = \langle w, x\rangle$, and $w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i$, therefore

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle.$$

7) Yes, we can write the above equation in the kernel form.

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(x_i, x).$$

8) ϵ plays the opposite role of C. The smaller the value of ϵ, the harder SVM tries to fit smaller errors around the learnt SVM function, and leads to a more complex model. Smaller ϵ also leads to a less sparse solution (more support vectors).
Small ϵ - More complex model. Low Bias, High variance. Large ϵ - Less complex model. High Bias, Low Variance.
9) C plays a similar role as it did during classification. It is a measure of how strongly we penalize errors. It should be tuned for bias vs variance with model selection. The higher the value of C, the larger the tendency of SVM to penalize errors and overfit the data. The lower the value of C, the larger its tendency to ignore errors and underfit the data.
Large C - More complex model. Low Bias, High variance.
Small C - Less complex model. High Bias, Low Variance.

1.5 Problem: C4.5 Decison Tree (5 pts.)

In this problem, we are set to construct a decision tree using the C4.5 algorithm. This particular algorithm leverages the Information Gain Ratio as a construction principle, guiding the formation of the decision tree.
Let's denote our dataset as D and its size as $|D|$. Suppose the dataset contains K distinct classes, labeled as C_k, where k ranges from 1 to K. The size of the individual data points allocated within each class C_k is expressed as $|C_k|$. As such, the sum of the sizes of all classes, i.e. $\sum_{k=1}^{K}|C_k| = |D|$. Given that the attribute A comprises n unique values, denoted as $\{a_1, a_2, \cdots, a_n\}$, it can bifurcate

the dataset D into n subsets $D_1, D_2, \cdots, D_n$. Each subset size is in turn represented as $|D_i|$. It follows that the aggregation of all these subset sizes, i.e. $\sum_{i=1}^{n} |D_i|$, equates to the overall size of the original dataset, that is, $|D|$. Let the set of samples in subset D_i that belong to class C_k be represented by D_{ik}, that is, $D_{ik} = D_i \cap C_k$. The count of samples in D_{ik} is represented by $|D_{ik}|$. The empirical entropy of dataset is:

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}.$$

The empirical conditional entropy of attribute A respect to dataset D is:

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}.$$

The information gain is:

$$g(D, A) = H(D) - H(D|A).$$

We can calculate the Information Gain Ratio of attribute A for the dataset D is:

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}.$$

where $H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$ and n is the number of unique values of attribute A.
We can utilize the Information Gain Ratio to build a decision tree through the C4.5 algorithm, as described in Algorithm 1. For more information about the C4.5 algorithm, you can refer to the example `https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decision-tree-example/`.

Algorithm 1 C4.5 Algorithm

Require: Training dataset D, attribute set A, threshold ϵ
Ensure: Decision tree T

1: If D belongs to the same class C_k, T is a single-node tree, and class C_k is assigned as the label of the node. Return T
2: If A is an empty set, set T as a single-node tree, and assign the label of the node as the class with the highest number of instances. Return T
3: Calculate g_R for each attribute A, select the attribute A_g with the maximum **information gain ratio**
4: If the **information gain ratio** of A_g is less than ϵ, T is a single-node tree, and the class C_k with the maximum number of instances in D is assigned as the label. Return T
5: Split A_g into several non-empty subsets D_i
6: For each subset D_i, use it as the training dataset, $A - \{A_g\}$ as the attribute set, and recursively call the previous steps to obtain T_i. Return T_i

According to the training dataset provided Table 1, generate a decision tree using the Information Gain Ratio (C4.5 algorithm).

