ECE/CS 498 DSU/DSG Spring 2020
In-Class Activity 6

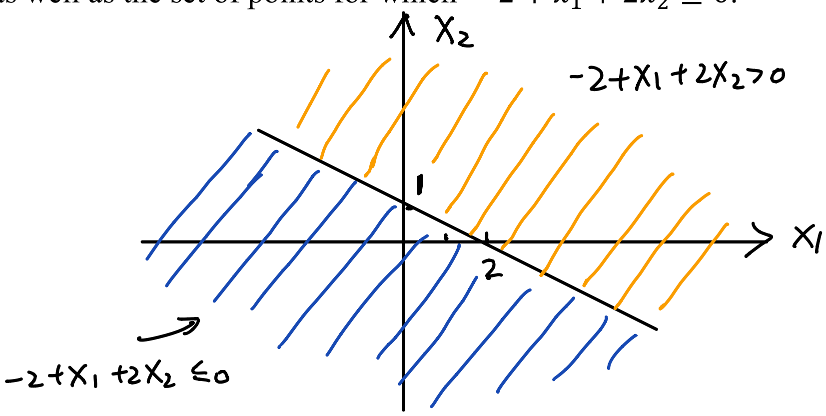
NetID: chuhaof2

The purpose of the in-class activity is for you to:
(i) Review concepts related to SVM and neural networks
(ii) Work out steps in backpropagation for optimization of a neural network

Support Vector Machine

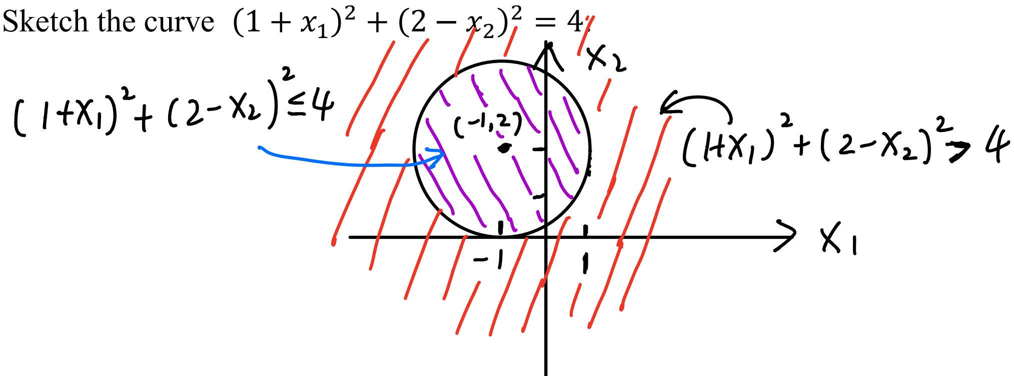
1. Linear decision boundary

Sketch the hyperplane $-2 + x_1 + 2x_2 = 0$. Indicate the set of points for which $-2 + x_1 + 2x_2 > 0$, as well as the set of points for which $-2 + x_1 + 2x_2 \leq 0$.



2. Non-linear decision boundary

a) Sketch the curve $(1 + x_1)^2 + (2 - x_2)^2 = 4$



b) On your sketch, indicate the region for which $(1 + x_1)^2 + (2 - x_2)^2 > 4$, as well as the region for which $(1 + x_1)^2 + (2 - x_2)^2 \leq 4$.

c) Suppose that a classifier assigns an observation (x_1, x_2) to the blue class if $(1 + x_1)^2 + (2 - x_2)^2 > 4$, and to the red class otherwise. To what class is the observation $(0, 0)$ classified? $(-1, 1)$?

$(0, 0) \rightarrow$ blue class

$(-1, 1) \rightarrow$ red class

d) The decision boundary equation in c) is not linear in terms of an input $\boldsymbol{x} = (x_1,\ x_2)$ since it has x_1^2 and x_2^2 terms. However, suppose that we instead consider $\boldsymbol{x} = (x_1,\ x_1^2,\ x_2,\ x_2^2)$. This might be the case after applying a kernel transformation to the dataset. Is the decision boundary equation linear then? What might be a reason for applying this kernel transformation?

after sinplificatiom: $(1+x_1)^2 + (2-x_2)^2 = 4 \Rightarrow 1+2x_1+x_1^2-4x_2+x_2^2 = 0$

