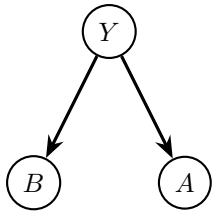
CSCD84: Artificial Intelligence

Worksheet Solution: Naive Bayes, Perceptron, and Logistic Regression

Q1. **(Naive Bayes)** In this question, we will train a Naive Bayes classifier to predict class labels Y as a function of input features A and B. Y , A, and B are all binary variables, with domains 0 and 1. We are given 10 training points from which we will estimate our distribution.

A	1	1	1	1	0	1	0	1	1	1
B	1	0	0	1	1	1	1	0	1	1
Y	1	1	0	0	0	1	1	0	0	0



1.a. What are the maximum likelihood estimates for the tables $\mathbb{P}(Y)$, $\mathbb{P}(A \mid Y)$, and $\mathbb{P}(B \mid Y)$?

Answer. Relative frequencies are the maximum likelihood estimates. Hence,

$$\begin{aligned}
\mathbb{P}(Y = 0) &= \frac{6}{10}, \\
\mathbb{P}(B = 0 \mid Y = 0) &= \frac{2}{6}, \\
\mathbb{P}(B = 0 \mid Y = 1) &= \frac{1}{4}, \\
\mathbb{P}(A = 0 \mid Y = 0) &= \frac{1}{6}, \\
\mathbb{P}(A = 0 \mid Y = 1) &= \frac{1}{4}.
\end{aligned}$$

1.b. Consider a new data point $(A = 1, B = 1)$. What label would this classifier assign to this sample?

Answer. $\mathbb{P}(Y \mid A = 1, B = 1) = \alpha \mathbb{P}(Y, A = 1, B = 1) = \alpha \mathbb{P}(Y)\mathbb{P}(B = 1 \mid Y)\mathbb{P}(A = 1 \mid Y) = \alpha \begin{bmatrix} 1/3 \\ 9/40 \end{bmatrix}$. Hence, the classifier will label this point as 0.

1.c. Compute the new distribution for $\mathbb{P}(A \mid Y)$ given Laplace Smoothing with $k = 2$.

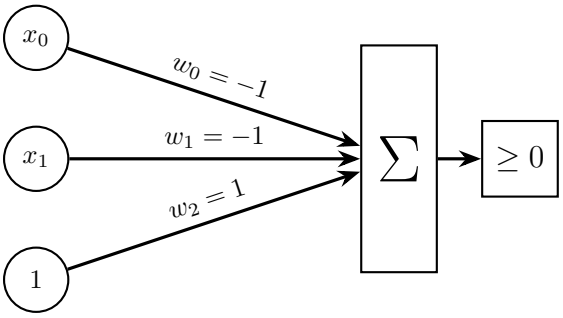
Answer.

$$\begin{aligned}
\mathbb{P}(A = 0 \mid Y = 0) &= \frac{1 + 2}{6 + |A| \times 2} = \frac{3}{10}, \\
\mathbb{P}(A = 0 \mid Y = 1) &= \frac{1 + 2}{4 + |A| \times 2} = \frac{3}{8}.
\end{aligned}$$

Q2. **(Perceptrons)** For all parts of this question perceptrons should output 1 if $w_n + \sum_{i=0}^{n-1} w_i x_i \geq 0$ and 0 otherwise. The weight w_n is called the bias weight.

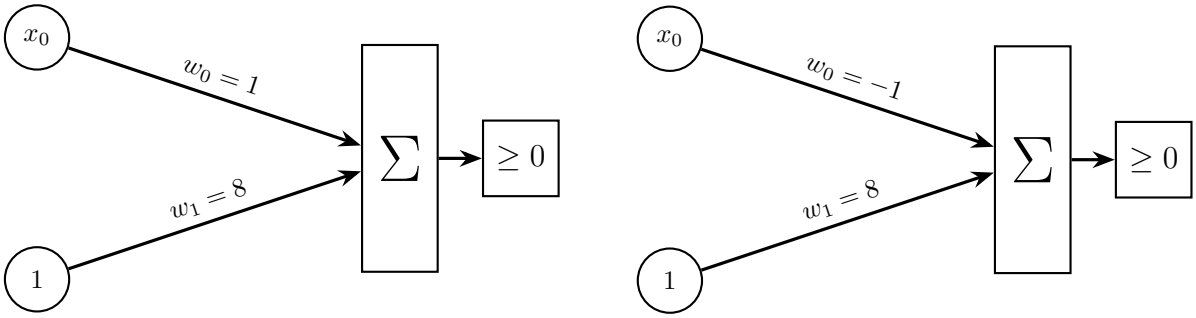
2.a. Assuming there will be two inputs x_0 and x_1, each with possible values in $\{0, 1\}$, give values for a triple of weights (w_0, w_1, w_2) such that the corresponding perceptron would act as a NAND gate for the two inputs. (Weight w_2 is the bias weight.)
Note that a NAND gate outputs 1 when at least one of the inputs is 0; it outputs 0 otherwise.

