ECE/CS 498 DSU/DSG Spring 2019
In-Class Activity 6

NetID:

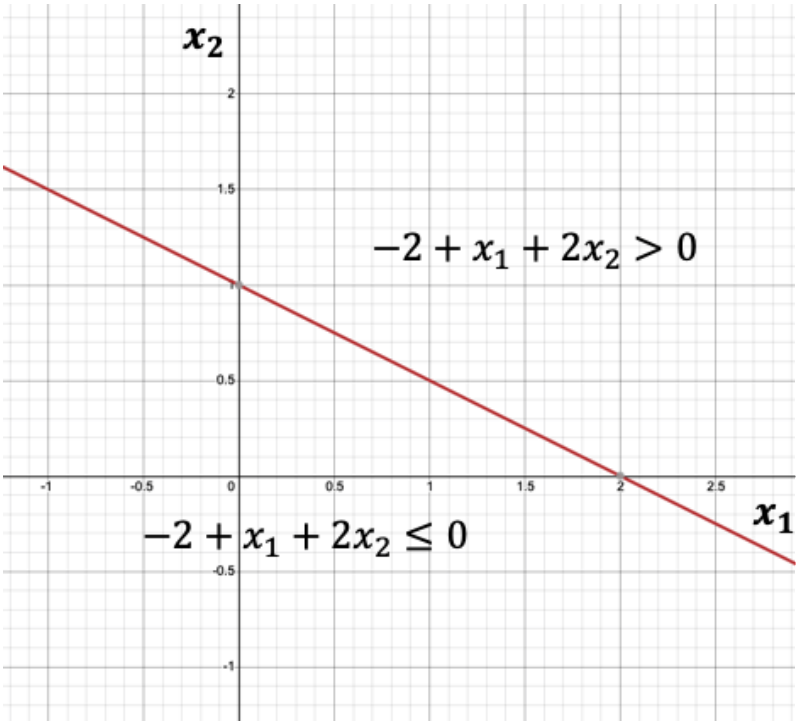
The purpose of the in-class activity is for you to:

(i) Review concepts related to SVM and neural networks
(ii) Work out steps in backpropagation for optimization of a neural network

Support Vector Machine

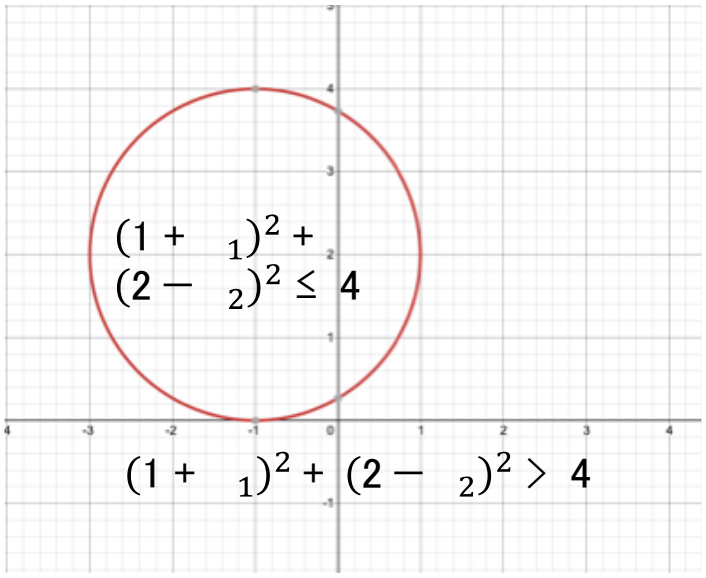
1. Linear decision boundary

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 $(1 + x_1)^2 + (2 - x_2)^2 = 4$.

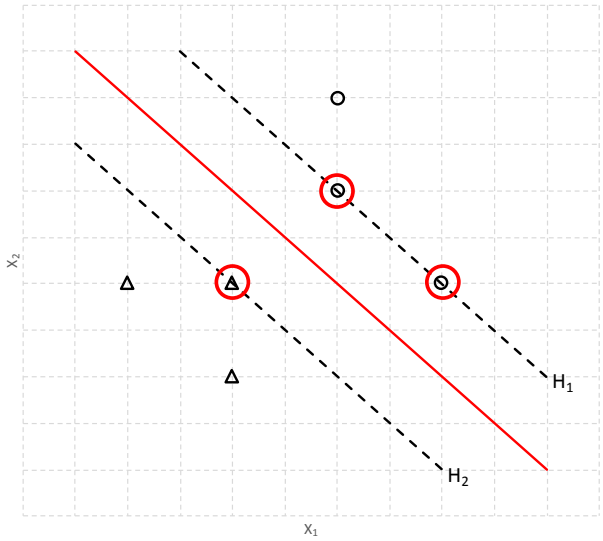


b) On your sketch, indicate the region for which $(1 + x_1)^2 + (2 - x_2)^2 > 4$, as well as the region for which $(1 + x_1)^2 + (2 - x_2)^2 \leq 4$.
See the plot above.