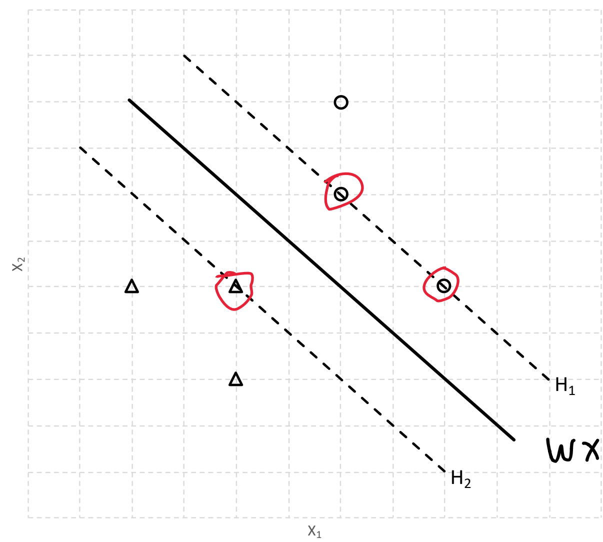
$\phi(x) = [1, x_1, x_2, x_1^2, x_2^2] \in K(x_i, x_j) = 1+x_1+x_2+x_1^2+x_2^2+x_1x_2$

(Polynomial kernel of order 2) → the decision boundary will be linear

Original data may not be separated by SVM, but it may be separated after applying this kernel transformation.

3. Hard margin SVM

Suppose we are learning a hard margin SVM with two real-valued features x_1, x_2 and binary label $y \in \{-1, +1\}$ (represented by $\triangle$ and $\circ$, respectively). The training data is pictured in the figure below. Our linear classifier takes the form $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$.



$wx + b = 0$ (decision boundary)

a) According to the maximum margin principle, identify the support vectors, and sketch the decision boundary of the trained SVM.
The red-circled points are the support vetors.

b) Suppose hyperplane H_1 takes the form $\boldsymbol{w} \cdot \boldsymbol{x} + b = 1$, write down the equations for H_2.

$H_2:\ w \cdot x + b = -1$

c) The constraints for linear hard margin SVM can be written as $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$, $\forall i \in \{1,\ \ldots,\ N\}$. Explain why.

This equation can be interpreted as any given point in the space is at least $\frac{1}{\|w\|}$ away from the hyperplane and the training set are seperated 100% correct → this is the requirement of the hard Margin SVM

d) What condition do the data points have to satisfy such that a feasible $\boldsymbol{w}$ exists?

All data points are classified correctly

e) Calculate the distance between H_1 and H_2.

$$\text{Margin distance} = \frac{\vec{w}^T\vec{x}+b+\vec{1}-(\vec{w}^T\vec{x}+b-\vec{1})}{||\vec{w}||} = \frac{2}{||\vec{w}||}$$

f) Based on your answer to the above questions, write down the optimization problem whose solution is hard margin SVM.

$$\max\frac{2}{||\vec{w}||} \Rightarrow \min||\vec{w}|| \Rightarrow \min\frac{||\vec{w}||^2}{2}$$

$$\Rightarrow \operatorname{argmin}_{\vec{w}}\frac{1}{2}||\vec{w}||^2 \text{ such that } 1 \leq y_i(\vec{w}^T\vec{x_i} + b)\ \forall i \in \{1, \ldots, N\}$$

g) For the following data points $\boldsymbol{x_1} = (\mathbf{1}, \mathbf{2}, \mathbf{3}), \boldsymbol{x_2} = (\mathbf{4}, \mathbf{1}, \mathbf{2}), \boldsymbol{x_3} = (-\mathbf{1}, \mathbf{2}, -\mathbf{1})$ corresponding to class $y_1 = +1, y_2 = +1, y_3 = -1$, one of the following $\boldsymbol{w}$, b gives the correct SVM decision boundary ($\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$). Which one is it? Show your work.

A. $w = [0.3, 0, 0.4]'$, $b = -0.4$
B. $w = [0.2, 0, 0.4]'$, $b = -0.4$
C. $w = [0.1, 0, 0.4]'$, $b = -0.4$
D. $w = [0.4, 0, 0.2]'$, $b = -0.4$

B is the correct SVM decision boundary.

