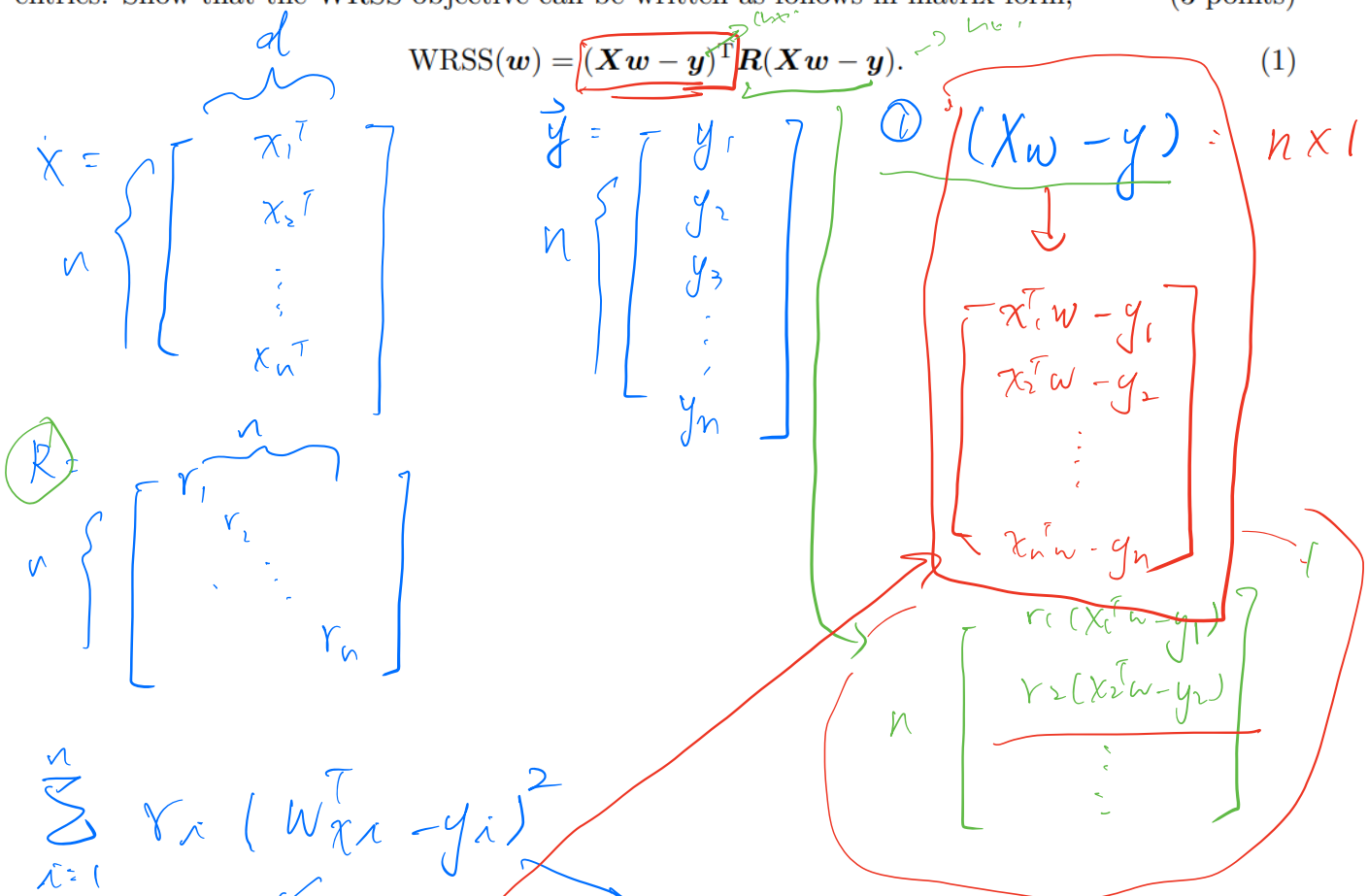
1 Weighted linear regression (10 points)