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): $(1 + 0)^2 + (2 - 0)^2 = 5 > 4$ blue
(-1,1): $(1 - 1)^2 + (2 - 1)^2 = 1 < 4$ red

d) Argue that while the decision boundary in c) is not linear in terms of x_1 and x_2, it is linear in terms of x_1, x_1^2, x_2, and x_2^2.
$2x_1 + x_1^2 - 4x_2 + x_2^2 + 1 = 0$

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 Δ and $\circ$, respectively). The training data is pictured in the figure below. Our linear classifier takes the form $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$.



a) According to the maximum margin principle, identify the support vectors, and sketch the decision boundary of the trained SVM.
The support vectors are annotated with red circles, and the decision boundary is shown by the red line.

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

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$$

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.
The constraints correspond to the case where no data points lie within the margin, and all points are classified correctly.

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

e) Calculate the distance between H_1 and H_2.

$$\frac{2}{\|w\|}$$

https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/

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

The goal is to maximize the margin between the two classes (distance between hyperplanes H_1 and H_2) while satisfying the constraint that all points are correctly classified and none of them lies within the margin.

$$\underset{w}{\text{argmax}} \frac{2}{\|w\|} \text{ or } \underset{w}{\text{argmin}} \frac{1}{2}\|w\|^2 \qquad s.t.\ y_i(\boldsymbol{w}\cdot\boldsymbol{x_i}+b) \geq 1,\ \forall i \in \{1,\ \ldots,\ N\}$$

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 gives the correct decision boundary.

$$\underset{w}{\text{argmax}} \frac{2}{\|w\|} \text{ or } \underset{w}{\text{argmin}} \frac{1}{2}\|w\|^2 \qquad s.t.\ y_i(\boldsymbol{w}\cdot\boldsymbol{x_i}+b) \geq 1,\ \forall i \in \{1,\ \ldots,\ N\}$$

C and D do not satisfy the above constraints (not all points are in the correct side of the margin).

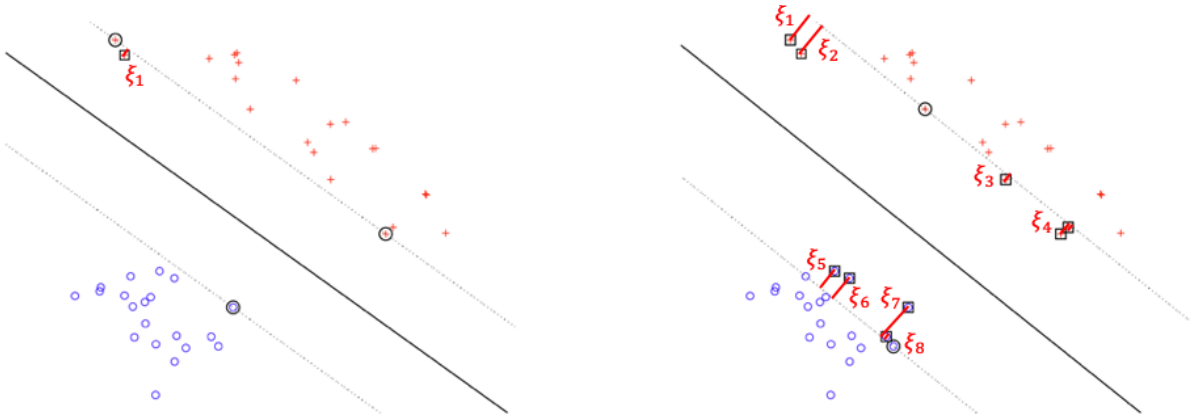
B gives a larger $\frac{2}{\|w\|}$ (margin) than A does.