Answer. $(w_0, w_1, w_2) = (-1, -1, 1)$:



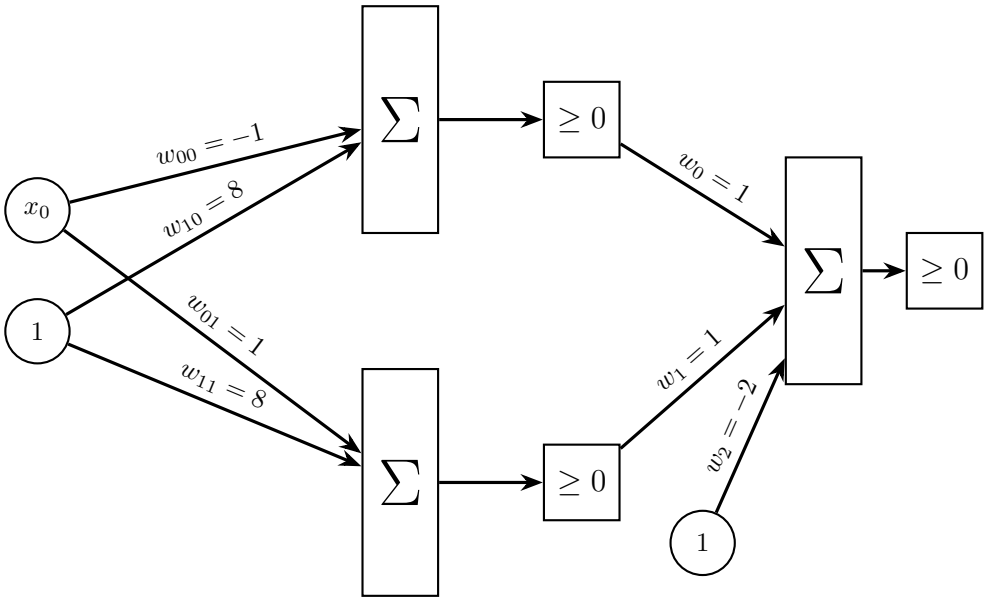
2.b. Draw a perceptron, with weights, that accepts a single integer x and outputs 1 if and only if the input is greater than or equal to -8. Be sure to include the bias input of value 1 and its weight in your diagram. Draw another perceptron that outputs 1 if and only if the input is less than or equal to 8.

Answer.



2.c. Using the previous perceptrons, create a two-layer perceptron that outputs 1 if $|x| \leq 8$, and 0 otherwise.

Answer.



2.d. Suppose we want to train a perceptron to compare two numbers x_0 and x_1 and produce output $y = 1$ provided that x_1 exceeds x_0 by at least 5. Assume that the initial weight vector is: $(w_0, w_1, w_2) =$

$(-1, -1, 1)$. Consider a first training example: $((x_0, x_1), y) = ((3, 8), 1)$. This says that with inputs 3, and 8, the output y should be 1, since 8 exceeds 3 by 5. What will be the new values of the weights after this training example has been processed one time? Assume the learning rate is 2.

Answer. The point $(x_0 = 3, x_1 = 8, x_2 = 1)$ is a miss-classified point with true label 1. Hence, the weight update would be as follows.

$$(w_0, w_1, w_2) \leftarrow (w_0, w_1, w_2) + (3, 8, 1).$$

Therefore, the new weights would be $(2, 7, 2)$.