Consider a modification of the standard linear regression setup where each datapoint is associated with an importance weight. Formally, given a dataset of n datapoints $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, each datapoint is associated with an importance weight $r_i > 0$. The weighted residual sum of squares objective is defined as,

$$\text{WRSS}(\boldsymbol{w}) = \sum_{i=1}^{n} r_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right)^2.$$

(a) Let $\boldsymbol{X}$ be the $n \times d$ matrix whose i-th row is $\boldsymbol{x}_i^{\mathrm{T}}$, $\boldsymbol{y}$ be the n-dimensional column vector whose i-th entry is y_i and $\boldsymbol{R}$ be the diagonal matrix where $\boldsymbol{R}_{ii} = r_i$ for all i and 0 for all other entries. Show that the WRSS objective can be written as follows in matrix form, (3 points)

$$\text{WRSS}(\boldsymbol{w}) = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}} \boldsymbol{R} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}). \quad (1)$$



$$= \sum_{i=1}^{n} (w^T x_i - y_i) r_i (w^T x_i - y_i)$$

$$= \sum_{i=1}^{n} [(Xw - y)^T]_i [R(Xw - y)]_i$$

element-wise addition for each row of the two matrix

$\Rightarrow$ dot product

$$= (Xw - y)^T R (Xw - y)$$

(b) Solve for the closed-form solution $\boldsymbol{w}^*$ which minimizes Eq 1 (assuming invertibility of any matrices as needed). (7 points)

Hint: You might find it helpful to rewrite $WRSS(\boldsymbol{w})$ *in the form* $WRSS(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} - 2\boldsymbol{b}^T\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{R}\boldsymbol{y}$, *for some matrix* $\boldsymbol{A}$ *and some vector* $\boldsymbol{b}$ *(which you would need to find)*

$$WRSS(w) = (Xw - y)^T R (Xw - y) \qquad (A+B)^T = A^T + B^T$$

$$= w^T X^T R (Xw - y) - y^T R (Xw - y)$$

$$= w^T X^T R X w - w^T X^T R y - y^T R X w + y^T R y$$

$1\times d \quad d\times n \quad n\times n \quad n\times 1$

$$= w^T X^T R X w - 2 y^T R X w + y^T R y$$

R symmetric

$$\nabla WRSS(w) = \nabla w^T X^T R X w - 2 \nabla y^T R X w + \nabla y^T R y$$

$$= 2 X^T R X w - 2 X^T R^T y$$

$$\frac{\partial A^T x}{\partial x} = A$$

$$\frac{\partial (X^T A X)}{\partial X} = 2AX$$

why?

$$[\nabla (x^T A x)]_i$$

$$= \partial_i (x_j A_{jk} x_k) = [\partial_i x_j](A_{jk} x_k) + x_j A_{jk} \partial_i x_k$$