4. Soft margin SVM

Recall the program for solving the soft margin SVM:

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

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



(a) (b)

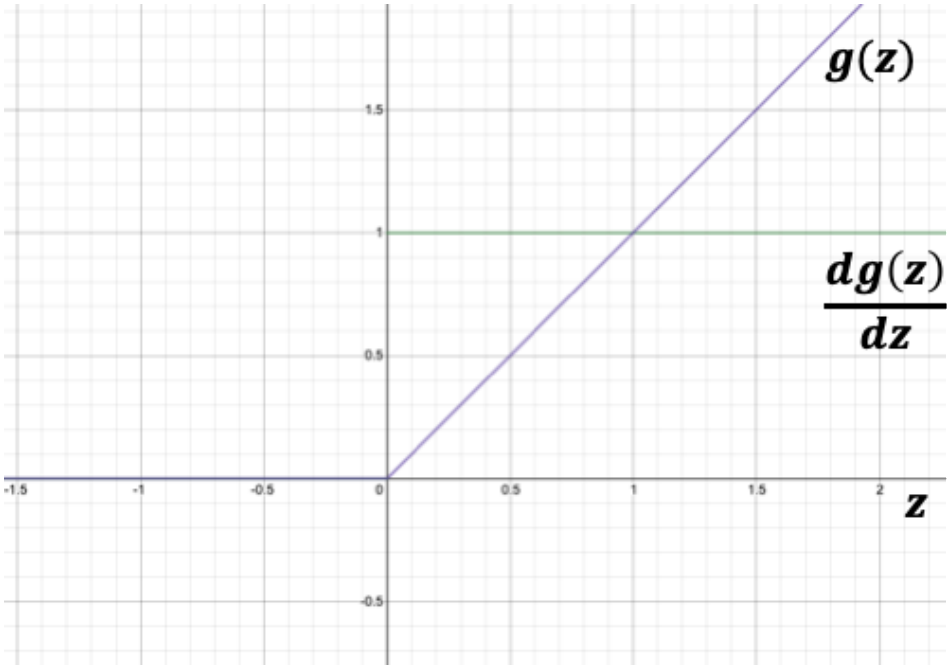
a) Indicate non-zero ξ_is on the plot
See the plot above.

b) Which figure corresponds to a larger value of C? Explain why.
In comparison with Figure (b), Figure (a) separates the two classes more strictly, which corresponds to the case where C is large and fewer errors are allowed.

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

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

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.



See the plot above. $g(z)$ is given as the purple curve, and $\frac{dg(z)}{dz}$ is given as the green curve (a step function).

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

$$\boldsymbol{w} + \mathrm{C}\sum_{i}^{N} \delta(y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) < 1)(-y_i\boldsymbol{x_i})$$

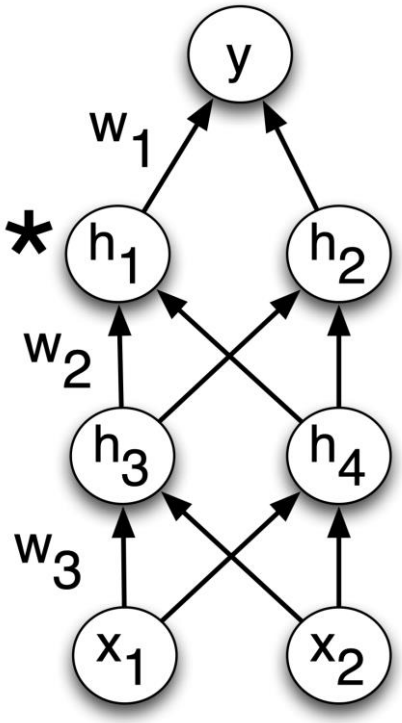
e) Evaluate the gradient at $\boldsymbol{w} = [2,2]', b = -1,\ \boldsymbol{x_1} = (1,1), y_1 = 1,\ \boldsymbol{x_2} = (-1,-1), y_2 = -1,\ \lambda = 1.$

$y_1(\boldsymbol{w}^T\boldsymbol{x_1} + b) = 1 \times (2 + 2 - 1) = 3 > 1,\ \delta(y_1(\boldsymbol{w}^T\boldsymbol{x_1} + b) < 1) = 0,$
$\delta(y_1(\boldsymbol{w}^T\boldsymbol{x_1} + b) < 1)(-y_1\boldsymbol{x_1}) = 0$
$y_2(\boldsymbol{w}^T\boldsymbol{x_2} + b) = -1 \times (-2 - 2 - 1) = 5 > 1,\ \delta(y_2(\boldsymbol{w}^T\boldsymbol{x_2} + b) < 1) = 0,$
$\delta(y_2(\boldsymbol{w}^T\boldsymbol{x_2} + b) < 1)(-y_2\boldsymbol{x_2}) = 0$
Thus, the gradient $\boldsymbol{w} + \mathrm{C}\sum_i^N \delta(y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) < 1)(-y_i\boldsymbol{x_i}) = \boldsymbol{w} + 0 = [2,2]'$

Neural Networks

1. Partial derivatives
   Consider the network shown in the figure below. All of the hidden units use the ReLU- $h_i = \max\{z_i, 0\}$. 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 -1 on a particular training case, 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) $\frac{\partial C}{\partial w_1}$

YES. Because h_1 is zero, and therefore changing w_1 doesn't affect the input to unit y. Therefore, it doesn't affect the output of the network, or the cost.