2.e. Continuing with the last example, now suppose that the next step of training involves a different training example: $((2, 5), 0)$. The output for this example should be 0, since 5 does not exceed 2 by at least 5. Starting with the weights already learned in the first step, determine what the adjusted weights should be after this new example has also been processed once.

Answer. With weights $(2, 7, 2)$, the point $(x_0 = 2, x_1 = 5, x_2 = 1)$ is a miss-classified point with true label 0. Hence, the weight update would be as follows.

$$(w_0, w_1, w_2) \leftarrow (w_0, w_1, w_2) - (2, 5, 1).$$

Therefore, the new weights would be $(0, 2, 1)$.

Q3. **(Multiclass Perceptrons)** Consider the problem of classifying text items into three topic areas: AI, Medicine, and Hiking. We have four training examples:

$\mathbf{e_1}$: Hill climbing informs gradient descent. (AI)

$\mathbf{e_2}$: Alzheimers is a gradual descent. (Medicine)

$\mathbf{e_3}$: Hill climbing helps. (Medicine)

$\mathbf{e_4}$: Gorp helps hill climbing. (Hiking)

3.a. Convert the four training examples into the (vector, category) form indicated by the table. (You'll use the reference vocabulary given in the table.)

$\mathbf{e_i}$	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill	(category)
$\mathbf{e_1}$									
$\mathbf{e_2}$									
$\mathbf{e_3}$									
$\mathbf{e_4}$									

Answer.

$\mathbf{e_i}$	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill	(category)
$\mathbf{e_1}$	0	1	1	0	1	0	0	1	AI
$\mathbf{e_2}$	1	0	1	0	0	1	0	0	Medicine
$\mathbf{e_3}$	0	1	0	0	0	0	1	1	Medicine
$\mathbf{e_4}$	0	1	0	1	0	0	1	1	Hiking

3.b. We will start with a weight vector for each of the 3 categories ($\mathbf{W}_\text{A}$ for AI, $\mathbf{W}_\text{M}$ for Medicine, and $\mathbf{W}_\text{H}$ for Hiking), with all weights 0 except 1 for the bias on the AI vector.

$\mathbf{W}$	bias	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill
$\mathbf{W}_\text{A}$	1	0	0	0	0	0	0	0	0
$\mathbf{W}_\text{M}$	0	0	0	0	0	0	0	0	0
$\mathbf{W}_\text{H}$	0	0	0	0	0	0	0	0	0

Perform one epoch of training showing the resulting changed vectors whenever there is a change made to a weight vector. When each training example is processed, if any weight vector does not change, do not rewrite that vector. Whenever a vector is updated, write the new vector on its own line below, clearly indicating which category it belongs to. For example, $W_H = \ldots$

Answer.

Step 1 (datapoint $\mathbf{e}_1$): $\mathbf{e}_1.\mathbf{W}_\text{A} = 1$, $\mathbf{e}_1.\mathbf{W}_\text{M} = 0$, $\mathbf{e}_1.\mathbf{W}_\text{H} = 0$. Hence, it would be correctly classified and the wights won't be updated.

Step 2 (datapoint $\mathbf{e}_2$): $\mathbf{e}_2.\mathbf{W}_\text{A} = 1$, $\mathbf{e}_2.\mathbf{W}_\text{M} = 0$, $\mathbf{e}_2.\mathbf{W}_\text{H} = 0$. Hence, it is missclassified as AI instead of Medicine. Therefore,

$$\mathbf{W}_\text{A} \leftarrow \mathbf{W}_\text{A} - \mathbf{e}_2,$$
$$\mathbf{W}_\text{M} \leftarrow \mathbf{W}_\text{M} + \mathbf{e}_2.$$

Thus, the updated weights would be as follows.

$\mathbf{W}$	bias	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill
$\mathbf{W}_\text{A}$	0	-1	0	-1	0	0	-1	0	0
$\mathbf{W}_\text{M}$	1	1	0	1	0	0	1	0	0
$\mathbf{W}_\text{H}$	0	0	0	0	0	0	0	0	0

Step 3 (datapoint $\mathbf{e_3}$): $\mathbf{e}_3.\mathbf{W}_\text{A} = 0$, $\mathbf{e}_3.\mathbf{W}_\text{M} = 1$, $\mathbf{e}_3.\mathbf{W}_\text{H} = 0$. Hence, it would be correctly classified and the wights won't be updated.

Step 4 (datapoint $\mathbf{e_4}$): $\mathbf{e}_4.\mathbf{W}_\text{A} = 0$, $\mathbf{e}_4.\mathbf{W}_\text{M} = 4$, $\mathbf{e}_4.\mathbf{W}_\text{H} = 0$. Hence, it is missclassified as Medicine. Therefore,

$$\mathbf{W}_\text{M} \leftarrow \mathbf{W}_\text{M} - \mathbf{e}_4,$$
$$\mathbf{W}_\text{H} \leftarrow \mathbf{W}_\text{H} + \mathbf{e}_4.$$

Thus, the updated weights would be as follows.

$\mathbf{W}$	bias	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill
$\mathbf{W}_\text{A}$	0	-1	0	-1	0	0	-1	0	0
$\mathbf{W}_\text{M}$	0	1	-1	1	-1	0	1	-1	-1
$\mathbf{W}_\text{H}$	1	0	1	0	1	0	0	1	1

3.c. Training could take many epochs to converge. See if you can skip the training and manually provide a set of three weight vectors that will correctly handle the training examples.

Answer.

$\mathbf{W}$	bias	Alzheimers	climbing	descent	gorp	gradient	gradual	helps	hill
$\mathbf{W}_\text{A}$	0	0	0	0	0	2	0	0	0
$\mathbf{W}_\text{M}$	0	1	1	0	0	0	0	0	0
$\mathbf{W}_\text{H}$	0	0	0	0	2	0	0	0	0

Q4. **(Hinge Loss)** Here are two reviews of Perfect Blue, from Rotten Tomatoes:

Rotten Tomatoes has classified these reviews as "positive" and "negative," respectively, as indicated by the intact tomato on the top and the splatter on the bottom.

In this assignment, you will create a simple text classification system that can perform this task automatically. We'll warm up with the following set of four mini-reviews, each labeled positive $(+1)$ or negative (-1):

1. x_1 : not good; label: (-1)
2. x_2 : pretty bad; label: (-1)
3. x_3 : good plot; label: $(+1)$
4. x_4 : pretty scenery; label: $(+1)$

Each review x is mapped onto a feature vector $\phi(x)$, which maps each word to the number of occurrences of that word in the review. For example, the second review maps to the (sparse) feature vector $\phi(x) = \{pretty : 1, bad : 1\}$. The hinge loss is defined as

$$L_{\text{hinge}}(x, y, \mathbf{w}) = \max\{0, 1 - \mathbf{w}.\phi(x)y\},$$

where x is the review text, y is the correct label, $\mathbf{w}$ is the weight vector.

4.a. **(Gradient Descent)** Suppose we run stochastic gradient descent once for each of the 4 samples in the order given above, updating the weights according to

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L_{\text{hinge}}(x, y, \mathbf{w}).$$

After the updates, what are the weights of the six words ("pretty", "good", "bad", "plot", "not", "scenery") that appear in the above reviews?

- Use $\eta = 0.1$ as the step size.
- Initialize $\mathbf{w} = [0, 0, 0, 0, 0, 0]$.
- The gradient $\nabla_{\mathbf{w}} L_{\text{hinge}}(x, y, \mathbf{w}) = 0$ when margin is exactly 1 (*i.e.*, when $\mathbf{w}.\phi(x)y = 1$).

Present your answer as a weight vector that contains a numerical value for each of the tokens in the reviews ("pretty", "good", "bad", "plot", "not", "scenery").

Answer. First, let's formulate the gradient of the loss function.

$$\nabla_{\mathbf{w}} L_{\text{hinge}}(x, y, \mathbf{w}) = \begin{cases} -\phi(x)y, & \text{if } \mathbf{w}.\phi(x)y < 1, \\ \vec{\mathbf{0}}, & \text{if } \mathbf{w}.\phi(x)y \geq 1, \end{cases}$$

where $\vec{\mathbf{0}}$ is a vector of all zeros.

Step 1 (datapoint x_1): $\nabla_{\mathbf{w}} L_{\text{hinge}}(x_1, y_1, \mathbf{w}) = -\phi(x_1)y_1$. Therefore,

$$\mathbf{w} \leftarrow \mathbf{w} - 0.1(-\phi(x_1)y_1).$$

Hence, $\mathbf{w} = [0, -0.1, 0, 0, -0.1, 0]$.