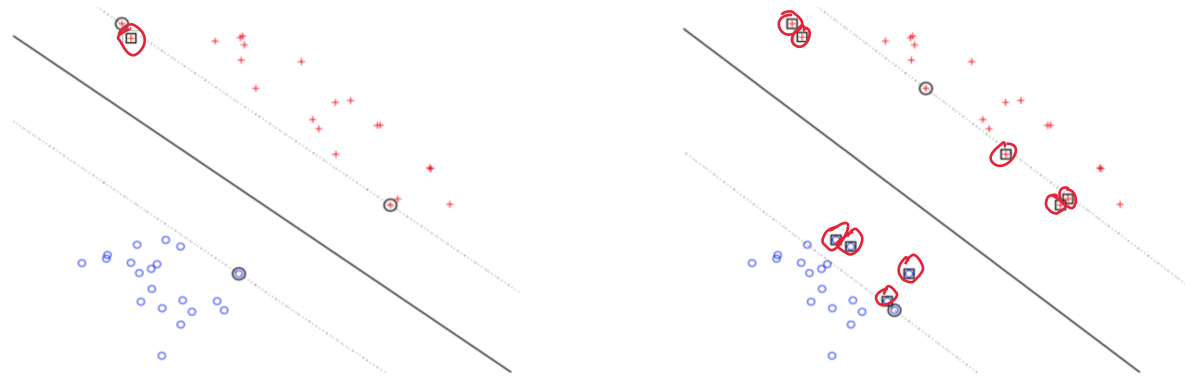
```
A.
For y1: 1.1
For y2: 1.6
For y3: 1.1
B.
For y1: 1.0
For y2: 1.2000000000000002
For y3: 1.0
C.
For y1: 0.9000000000000002
For y2: 0.8000000000000002
For y3: 0.9
D.
For y1: 0.6
For y2: 1.6
For y3: 1.0

Compare the objective for A and B
A: 0.125
B: 0.10000000000000002
```

4. Soft margin SVM

Recall the program for solving the soft margin SVM:

$$\min_{\boldsymbol{w},\ \xi_I \geq 0} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^N \xi_i \qquad s.t. \quad 1 - \xi_i \leq y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) \quad \forall i \in \{1,\ \ldots,\ N\}$$

We have plotted the SVM solutions for a training dataset in Figure (a) and (b) corresponding to two values of C:



(a) (b)

a) Indicate non-zero ξ_is on both plots (a) and (b).

The red-circled points are those non-zero ξ_is.

b) Which figure corresponds to a larger value of C? Explain why.

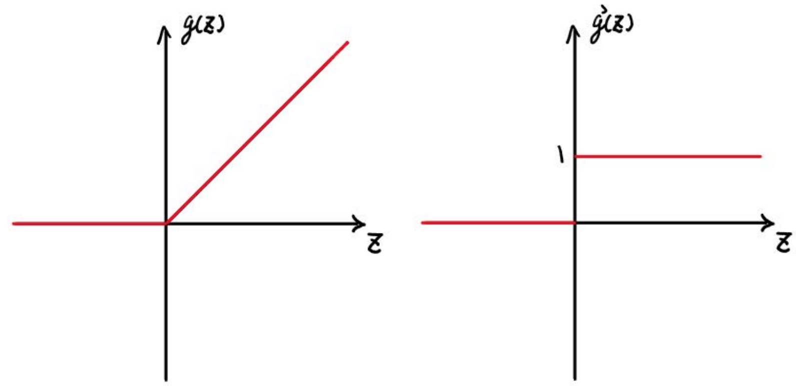
Figure (a) corresponds to a larger value of C, because the amount of allowed slack is smaller in (a) than that in (b). With a larger C, the amount of allowed slack should be smaller.

The unconstrained form of the above optimization problem is given as:

$$\min_{\boldsymbol{w},\mathrm{b}} \frac{1}{2}||\boldsymbol{w}||^2 + \mathrm{C}\sum_{i}^{N} \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\}$$

c) Draw the function $g(z) = \max\{0, \mathrm{z}\}$ for scalar variable z. What is the derivative $\frac{dg(z)}{dz}$? Draw the derivative.

$$\frac{dg(z)}{dz} = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \\ \text{undefined} & z = 0 \end{cases}$$



d) Compute the gradient of this unconstrained program w.r.t $\boldsymbol{w}$. [Hint: Think of z in part (c) as $1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)$, you can express max in terms of an indicator function]

$$\Rightarrow C\vec{w} + \sum_{i=1}^{M} -y_i \vec{x_i}, \text{ where } x_i \in \{x_1, x_2, \ldots, x_M\} \text{ satisfy } 1 - y_i(\vec{w}^T \vec{x}_i + b) > 0$$

e) Suppose you are training your SVM using gradient descent and the gradient derived in (d), the original gradient is $\boldsymbol{w} = [2,2]', b = -1$. The first iteration we trained on data point $\boldsymbol{x_1} = (1,1), y_1 = 1$; the second iteration we trained on data point $\boldsymbol{x_2} = (-1,-1), y_2 = -1$, $\lambda = 1$, assuming both C and learning-rate to be 1 calculate the gradient after these two iterations.