ID	Age	Work	House	Credit	Class
1	young	No	No	Normal	No
2	young	No	No	Good	No
3	young	Yes	No	Good	Yes
4	young	Yes	Yes	Normal	Yes
5	young	No	No	Normal	No
6	middle	No	No	Normal	No
7	middle	No	No	Good	No
8	middle	Yes	Yes	Good	Yes
9	middle	No	Yes	Excellent	Yes
10	middle	No	Yes	Excellent	Yes
11	old	No	Yes	Excellent	Yes
12	old	No	Yes	Good	Yes
13	old	Yes	No	Good	Yes
14	old	Yes	No	Excellent	Yes
15	old	No	No	Normal	No

Table 1: Dataset of Loan Application

Solution 1.5

We can calculate Information Gain Ratio by $g_R(D, A) = \frac{g(D,A)}{H_A(D)}$.
Let A_1, A_2, A_3 and A_4 denote the "Age", "Work", "House" and"Credit" respectively, with D representing the entire dataset. Then we have:

$$
\begin{aligned}
&H(D) = 0.971 \\
&g(D, A_1) = 0.083,\ H_{A_1}(D) = 1.585,\ g_R(D, A_1) = 0.052 \\
&g(D, A_2) = 0.324,\ H_{A_2}(D) = 0.918,\ g_R(D, A_2) = 0.353 \\
&g(D, A_3) = 0.420,\ H_{A_3}(D) = 0.971,\ g_R(D, A_3) = 0.433 \\
&g(D, A_4) = 0.363,\ H_{A_4}(D) = 1.566,\ g_R(D, A_4) = 0.232
\end{aligned}
$$

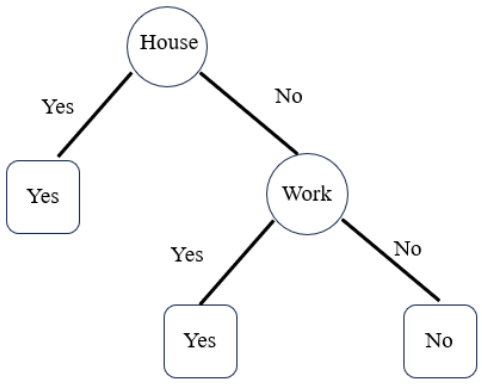
Therefore, based on the principle of information gain ratio, we choose the feature "House" as the splitting point.
In the left node, all instances with "House = yes" have the class "yes", so the left node is a leaf node with a class of "yes".
In the right node, we first organize the sub-dataset D'.

ID	Age	Work	Credit	Class
1	young	No	Normal	No
2	young	No	Good	No
3	young	Yes	Good	Yes
4	young	No	Normal	No
5	middle	No	Normal	No
6	middle	No	Good	No
7	old	Yes	Good	Yes
8	old	Yes	Excellent	Yes
9	old	No	Normal	No

Then we have:

$$
\begin{aligned}
&H(D') = 0.918 \\
&g(D', A_1) = 0.252,\ H_{A_1}(D') = 1.530,\ g_R(D', A_1) = 0.165 \\
&g(D', A_2) = 0.918,\ H_{A_2}(D') = 0.918,\ g_R(D', A_2) = 1 \\
&g(D', A_4) = 0.474,\ H_{A_4}(D') = 1.392,\ g_R(D', A_4) = 0.341
\end{aligned}
$$

So, when the feature "House" is chosen for the right node, after the split, the instance data subsets for both left and right nodes consist of a single category.
Thus, we have completed the C4.5 decision tree based on information gain ratio.



2 Coding Problems (50 pts.)

This part consists of two main tasks: Logistic Regression (25 pts.) and Support Vector Machine (25 pts.), aimed at deepening your practical understanding of these algorithma. You will find two files, "Logistic_Regression.ipynb" and "Support_Vector_Machine.ipynb." Your objective is to fill in the code in the **TASK** blank to complete the algorithm using the given dataset. Kindly note that the "weatherAUS.csv" dataset is suitable for a Logistic Regression task, while "pulsar_stars.csv" is better suited for a Support Vector Machine task.

Submission Requirement

- Your completed two jupyter notebook files, naming "StudentID_Logisitc_Regression.ipynb" and "StudentID_Support_Vector_Machine.ipynb". Your jupyter notebook files should **contain the running output** of each step (numbers, plots, etc.). If your notebook has only code but no output results, you will get a discounted score.
- Detailed report named "StudentID_homework2.pdf" on the problems in the writing part. Ensure that you include a comprehensive description of your solutions to these problems. Handwritten is also acceptable.
- Submission list: 1 pdf report for the writing part and 2 jupyter notebook for coding part.