$$= \delta_{ik} x_k + x_j A_{ji}$$

$$w^* \Rightarrow 2 X^T R X w^* - 2 X^T R^T y = 0$$

$$\Rightarrow w^* = (X^T R X)^{-1} X^T R^T y$$

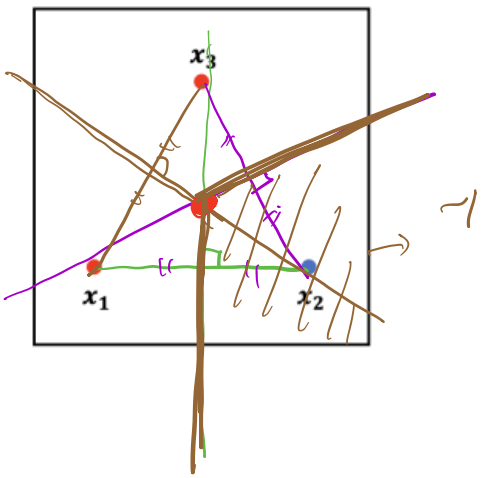
$= (A + A^T)x$

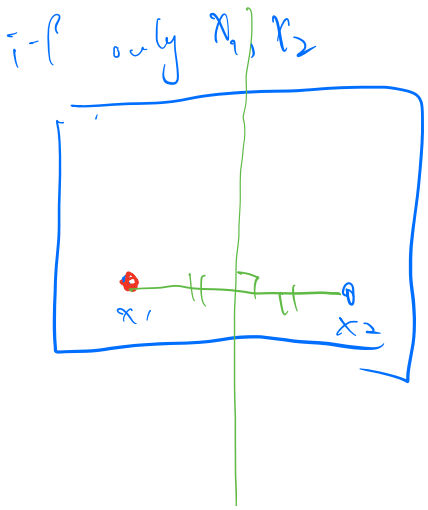
5 Short answer questions (15 points)

5.1 1-NN (3 points)

Consider the two-dimensional binary classification dataset in the figure below. There are three points with labels as follows: $\{(\boldsymbol{x}_1, +1), (\boldsymbol{x}_2, -1), (\boldsymbol{x}_3, +1)\}$ (in the figure, red corresponds to label $+1$ and blue corresponds to -1). If we train a 1-nearest neighbor model on this data using the usual Euclidean distance $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2$ to compute distances, what will be the decision boundary we get? Mark the decision bounary in the figure, and also explain your answer briefly.

Hint: You may find it helpful to first consider the case where only $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are present, and then the case where only $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are present.





Generalizatoin

key: how many function we are consider? : $|\mathcal{F}|$

Assumptions for today's theory

Finite sized function classes.

Def: A function class $\mathcal{F}$ is finite-sized if $|\mathcal{F}|$ is finite.

e.g. $\mathcal{F} = \{ f(x) = w^T x : w \in \{-1, 0, 1\}^d \}$

$|\mathcal{F}| = 3^d$ each dim 3 choice

Realizability: There exists $f^* \in \mathcal{F}$ s.t. $y = f^*(x) \; \forall \; x \in \mathcal{X}$.

Theorem: Let $\mathcal{F}$ be a function class with size $|\mathcal{F}|$. Let $y = f^*(x)$ for some $f \in \mathcal{F}$. Suppose we get a training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of size n drawn i.i.d. from the data distribution D. Let

$$f_S^{ERM} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

If $n \geq \frac{\ln(|\mathcal{F}|/\delta)}{\varepsilon}$, then with probability $(1-\delta)$ over $\{(x_i, y_i), i \in [n]\}$, $R(f_S^{ERM}) \leq \varepsilon$ (for constants ε, δ).

Rule of thumb for generalization

Suppose the functions f in our function class $\mathcal{F}$ have d parameters which can be set. Assume we discretize these parameters so they can take W possible values each. How much data do we need to have small generalization gap?

$|\mathcal{F}| = W^d$

$\therefore$ generalization gap is at most ε with $n \geq \frac{\ln(|\mathcal{F}|/\delta)}{\varepsilon}$ samples

$= \frac{d \ln(W/\delta)}{\varepsilon}$ samples

This ε maybe ε^2 without realizability

A useful rule of thumb: to guarantee generalization, make sure that your training data set size n is at least linear in the number d of free parameters in the function that you're trying to learn.

5.2 Generalization (4 points)

Recall the generalization theorem that we proved in class:

Theorem 1. *Let $\mathcal{F}$ be a function class with size $|\mathcal{F}|$. Let $y = f^*(\boldsymbol{x})$ for all inputs $\boldsymbol{x}$, for some $f^* \in \mathcal{F}$. Suppose we get a training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ of size n with each datapoint drawn i.i.d. from the data distribution D. Let*

$$f_S^{ERM} = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i),$$

where $\ell(f(\boldsymbol{x}_i), y_i)$ denotes the 0-1 loss. For any constants $\epsilon, \delta \in (0,1)$, if $n \geq \frac{\ln(|\mathcal{F}|/\delta)}{\epsilon}$, then with probability $(1-\delta)$ over $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, $R(f_S^{ERM}) < \epsilon$.