Step 2 (datapoint x_2): $\nabla_{\mathbf{w}} L_{\text{hinge}}(x_2, y_2, \mathbf{w}) = -\phi(x_2)y_2$.

$$\mathbf{w} \leftarrow \mathbf{w} - 0.1(-\phi(x_2)y_2).$$

Hence, $\mathbf{w} = [-0.1, -0.1, -0.1, 0, -0.1, 0]$.

Step 3 (datapoint x_3): $\nabla_{\mathbf{w}} L_{\text{hinge}}(x_3, y_3, \mathbf{w}) = -\phi(x_3)y_3$.

$$\mathbf{w} \leftarrow \mathbf{w} - 0.1(-\phi(x_3)y_3).$$

Hence, $\mathbf{w} = [-0.1, 0, -0.1, 0.1, -0.1, 0]$.

Step 4 (datapoint x_4): $\nabla_{\mathbf{w}} L_{\text{hinge}}(x_4, y_4, \mathbf{w}) = -\phi(x_4)y_4$.

$$\mathbf{w} \leftarrow \mathbf{w} - 0.1(-\phi(x_4)y_4).$$

Hence, $\mathbf{w} = [0, 0, -0.1, 0.1, -0.1, 0.1]$.

4.b. (Linearly Inseparable) Given the following dataset of reviews:

1. x_1 : bad; label: (-1)
2. x_2 : good; label: $(+1)$
3. x_3 : not bad; label: $(+1)$
4. x_4 : not good; label: (-1)

Prove that no linear classifier using word features (*i.e.*, word count) can get zero error on this dataset. Remember that this is a question about classifiers, not optimization algorithms; your proof should be true for any linear classifier of the form $f_{\mathsf{w}}(x) = \text{sign}(\mathbf{w}.\phi(x))$, regardless of how the weights are learned. Propose a single additional feature for your dataset that we could augment the feature vector with that would fix this problem.

Answer. Assume by contradiction that there exists a weight vector $\hat{\mathbf{w}}$ that gets zero error on this dataset, *i.e.*,

$$\begin{aligned}
\hat{\mathbf{w}}.\phi(x_1) &< 0,\\
\hat{\mathbf{w}}.\phi(x_2) &> 0,\\
\hat{\mathbf{w}}.\phi(x_3) &> 0,\\
\hat{\mathbf{w}}.\phi(x_4) &< 0.
\end{aligned}$$

Let $x_5 =$ not. It is easy to see that $\phi(x_3) = \phi(x_5) + \phi(x_1)$ and $\phi(x_4) = \phi(x_5) + \phi(x_2)$. Thus,

$$\begin{aligned}
\hat{\mathbf{w}}.\phi(x_1) &< 0,\\
\hat{\mathbf{w}}.\phi(x_2) &> 0,\\
\hat{\mathbf{w}}.(\phi(x_5) + \phi(x_1)) &> 0,\\
\hat{\mathbf{w}}.(\phi(x_5) + \phi(x_2)) &< 0.
\end{aligned}$$

Putting the above four inequalities together, we get $0 < \hat{\mathbf{w}}.\phi(x_2) < -\hat{\mathbf{w}}.\phi(x_3) < \hat{\mathbf{w}}.\phi(x_1) < 0$, which is a contradiction. That concludes the proof.

To make the datapoints linearly separable, it suffices to introduce the new feature which is the multiple of "good" count and "no" count. After augmenting with this feature, the feature vectors will be as follows.

$\phi(x_i)$	good	bad	not	not × good	y_i
$\phi(x_1)$	0	1	0	0	-1
$\phi(x_2)$	1	0	0	0	+1
$\phi(x_3)$	0	1	1	0	+1
$\phi(x_4)$	1	0	1	1	-1

It is easy to check that the weight vector $\mathbf{w} = [+1, -1, +2, -4]$ has a loss zero on this augmented dataset.

Q5. **(Squared Loss)** Suppose that we are now interested in predicting a numeric rating for movie reviews. We will use a non-linear predictor that takes a movie review x and returns $\sigma(\mathbf{w}.\phi(x))$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function that squashes a real number to the range $(0, 1)$. For this problem, assume that the movie rating y is a real-valued variable in the range $[0, 1]$.

5.a. Suppose that we wish to use squared loss. Write out the expression for $L(x, y, \mathbf{w})$ for a single datapoint (x, y).

Answer.

$$L(x, y, \mathbf{w}) = \frac{1}{2}\left(y - \frac{1}{1 + e^{-\mathbf{w}.\phi(x)}}\right)^2.$$

5.b. Given $L(x, y, \mathbf{w})$ from the previous part, compute the gradient of the loss with respect to $\mathbf{w}$, $\nabla_{\mathbf{w}} L(x, y, \mathbf{w})$. Write the answer in terms of the predicted value $p = \sigma(\mathbf{w}.\phi(x))$.

Answer.

$$\begin{aligned}
\nabla_{\mathbf{w}} L(x, y, \mathbf{w}) &= \frac{1}{2}\nabla_{\mathbf{w}}\left(y - \frac{1}{1 + e^{-\mathbf{w}.\phi(x)}}\right)^2 = \left(y - \frac{1}{1 + e^{-\mathbf{w}.\phi(x)}}\right)\nabla_{\mathbf{w}}\left(y - \frac{1}{1 + e^{-\mathbf{w}.\phi(x)}}\right) \\
&= -(y - p)\nabla_{\mathbf{w}}\left(\frac{1}{1 + e^{-\mathbf{w}.\phi(x)}}\right) = \frac{y - p}{\left(1 + e^{-\mathbf{w}.\phi(x)}\right)^2}\nabla_{\mathbf{w}}\left(1 + e^{-\mathbf{w}.\phi(x)}\right) \\
&= \frac{y - p}{\left(1 + e^{-\mathbf{w}.\phi(x)}\right)^2}\nabla_{\mathbf{w}}\left(e^{-\mathbf{w}.\phi(x)}\right) \\
&= \frac{y - p}{\left(1 + e^{-\mathbf{w}.\phi(x)}\right)^2} e^{-\mathbf{w}.\phi(x)}\nabla_{\mathbf{w}}\left(-\mathbf{w}.\phi(x)\right) \\
&= -\frac{y - p}{\left(1 + e^{-\mathbf{w}.\phi(x)}\right)^2} e^{-\mathbf{w}.\phi(x)}\phi(x) \\
&= -\frac{y - p}{1 + e^{-\mathbf{w}.\phi(x)}}\frac{e^{-\mathbf{w}.\phi(x)}}{1 + e^{-\mathbf{w}.\phi(x)}}\phi(x) \\
&= (p - y)p(1 - p)\phi(x)
\end{aligned}$$

5.c. Suppose there is one datapoint (x, y) with some arbitrary non-zero $\phi(x)$ and $y = 1$. Specify conditions for $\mathbf{w}$ to make the magnitude of the gradient of the loss with respect to $\mathbf{w}$ arbitrarily small (*i.e.*, minimize the magnitude of the gradient). Can the magnitude of the gradient with respect to $\mathbf{w}$ ever be exactly zero? You are allowed to make the magnitude of $\mathbf{w}$ arbitrarily large but not infinity.

Why does it matter? the reason why we're interested in the magnitude of the gradients is because it governs how far gradient descent will step. For example, if the gradient is close to zero when $\mathbf{w}$ is very far from the optimum, then it could take a long time for gradient descent to reach the optimum (if at all). This is known as the vanishing gradient problem when training neural networks.

Answer. Let $y = 1$. Then the gradient would be $\nabla_{\mathbf{w}} L(x, y, \mathbf{w}) = -p(1 - p)^2\phi(x)$. To make the gradient arbitrary small, either $p \to 0$ or $p \to 1$, *i.e.*, $e^{-\mathbf{w}.\phi(x)} \to +\infty$ or $e^{-\mathbf{w}.\phi(x)} \to 0$. Hence, we must have $\mathbf{w}.\phi(x) \to -\infty$ or $\mathbf{w}.\phi(x) \to +\infty$. This can be achieved for any arbitrary non-zero $\phi(x)$, by making the magnitude of the elements in $\mathbf{w}$ arbitrary large.

The gradient cannot be exactly zero for any arbitrary $\phi(x)$ since $0 < p < 1$.