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

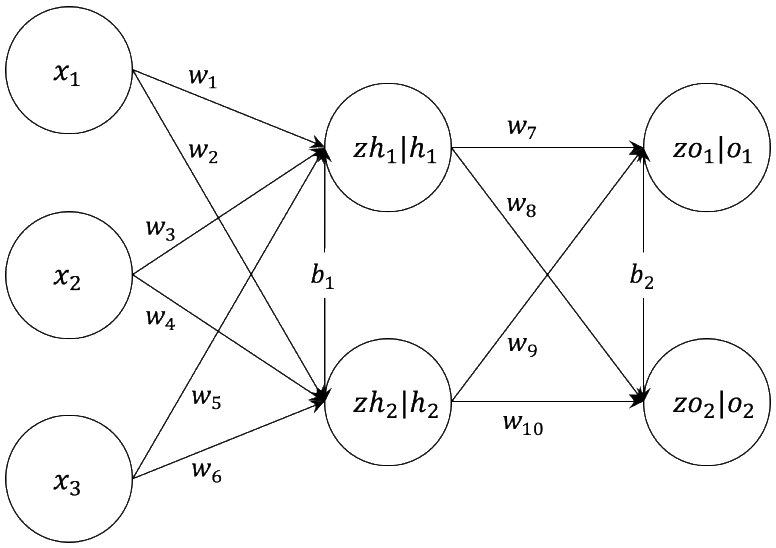
YES. Because the input z_1 is negative, $\frac{\partial h_1}{\partial z_1} = 0$, so changing w_2 by a small amount doesn't change z_1. Therefore, it has no effect on the output of the network.

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

NO. Changing w_3 by a small amount can change h_3, which can change h_2, which can change y, which can change C.

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.
(0.9866130821723351, 0.889550613969795)

```
wList = [w1,w2,w3,w4,w5,w6,w7,w8,w9,w10]
bList = [b1,b2]
xList = [x1,x2,x3]
h1, h2, o1, o2 = forwardProp(xList, wList, bList)

def forwardProp(xList, wList, bList):
    zh1 = wList[0]*xList[0] + wList[2]*xList[1] + wList[4]*xList[2] + bList[0]
    zh2 = wList[1]*xList[0] + wList[3]*xList[1] + wList[5]*xList[2] + bList[0]
    h1 = sigmoid(zh1)
    h2 = sigmoid(zh2)
    zo1 = wList[6]*h1 + wList[8]*h2 + bList[1]
    zo2 = wList[7]*h1 + wList[9]*h2 + bList[1]
    o1 = sigmoid(zo1)
    o2 = sigmoid(zo2)
    return h1,h2,o1,o2
```

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 $\frac{\partial E}{\partial w_7}, \frac{\partial E}{\partial \text{b}_2}, \frac{\partial E}{\partial w_1}$ according to the chain rule. Example for $\frac{\partial E}{\partial w_{10}}, \frac{\partial E}{\partial w_2}$ is given below.

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

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

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.

dE_do1 = o1-t1
do1_dzo1 = o1*(1-o1)
dzo1_dw7 = h1
dE_dw7 = dE_do1*do1_dzo1*dzo1_dw7

dE_do2 = o2-t2
do2_dzo2 = o2*(1-o2)
dzo2_dw8 = h1
dE_dw8 = dE_do2*do2_dzo2*dzo2_dw8

dzo1_db2 = 1
dzo2_db2 = 1
dE_db2 = dE_do1*do1_dzo1*dzo1_db2 + dE_do2*do2_dzo2*dzo2_db2

dzo1_dh1 = w7
dzo2_dh1 = w8
dE_dh1 = dE_do1*do1_dzo1*dzo1_dh1 + dE_do2*do2_dzo2*dzo2_dh1
dh1_dzh1 = h1*(1-h1)
dzh1_dw1 = x1
dE_dw1 = dE_dh1*dh1_dzh1*dzh1_dw1

For a step by step walkthrough:
https://www.anotsorandomwalk.com/backpropagation-example-with-numbers-step-by-step/

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 $h_2 = 1$, $o_2 = 0.8$.
(0.099980160881276, 0.6992346487159412, 0.4980254250216812)

w1 = w1 - alpha*dE_dw1
w7 = w7 - alpha*dE_dw7
b2 = b2 - alpha*dE_db2

3. Neural Network Playground
https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises
Please take a look ☺