$x_1 = [0, 0, 0, 1, 0, 1, \ldots]$ (d dim)

Consider the case where the input $\boldsymbol{x}$ is binary valued, i.e. $\boldsymbol{x} \in \{0,1\}^d$. Define the class of *conjunctions on two variables* as follows, $\mathcal{F} = \{f_{i,j}, 1 \leq i,j \leq d\}$ where $f_{i,j}(\boldsymbol{x})$ is defined as the AND function on the i-th and j-th coordinates of input $\boldsymbol{x}$:

$$f_{i,j}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}(i) = 1 \text{ AND } \boldsymbol{x}(j) = 1 \\ -1 & \text{otherwise.} \end{cases}$$

Here $\boldsymbol{x}(i)$ refers to the i-th coordinate of input $\boldsymbol{x}$.

(a) Suppose we are given a dataset S of n datapoints drawn i.i.d. from some distribution where the labels are given by some conjunction f^* on two variables. If we use empirical risk minimization to learn a conjunction on two variables f_S^{ERM} to fit the data, how large does n have to be to ensure that the predictor we learn has expected classification error $R(f_S^{ERM})$ at most 5% with probability at least 90% over the randomness in the n datapoints?

$|\mathcal{F}| = \binom{d}{2} \approx \left(\frac{d}{2}\right)^2$

$\delta = 0.1$

$\epsilon = 0.05$

(b) Now, consider the class of conjunctions defined on k variables where the predictor $f_{i_1,i_2,\ldots,i_k}(\boldsymbol{x})$ has k parameters $i_1, \ldots, i_k$ and is defined as follows:

$$f_{i_1,i_2,\ldots,i_k}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}(i_1) = 1 \text{ AND } \boldsymbol{x}(i_2) = 1 \text{ AND } \ldots \boldsymbol{x}(i_k) = 1 \\ -1 & \text{otherwise.} \end{cases}$$

Roughly, how much training data is needed to learn a conjunction over k variables which generalizes well to test data?

$|\mathcal{F}| = \binom{d}{k} \approx d^k$

6.3 Generalization (4 points)

Based on your previous successes, you are now working with a team of computation biologists who are trying to build a model to predict a phenotype of an individual, such as their risk of diabetes, from their gene sequence. A dataset of the gene sequence of n individuals has been compiled for this purpose. For each individual in the dataset, you have their DNA sequenced at $d = 10^6$ locations. The biologists in your team feel that mutations at certain *pairs* of DNA locations should together be responsible for the phenotype, hence you want to experiment with a linear classifier with the genes at pairs of DNA locations as the features.[1]

(a) You want to ensure that when the linear classifier you train is applied to other individuals from the same populations, it should get an error rate within some small constant of the error rate it obtained on your training data. Roughly how large should the training data be to ensure this is the case? (only a rough, order of magnitude answer with a brief explanation is needed)

Hint: What is the dimensionality of the input to the linear classifier?

$x \in \{a, t, c, g\}^{10^6}$ $x_1 = [a, t, c, g, \ldots]$ (10^6)

$f = [0, 0, 0, 0, 1, \ldots 1, 0, 0]$

$|F| \approx \binom{10^6}{2}$ in magnitude of 10^{12}

(b) As sequencing these many individuals is too expensive, how can you still train your model while ensuring that it does not overfit? If you know that the ground truth depends on only $k \approx 10^3$ gene pairs, then roughly how many samples do you need to learn a good model? (only a rough, order of magnitude answer with a brief explanation is needed)

→ l_1-regularization term to increase sparsity.

$|F| \propto 10^3$

3 Perceptron algorithm (10 points)

We will investigate the perceptron algorithm in this question (the algorithm is reproduced in Algorithm 2). The perceptron algorithm gets access to a dataset of n instances $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. It outputs a linear classifier $y = \text{SIGN}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$. Assume $\boldsymbol{x}_i \neq \mathbf{0}\ \forall\ i \in \{1, \ldots, n\}$.

