

Q1. Kernel Perceptron

Remember that the perceptron update rule looks like:

$$w \leftarrow w + yx$$

for input feature vectors x and class labels $y \in \{-1, 1\}$. If $w \cdot x = 0$, we predict positive label.

- (a) Suppose $w = [1, 1]$ initially, and we observe the following training examples:

x_0	x_1	y
1	2	-1
3	1	1
1	1	-1
1	0	1

What is the final value of w ? **[1, -3]**

- (b) Notice that because of the update rule, the final weight vector w^* is just a linear combination of training examples and the initializer. Suppose we iterate over the training set following the order in the table until all the training samples are classified correctly, fill in the coefficients below:

$$w = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \underline{-2} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \underline{1} \cdot \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \underline{-1} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \underline{0} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This means that instead of *explicitly* representing w as a vector of feature weights, we can *implicitly* represent the decision rule with a vector v with one weight per example.

- (c) Now suppose x and w are D -dimensional, and we have N training examples. How many numbers do I need to represent w *explicitly*?

D because w is D -dimensional.

- (d) How many numbers do I need to represent w *implicitly*? (Assume that the initial value for w is public information so you do not need to consume any memory to store it.)

N because we need one coefficient for each training sample.

- (e) Write the update rule for the implicit representation if you pick the i^{th} training sample (use e_i to represent a vector whose i^{th} position is 1 and all the other positions are 0s):

$$v \leftarrow v + ye_i$$

- (f) Write the prediction rule for the implicit representation if the initial w is a zero vector. You can use v_i to represent the i^{th} value from v and x_i to represent the i^{th} training sample:

$$\text{pred}(x) = \sum_i v_i \langle x_i, x \rangle$$

- (g) When is it more space-efficient to use the implicit representation? (Your answer should be at most three words/symbols, and be expressed in terms of D and N .)

$N < D$

(h) Now suppose x is two-dimensional, and we introduce a feature transformation

$$f(x) = [x_0^2, x_1^2, \sqrt{2}x_0, \sqrt{2}x_1, \sqrt{2}x_0x_1, 1]$$

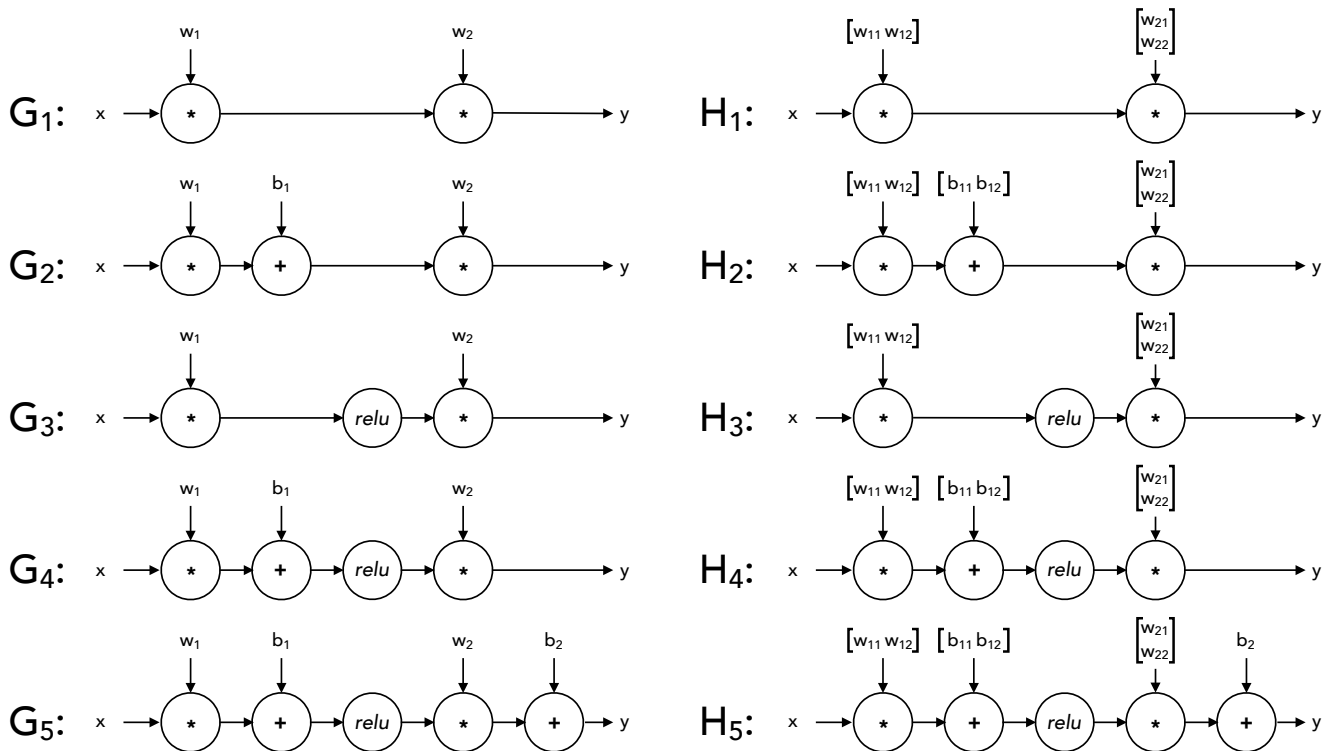
It is not too hard to show that for any vectors a and b ,

$$f(a) \cdot f(b) = (a \cdot b + 1)^2$$

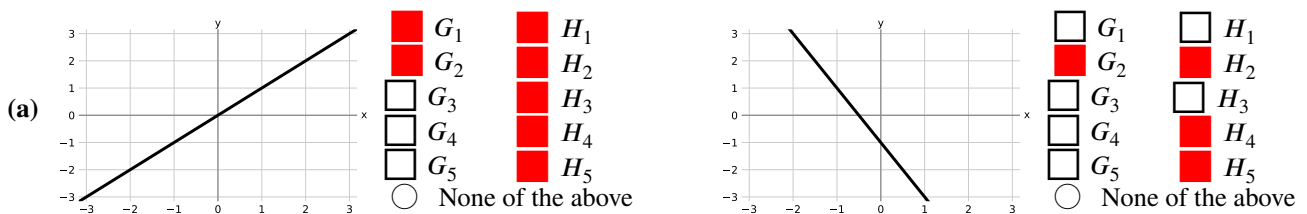
How can we take advantage of this fact when working with the implicit representation? (Your answer should be 1 sentence.)

We can replace the old features with the transformed features to classify non-linearly separable training set.

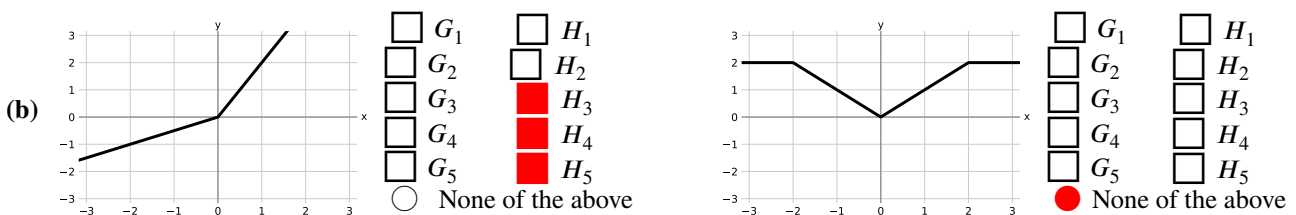
Q2. Neural Networks: Representation



For each of the piecewise-linear functions below, mark all networks from the list above that can represent the function **exactly** on the range $x \in (-\infty, \infty)$. In the networks above, *relu* denotes the element-wise ReLU nonlinearity: $relu(z) = \max(0, z)$. The networks G_i use 1-dimensional layers, while the networks H_i have some 2-dimensional intermediate layers.



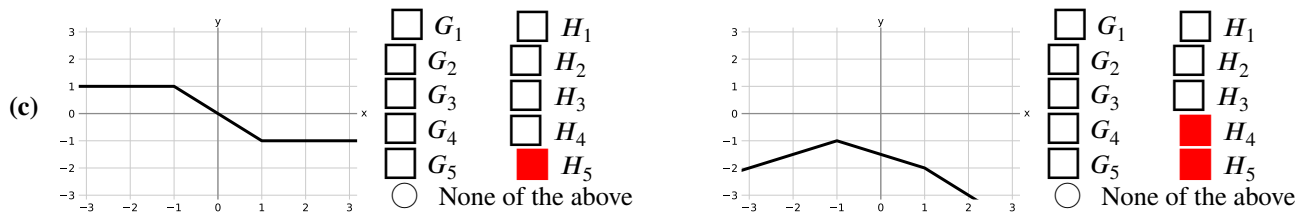
The networks G_3, G_4, G_5 include a ReLU nonlinearity on a scalar quantity, so it is impossible for their output to represent a non-horizontal straight line. On the other hand, H_3, H_4, H_5 have a 2-dimensional hidden layer, which allows two ReLU elements facing in opposite directions to be added together to form a straight line. The second subpart requires a bias term because the line does not pass through the origin.



These functions include multiple non-horizontal linear regions, so they cannot be represented by any of the networks G_i which apply ReLU no more than once to a scalar quantity.

The first subpart can be represented by any of the networks with 2-dimensional ReLU nodes. The point of nonlinearity occurs at the origin, so nonzero bias terms are not required.

The second subpart has 3 points where the slope changes, but the networks H_i only have a single 2-dimensional ReLU node. Each application of ReLU to one element can only introduce a change of slope for a single value of x .



Both functions have two points where the slope changes, so none of the networks G_i ; H_1 , H_2 can represent them.

An output bias term is required for the first subpart because one of the flat regions must be generated by the flat part of a ReLU function, but neither one of them is at $y = 0$.

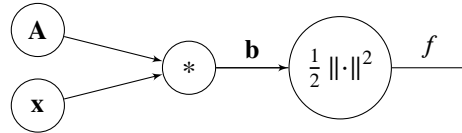
The second subpart doesn't require a bias term at the output: it can be represented as $-relu(\frac{-x+1}{2}) - relu(x + 1)$. Note how if the segment at $x > 2$ were to be extended to cross the x axis, it would cross exactly at $x = -1$, the location of the other slope change. A similar statement is true for the segment at $x < -1$.

Q3. Backpropagation

In this question we will perform the backward pass algorithm on the formula

$$f = \frac{1}{2} \|\mathbf{Ax}\|^2$$

Here, $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{b} = \mathbf{Ax} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, and $f = \frac{1}{2} \|\mathbf{b}\|^2 = \frac{1}{2} (b_1^2 + b_2^2)$ is a scalar.



(a) Calculate the following partial derivatives of f .

(i) Find $\frac{\partial f}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \end{bmatrix}$.

- $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
 $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$
 $\begin{bmatrix} b_2 \\ b_1 \end{bmatrix}$
 $\begin{bmatrix} f(b_1) \\ f(b_2) \end{bmatrix}$
 $\begin{bmatrix} A_{11} \\ A_{22} \end{bmatrix}$
 $\begin{bmatrix} b_1 + b_2 \\ b_1 - b_2 \end{bmatrix}$

(b) Calculate the following partial derivatives of b_1 .

(i) $\left(\frac{\partial b_1}{\partial A_{11}}, \frac{\partial b_1}{\partial A_{12}} \right)$

- (A_{11}, A_{12})
 $(0, 0)$
 (x_2, x_1)
 $(A_{11}x_1, A_{12}x_2)$
 (x_1, x_2)

(ii) $\left(\frac{\partial b_1}{\partial A_{21}}, \frac{\partial b_1}{\partial A_{22}} \right)$

- (A_{21}, A_{22})
 (x_1, x_2)
 $(1, 1)$
 $(0, 0)$
 $(A_{21}x_1, A_{22}x_2)$

(iii) $\left(\frac{\partial b_1}{\partial x_1}, \frac{\partial b_1}{\partial x_2} \right)$

- (A_{11}, A_{12})
 (A_{21}, A_{22})
 $(0, 0)$
 (b_1, b_2)
 $(A_{21}x_1, A_{22}x_2)$

(c) Calculate the following partial derivatives of f .

(i) $\left(\frac{\partial f}{\partial A_{11}}, \frac{\partial f}{\partial A_{12}} \right)$

- (A_{11}, A_{12})
 $(A_{11}b_1, A_{12}b_2)$
 $(A_{11}x_1, A_{12}x_2)$
 (x_1b_1, x_2b_1)
 (x_1b_2, x_2b_2)
 (x_1b_1, x_2b_2)

(ii) $\left(\frac{\partial f}{\partial A_{21}}, \frac{\partial f}{\partial A_{22}} \right)$

- (A_{21}, A_{22})
 $(A_{21}b_1, A_{22}b_2)$
 $(A_{21}x_1, A_{22}x_2)$
 (x_1b_1, x_2b_1)
 (x_1b_2, x_2b_2)
 (x_1b_1, x_2b_2)

(iii) $\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$

- $(A_{11}b_1 + A_{12}b_2, A_{21}b_1 + A_{22}b_2)$
 $(A_{11}b_1 + A_{21}b_2, A_{12}b_1 + A_{22}b_2)$
 $(A_{11}b_1 + A_{12}b_1, A_{21}b_2 + A_{22}b_2)$
 $(A_{11}b_1 + A_{21}b_1, A_{12}b_2 + A_{22}b_2)$

(d) Now we consider the general case where \mathbf{A} is an $n \times d$ matrix, and \mathbf{x} is a $d \times 1$ vector. As before, $f = \frac{1}{2} \|\mathbf{Ax}\|^2$.

(i) Find $\frac{\partial f}{\partial \mathbf{A}}$ in terms of \mathbf{A} and \mathbf{x} only.

- $\mathbf{x}^T \mathbf{A}^T \mathbf{Ax}$
 \mathbf{Axx}^T
 $\mathbf{A} (\mathbf{A}^T \mathbf{A})^{-1}$
 $\mathbf{AA}^T \mathbf{Ax}$
 \mathbf{A}

(ii) Find $\frac{\partial f}{\partial \mathbf{x}}$ in terms of \mathbf{A} and \mathbf{x} only.

- \mathbf{x}
 $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{x}$
 $\mathbf{xx}^T \mathbf{x}$
 $\mathbf{x}^T \mathbf{A}^T \mathbf{Ax}$
 $\mathbf{A}^T \mathbf{Ax}$