$$\because 1 - y_1(\vec{w}^T \vec{x_1} + b) = -2 < 0$$

$$\therefore \nabla\vec{w} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\therefore \vec{w}_1 = \vec{w} - 1 \times \nabla\vec{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\because 1 - y_2(\vec{w}^T \vec{x_2} + b) = 0$$

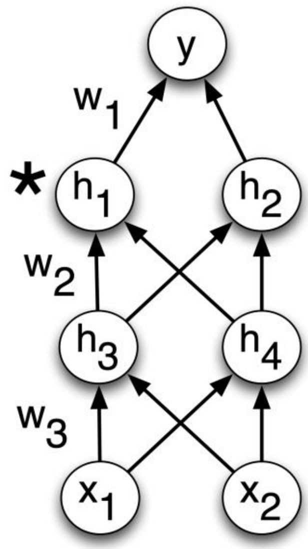
$$\therefore \nabla\vec{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \vec{x_2}$$

$$\therefore \vec{w}_2 = \vec{w_1} - 1 \times \nabla\vec{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Neural Networks

1. Partial derivatives

   Consider the network shown in the figure below. All the hidden units use the ReLU- $h_i(\mathrm{z_i}) = \max\{z_i, 0\}$. We are trying to minimize a cost function C which depends only on the activation of the output unit y. The unit h_1 (marked with a *) receives an input of $z_i = -1$ on a training iteration, so its output is 0. Based only on this information, which of the following weight derivatives are guaranteed to be 0 for this training case? Write YES or NO for each. Justify your answers informally. (Hint: don't work through the backprop computations. Instead think about what the partial derivatives really mean.)



a) $\frac{\partial C}{\partial w_1}$ NO

b) $\frac{\partial C}{\partial w_2}$ YES

c) $\frac{\partial C}{\partial w_3}$ YES

$\because z_i = -1$

$\therefore h_1'(z_i) = 0,$ according to problem 4.(c) in SVM

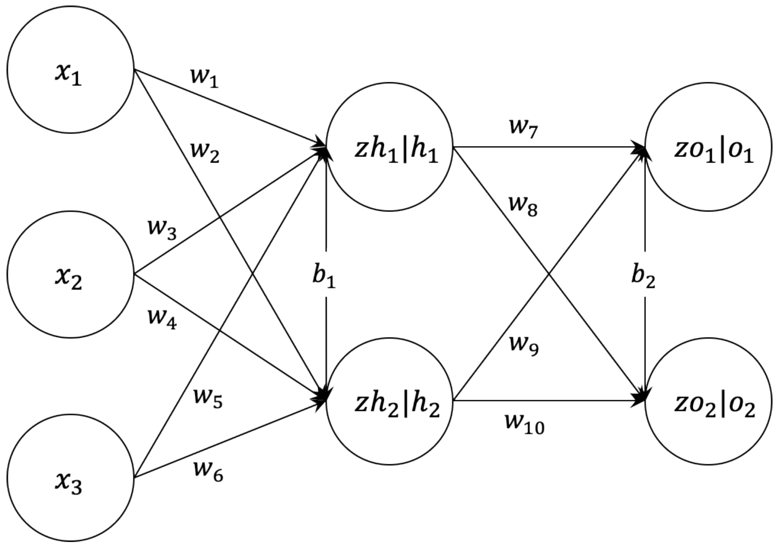
$\therefore$ any weight derivative including $\frac{dh_1(z_i)}{z_i}$ should be 0

$\because$ based on the network, $\frac{\partial C}{\partial w_2}$ and $\frac{\partial C}{\partial w_3}$ includes $\frac{dh_1(z_i)}{z_i}$

$\therefore \frac{\partial C}{\partial w_2}$ and $\frac{\partial C}{\partial w_3}$ are gauranteed to be 0 while the value of $\frac{\partial C}{\partial w_1}$ is not sure

2. Backpropagation

The neural network considered in this question has three input neurons, one hidden layer with two neurons, and one output layer with two neurons. b_1 and b_2 are bias terms.



a) Suppose $zh_1 = w_1x_1 + w_3x_2 + w_5x_3 + b_1$, $h_1 = \text{sigmoid}(\text{zh}_1)$, and similar relationships hold for the other neurons in the hidden/output layer. Assume the current parameters of the networks $w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10} = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.1$, $b_1, b_2 = 0.5, 0.5$. Given input $x_1, x_2, x_3 = 1, 4, 5$, use forward propagation to find out the value of h_1, o_1. You can think of writing a Python program.