Algorithm 2: Perceptron

Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, number of iterations T
Initialize $\boldsymbol{w}_{(0)} \leftarrow \mathbf{0}$;
for t *in* $\{0, \ldots, T-1\}$ **do**
  Pick a data point $(\boldsymbol{x}_i, y_i)$ randomly
  Make a prediction $\hat{y} = \text{SIGN}(\boldsymbol{w}_{(t)}{}^T \boldsymbol{x}_i)$ using $\boldsymbol{w}_{(t)}$
  if $\hat{y} \neq y_i$ **then**
    $\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)} + y_i \boldsymbol{x}_i$
  else
    $\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)}$

As the algorithm proceeds, suppose the same weight vector is seen twice, despite at least one update in between. In particular, suppose there is some j and k where $j < k$ such that $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$, and there is at least one ℓ where $j < \ell < k$ such that $\boldsymbol{w}_{(j)} \neq \boldsymbol{w}_{(\ell)}$. We will show that if this happens, then the given dataset is not linearly separable. To prove this, follow the following two steps.

(a) Write down an expression which relates $\boldsymbol{w}_{(j)}$ and $\boldsymbol{w}_{(k)}$. Your expression will involve the datapoints observed in the intermediate iterations. (5 points).

$w_{(j)} \rightarrow \ldots\ldots \rightarrow w_{(\ell)} \rightarrow \ldots w_{(k)}$

$$w_{(\ell)} = w_{(j)} + \sum_{i=j}^{\ell-1} \mathbb{I}\left(w_{(i)}^T x_i y_i < 0\right) y_i x_i .$$

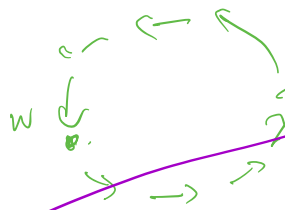
x_i is the instance being selected in iter. i

(b) Now suppose there is some linear classifier $\boldsymbol{w}^*$ which classifiers all datapoints perfectly, i.e. $y_i = \text{SIGN}(\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{x}_i)$ for all $i \in \{1, \ldots, n\}$. Use your expression from the previous part and the fact that $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$ to arrive at some contradiction, hence proving that the dataset cannot be linearly separable if $\boldsymbol{w}_{(j)} = \boldsymbol{w}_{(k)}$. (5 points).

$w_{(\ell)} - w_{(j)}$

$\hookrightarrow = w_{(j)} + \sum \ldots\ldots$

$$\Rightarrow \sum_{i=j}^{\ell-1} \mathbb{I}\left(w_{(i)}^T x_i y_i < 0\right) y_i x_i \Rightarrow 0$$



$x_j \ldots\ldots x_{e-1}$.

selece this being really used e updates.

$$\Rightarrow \begin{cases} \text{for } y_i = 1 \Rightarrow x_1, x_2, \ldots x_m \\ \text{for } y_i = -1 \Rightarrow x_{m+1}, x_{m+2} \ldots x_n \end{cases} \quad (n > m > 1)$$

$x_1 + x_2 + x_3 + \ldots + x_m = x_{m+1} + x_{m+2} + \ldots x_n$

Notice that w^* exist.

for $y_i = 1 \Leftrightarrow w^* x_i \geq 0$

$\Rightarrow$ $y_i = 1$

w^*

$w^*(x_1 + x_2 + \ldots x_m) = w^*(x_{m+1} + \ldots + x_n)$

$y_i = -1$?

≥ 0 $\quad$ < 0

$\not\Rightarrow w^*$

(2) Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a data matrix with each row corresponding to the features of an example and $\boldsymbol{y} \in \mathbb{R}^n$ be a vector of all the outcomes. Which of the following statements are true about linear regression (minimizing $F(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$)?

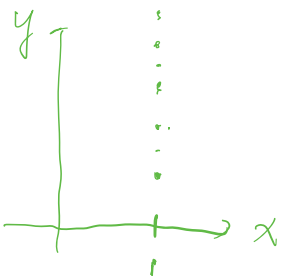
(A) If n is much larger than d, then the least squares solution $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ should have small gap between training and test accuracy.

(B) When $d = 1$ and $\boldsymbol{X}$ is the all ones vector ($\boldsymbol{X} = [1, 1, \ldots, 1]^{\mathrm{T}}$), the optimal value w^* for w is the average of the outcomes $\boldsymbol{y}$, i.e. $w^* = (1/n)\sum_{i=1}^n y_i$.

(C) When $n < d$, the problem of minimizing $F(\boldsymbol{w})$ is non-convex.

(D) If the step size is too large, then gradient descent on $F(\boldsymbol{w})$ may never converge.

Answer: ABD. A is true since the generalization gap goes down if we have a lot of data (as also seen for this problem in HW1). C is not true since the problem is always convex. D is true, as also seen in HW1.



$F(w) = \|w - y\|_2^2$

$\nabla F(w) = 0 \Rightarrow w = \text{mean of } y.$

(3) Which of the following statements are true about k-nearest neighbors (k-NN)?

(A) k-NN always gives a linear decision boundary.

(B) 50-NN is more likely to overfit the data compared to 1-NN.

(C) Using different distance metrics (such as ℓ_2 distance, ℓ_1 distance, cosine similarity etc.) never affects the decision boundary of 1-NN.

(D) k-NN is similar to kernel methods in the sense that both of them may require us to store the entire training data to make predictions on the test set.

Answer: D. B is false since 50-NN considers 50 neighbors and is more robust to overfitting.

(A) what distance is used?

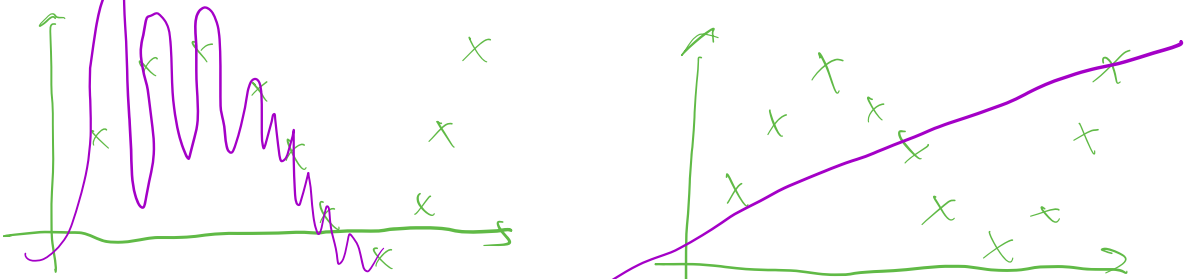
(B) Less likely

(D) non-parametric method.

(9) Which of the following statements are true about bias and variance of ML models?

(A) Bias and variance are terms used in the context of underfitting and overfitting.
(B) If a model has large variance, then its performance will not improve even if we add a lot of training data.
(C) If a model has large bias, then its performance will not improve even if we add a lot of training data.
(D) Choosing a complicated, non-linear mapping on the input features can increase the bias of a logistic regression model trained on the data, but will reduce its variance.

Answer: AC. B is incorrect because high variance means the model complexity is large, and adding more training data increases the performance. C is correct because large bias means the model complexity is small, and adding more training data does not improve the performance. D is incorrect because a complicated mapping increases the variance and reduces the bias.



(10) Which of the following are **NOT** valid kernel functions?

(A) $k(x, x') = \cos(x - x') = \cos(x)\cos(x') + \sin(x)\sin(x')$ (defined on univariate inputs x, x').
(B) $k(x, x') = (x - x')^2$ (defined on univariate inputs x, x').
(C) $k(x, x') = \mathbf{1}(|x - x'| \geq 1)$ (defined on univariate inputs x, x', $\mathbf{1}(\cdot)$ is the indicator function which is 1 if the input is true, 0 otherwise).
(D) $k(x, x') = \ln(xx')$ (defined on univariate inputs $x, x' > 0$).

Answer: BCD. A is a kernel as the sum of two kernels. BC are not kernels, using $x = 0, x' = 1$ to construct non-PSD Gram matrices. D is not a kernel, using $x = 1, x' = 2$ to construct a non-PSD Gram matrix.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = -2 \leq 0$$

PSD $\Rightarrow x^T A x \geq 0$

use $x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$