In [4]:
```
def sigmoid(x):
    return 1/(1 + np.exp(-x))

x = np.array([1,4,5])
w_h = np.array([[.1,.2],
                [.3,.4],
                [.5,.6]])
w_o = np.array([[.7,.9],
                [.8,.1]])
b_1 = np.array([.5,.5])
b_2 = np.array([.5,.5])

zh = np.dot(x, w_h) + b_1
zo = np.dot(zh, w_o) + b_2

print('h1\n', sigmoid(zh[0]))
print('o1\n', sigmoid(zo[0]))
```

```
h1
 0.9866130821723351
o1
 0.9995694429186754
```

b) Let t_1, t_2 represent the true labels. Define the sum of squared loss $E = \frac{1}{2}((o_1 - t_1)^2 + (o_2 - t_2)^2)$. **Write down the partial derivatives $\frac{\partial E}{\partial w_7}, \frac{\partial E}{\partial \mathbf{b}_2}, \frac{\partial E}{\partial w_1}$ according to the chain rule**. Example for $\frac{\partial E}{\partial w_{10}}, \frac{\partial E}{\partial w_2}$ is given below.

$$\frac{\partial E}{\partial w_{10}} = \frac{\partial E}{\partial o_2}\frac{\partial o_2}{\partial zo_2}\frac{\partial zo_2}{\partial w_{10}}$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial o_2}\frac{\partial o_2}{\partial zo_2}\frac{\partial zo_2}{\partial h_2}\frac{\partial h_2}{\partial zh_2}\frac{\partial zh_2}{\partial w_2} + \frac{\partial E}{\partial o_1}\frac{\partial o_1}{\partial zo_1}\frac{\partial zo_1}{\partial h_2}\frac{\partial h_2}{\partial zh_2}\frac{\partial zh_2}{\partial w_2}$$

For $\frac{\partial E}{\partial w_2}$, look at the two paths which lead from w_2 to E. Backpropagation includes both the paths in the calculation.

$$\frac{\partial E}{\partial w_7} = \frac{\partial E}{\partial o_1}\frac{\partial o_1}{\partial zo_1}\frac{\partial zo_1}{\partial w_7}$$

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial o_1}\frac{\partial o_1}{\partial zo_1}\frac{\partial zo_1}{\partial b_2} + \frac{\partial E}{\partial o_2}\frac{\partial o_2}{\partial zo_2}\frac{\partial zo_2}{\partial b_2}$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial o_1}\frac{\partial o_1}{\partial zo_1}\frac{\partial zo_1}{\partial h_1}\frac{\partial h_1}{\partial zh_1}\frac{\partial zh_1}{\partial w_1} + \frac{\partial E}{\partial o_2}\frac{\partial o_2}{\partial zo_2}\frac{\partial zo_2}{\partial h_1}\frac{\partial h_1}{\partial zh_1}\frac{\partial zh_1}{\partial w_1}$$

c) Suppose $t_1, t_2 = 0.1, 0.05$, learning rate $\alpha = 0.01$. Calculate the updated w_7, b_2, w_1 after one iteration of backpropagation. Use $h_2 = 1$, $o_2 = 0.8$.

$$\because \frac{\partial E}{\partial o_1} = o_1 - t_1 = 0.89956944, \ \frac{\partial E}{\partial o_2} = o_2 - t_2 = 0.75$$

$$\because \frac{\partial o_1}{\partial zo_1} = o_1(1 - o_1) = 0.00043037, \ \frac{\partial o_2}{\partial zo_2} = o_2(1 - o_2) = 0.16$$

$$\because \frac{\partial zo_1}{\partial w_7} = h_1 = 0.98661308, \ \frac{\partial zo_1}{\partial b_2} = 1, \ \frac{\partial zo_2}{\partial b_2} = 1$$

$$\because \frac{\partial zo_1}{\partial h_1} = w_7 = 0.7, \ \frac{\partial zo_2}{\partial h_1} = w_8 = 0.8$$

$$\because \frac{\partial h_1}{\partial zh_1} = h_1(1 - h_1) = 0.013207710, \ \frac{\partial zh_1}{\partial w_1} = x_1 = 1$$

$$\therefore \frac{\partial E}{\partial w_7} = 0.00038196, \ \frac{\partial E}{\partial b_2} = 0.12038715, \ \frac{\partial E}{\partial w_1} = 0.00127152$$

$$\therefore w_7^+ = w_7 - \alpha \times \frac{\partial E}{\partial w_7} = 0.69999618$$

$$\therefore b_2^+ = b_2 - \alpha \times \frac{\partial E}{\partial b_2} = 0.49879613$$

$$\therefore w_1^+ = w_1 - \alpha \times \frac{\partial E}{\partial w_1} = 0.09998728$$

3. Neural Network Playground (for you to explore)
https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises