CAP 5615 Introduction to Neural Networks

2021 Summer

Homework 3 (**18 pts, Due: June 13 2021, Late Penalty: -2/day**)

[If two homework submissions are found to be similar to each other, both submissions will receive 0 grade]

- Homework solutions must be submitted through Canvas. No email submission is accepted.
- Please try to put all results in one file (e.g., one pdf or word file).
- If you have multiple files, please upload them separately (only **pdf**, **word**, and **html** files are allowed).

You can always update your submissions. Only the latest version will be graded.]

Question 1 [1 pt] A neural network is specified by three major components: (1) architecture, (2) neuron model, and (3) learning algorithm.

- What is neural network architecture? What are common neural network architectures?

The architecture of the neural network specifies the connection (link) of neurons. In other words, it specifies how neuros are connected to each other to form a network. The architecture of the neural network determines the functionality and learning capability of the network.

Common neural network architectures include single layer feedforward neural network, multi-layer feedforward neural network, and recurrent neural network.

- What is neuron model? What are key components of a neuron model?

Neuron model defines the structure of the single neuron used in the network. It specifies the information processing process in each unit of the neural network.

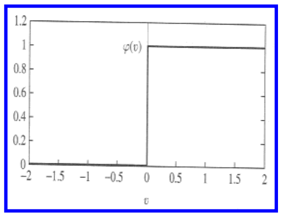
The key components of a neuron model include: input, weight, summing function, local field, activation function, and output

- What is learning algorithm? What is the purpose of the learning algorithm?

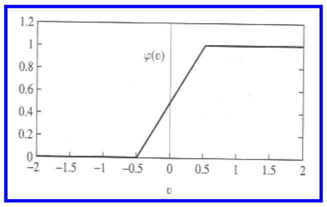
Learning algorithms defines the way of modifying the weight values of the neural networks, in order to train the network to solve the underlying learning task.

The purpose of the learning algorithm is to help tune/update the weight values of the network, so the network can classify input instances into correct categories.

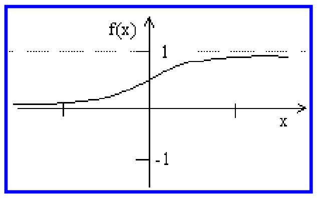
Question 2 [1pt] Figure 1 shows four activation function in the neural network, please show the mathematical formulation of each activation function [0.5], and explain the characteristics of each activation function [0.5 pt].



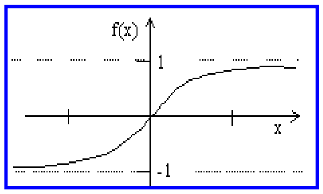
(a)



(b)



(c)



(d)

Figure 1 Activation Functions.

(a): hard-limiter function. It's a step function controlled by a threshold value. It's not a continuous function.

$$\varphi(v)=\begin{cases}1 & if\ v\geq 0\\ 0 & if\ v<0\end{cases}$$

(b): piecewise linear function. It's a function contains a linear segment specifying the linear response of the active function with respect to the input value. It's not a continuous function.

$$\varphi(v)=\begin{cases}1 & if\ v\geq 1/2\\ v & if\ \ 1/2\geq v\geq -1/2\\ 0 & if\ v\leq -1/2\end{cases}$$

(c) Sigmoid function: It's a continuous no-linear function differentiable everywhere.

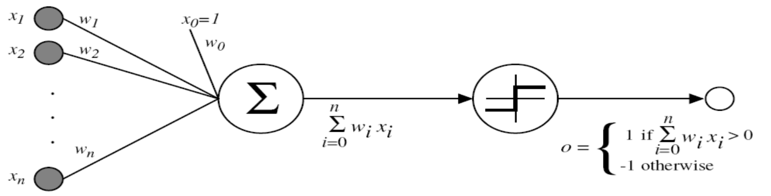
$$\varphi(v)=\frac{1}{1+\exp(-av)}$$

(d) Hyperbolic tangent function: It's a continuous no-linear function differentiable everywhere. Comparing to the sigmoid function, hyperbolic tangent function is centered around 0. Its first derivative has a larger range [0,1], than sigmoid function. Which gives hyperbolic function stronger gradient for learning

$$\varphi(v)=\tanh(v)$$

Question 3 [1 pt] Please show the perceptron structure and explain the function of each component [0.5 pt]. What is the purpose of using training examples in a neural network? Given proper weight values, what is expected output vs. actual output of an example? [0.5 pt]

Perceptron structure


x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

x_1, x_2, …, x_n correspond to the input features.

x_0 is the bias of the perceptron.

w_1, w_2, …, w_n are the weight values corresponding to each input feature.

Σ is the adder function which calculates weighted sum of each feature and the corresponding weight values.

An activation function is used to determine the output of the network. In the above perceptron structure, a thresholding function determines the output of the perceptron, depending on the results from Σ.

O() is the output generated from the network.

What is the purpose of using training examples in a neural network?

Using training sample (including positive vs. negative samples) can help train the weight values of the network to minimize the errors.

Given proper weight values, what is expected output vs. actual output of an example?

Expected output is the label of the input instance.

Actual output is the actual result generated from the perceptron, i.e. the O() value.

Question 4 [1 pt] What is a Perceptron Learning Rule? [0.5 pt]. What is the learning objective of the perceptron learning rule? [0.25 pt]

A perceptron learning rule is considered supervised training, where the learning rule is provided with a set of examples of proper network behavior. As each input is applied to the network, the network output is compared to the target. More specifically, each misclassified training example will incur a weight updating process, and the learning rule will adjust the weights and biases of the network in order to move the network output to minimize the total number of misclassified training examples.

<u>What is the learning objective of the perceptron learning rule?</u>

The learning objective of the perceptron learning rule is to minimize the number of misclassified instances (or examples)

Question 5 [1 pt] What is a Gradient Descent Learning Rule? [0.25 pt]. What is the learning objective of the Gradient Descent Learning rule [0.25 pt]. What are the differences between perceptron learning rule vs. gradient descent learning rule? [0.5 pt]

Gradient descent learning rule is a commonly used learning approach to update weight values for neural networks. The gradient descent rules specifies that the weight updating of the network should follow the steep descent (i.e., the opposite of the gradient) of the objective function E(W) with respect to the network weight values.

$$\boxed{w(k+1) = w(k) - \eta(\text{gradient of } E(W))}$$

<u>What is the learning objective of the Gradient Descent Learning rule [0.25 pt]?</u>

The learning objective of the gradient descent learning rule is to minimize the (total) squared error of the network.

<u>What are the differences between perceptron learning rule vs. gradient descent learning rule? [0.5 pt]</u>

The differences between perceptron learning rule, gradient descent learning rule, and Delta rule

	Perceptron	Gradient Descent
Architecture	Single-Layer	Single-Layer
Linear separable	Linear separable	Even when training data contains noise and not linear separable.
Learning Algorithm	Minimize the number of unclassified examples	Minimize the squared error

Weight update	After each misclassified example	After all training examples were fed into the network
Application	Linear classification	Linear classification and regression

Question 6 [1 pt] What is stochastic gradient descent learning [0.5 pt]? What are the differences between gradient descent learning vs. stochastic gradient descent learning [0.5 pt]?

Stochastic gradient descent learning is a special case of the gradient descent learning to update the weight values of a network by using gradient with respect to the weight values. Stochastic gradient descent weight updating is based on the gradient calculated using a single instance, instead of using all training instances (like gradient descent learning does).

What are the differences between gradient descent learning vs. stochastic gradient descent learning [0.5 pt]?

The main difference between gradient descent learning vs. stochastic gradient descent learning is the calculation of the gradient for weight updating. Gradient descent learning calculate gradient using a batch of instances D.

- Gradient descent
  - $w(k+1)=w(k) - \eta\nabla E_D(W)$ over the entire data D

  where $E_D(W)=1/2\Sigma_n(d(n)-o(n))^2$

Stochastic gradient calculates the gradient using a single instanc

- Stochastic Gradient descent
  - $w(k+1)=w(k) - \eta\nabla E_n(W)$ over individual training examples
  - $E_n(W)=1/2\ (d(n)-o(n))^2$

Question 7 [1.5 pts] Figure 2 shows two groups of instances (green vs. red) which belongs to class C_1 and C_2, respectively. Assume a neuron with weight values $[w_0, w_1, w_2]$ is used to learn decision surface to separate the two group instances (w_0 is the weight value for bias),

- Draw decision surfaces corresponding to [1, 1, 1], [-1, 1, 1], and [0, 1, -1], respectively [0.5 pt] (mark each line on the plot) [0.5 pt]

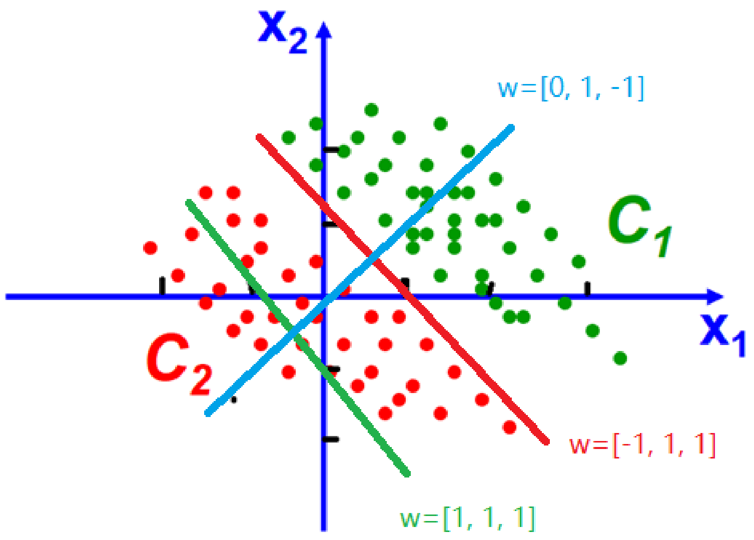


Figure 2

- Explain how does each weight values [w_0, w_1, w_2] control the decision surface, respectively [0.5 pt]

W1 and W2 determine the slope of the decision surface.

W0 determines the intercept of the decision surface.

By adjusting W0, W1, W2 values, we can adjust decision surface to separate instances into groups.

- Among the three decision surfaces, which line is the best decision surface to separate instances, why? [0.t pt]

Green line w=[-1, 1, 1] is the best decision surface, because all green dots (C1) are above the line, whereas all red dots (C2) are underneath the line. For other two lines (red and blue), there are misclassifications.

Question 8 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming the class lable for C1 and C2 are 1 and 0, respectively, the learning rate η=0.5, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use perceptron learning rule to learn a linear decision surface for these two classes. List the results in the first two rounds by using tables in the following form. [show your calculations.]

Assume the activation is defined as follow.

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & otherwise \end{cases}$$

The First Round

Input	Weight	v	Desired	Actual	New Weight
(1,-1,-1)	**(1,1,1)**	**-1**	**0**	**0**	**(1,1,1)**
(1,1,1)	**(1,1,1)**	**3**	**1**	**1**	**(1,1,1)**
(1,0,-1)	**(1,1,1)**	**0**	**1**	**1**	**(1,1,1)**
(1,0,1)	**(1,1,1)**	**2**	**0**	**1**	**(0.5,1,0.5)**
(1,-1,0)	**(0.5,1,0.5)**	**-0.5**	**0**	**0**	**(0.5,1,0.5)**
(1,1,0)	**(0.5,1,0.5)**	**1.5**	**1**	**1**	**(0.5,1,0.5)**

The Second Pass

Input	Weight	v	Desired	Actual	New Weight
(1,-1,-1)	**(0.5,1,0.5)**	**-1**	**0**	**0**	**(0.5,1,0.5)**
(1,1,1)	**(0.5,1,0.5)**	**2**	**1**	**1**	**(0.5,1,0.5)**
(1,0,-1)	**(0.5,1,0.5)**	**0**	**1**	**1**	**(0.5,1,0.5)**
(1,0,1)	**(0.5,1,0.5)**	**1**	**0**	**1**	**(0,1,0)**
(1,-1,0)	**(0,1,0)**	**-1**	**0**	**0**	**(0,1,0)**
(1,1,0)	**(0,1,0)**	**1**	**1**	**1**	**(0,1,0)**

The network weight values after the send pass is [0, 1, 0]

Question 9 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming the class lable for C1 and C2 are 1 and 0, respectively, the learning rate η=0.2, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use gradient descent learning rule to learn a linear decision surface for these two classes. [show your calculations.]

- List the results in the first two rounds by using tables in the following form
- Please also report the mean squared errors E(W) after the weight updating for each round.

The First Round

Input	Weight	v	Desired	Output	ΔW
(1,-1,-1)					
(1,1,1)					
(1,0,-1)					
(1,0,1)					
(1,-1,0)					
(1,1,0)					

The Second Round

Input	Weight	v	Desired	Output	ΔW
(1,-1,-1)					
(1,1,1)					
(1,0,-1)					
(1,0,1)					
(1,-1,0)					
(1,1,0)					

$V=W^T \times X$ (which is the local field)

"Actual" is the same as v, because there is no activation function.

Initial $E(W)=((0+1)^2+(1-3)^2+(1-0)^2+(0-2)^2+(0-0)^2+(1-2)^2)/2 = 11/2 = 5.5$

The First Round

Input	Weight	v	Desired	Actual	ΔW
(1,-1,-1)	(1,1,1)	-1	0	-1	**(0.2,-0.2,-0.2)**
(1,1,1)	(1,1,1)	3	1	3	**(-0.4,-0.4,-0.4)**
(1,0,-1)	(1,1,1)	0	1	0	**(0.2,0,-0.2)**
(1,0,1)	(1,1,1)	2	0	2	**(-0.4,0,-0.4)**
(1,-1,0)	(1,1,1)	0	0	0	**(0,0,0)**
(1,1,0)	(1,1,1)	2	1	2	**(-0.2,-0.2,0)**

The total delta weight values is: $\Delta w=(-0.6, -0.8, -1.2)$
Therefore, the new weight values $W(K+1)=W(K)+\Delta w=(1, 1, 1) + (-0.6, -0.8, -1.2)$
$W=(0.4, 0.2, -0.2)$

E(W) after the first round weight updating is

$((0-0.4)^2+(1-0.4)^2+(1-0.6)^2+(0-0.2)^2+(0-0.2)^2+(1-0.6)^2)/2$

$=(0.16+0.36+0.16+0.04+0.04+0.16)/2=0.92/2=0.46$

The mean squared error is 0.46/6=0.076

The Second Round

Input	Weight	v	Desired	Actual	ΔW
(1,-1,-1)	(0.4,0.2,-0.2)	0.4	0	0.4	**(-0.08,0.08,0.08)**
(1,1,1)	(0.4,0.2,-0.2)	0.4	1	0.4	**(0.12,0.12,0.12)**
(1,0,-1)	(0.4,0.2,-0.2)	0.6	1	0.6	**(0.08,0,-0.08)**
(1,0,1)	(0.4,0.2,-0.2)	0.2	0	0.2	**(-0.04,0,-0.04)**
(1,-1,0)	(0.4,0.2,-0.2)	0.2	0	0.2	**(-0.04,0.04,0)**
(1,1,0)	(0.4,0.2,-0.2)	0.6	1	0.6	**(0.08,0.08,0)**

The total delta weight values is: Δw=(0.12,0.32,0.08)

Therefore, the new weight values W(K+1)=W(K)+ Δw=(0.4, 0.2, -0.2) + (0.12,0.32,0.08)

W=(0.52, 0.52, -0.12)

E(W) after the first round weight updating is

$((0-0.12)^2+(0.92-1)^2+(0.64-1)^2+(0-0.4)^2+(0-0)^2+(1-1.04)^2)/2$

$=(0.0144+0.0064+0.1296+0.16+0+0.0016)/2=0.312/2=0.156$

The mean squared error is 0.156/6=0.026

The Third Round (we need this table to calculate the E(W) after the 2nd round weight updating).

Input	Weight	v	Desired	Actual	ΔW
(1,-1,-1)	(0.52, 0.52, -0.12)	0.12	0	0.12	
(1,1,1)	(0.52, 0.52, -0.12)	0.92	1	0.92	
(1,0,-1)	(0.52, 0.52, -0.12)	0.64	1	0.64	
(1,0,1)	(0.52, 0.52, -0.12)	0.4	0	0.4	
(1,-1,0)	(0.52, 0.52, -0.12)	0	0	0	
(1,1,0)	(0.52, 0.52, -0.12)	1.04	1	1.04	

The above results show that E(W) is continuously decreasing from 5.5, to 0.46, then to 0.156

Question 10 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming the class lable for C1 and C2 are 1 and 0, respectively, the learning rate η=0.2, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use Delta learning rule (AdaLine) to learn a linear decision surface for these two classes. [show your calculations.]

- List the results in the first round by using table in the following form
- Please also report the mean squared errors E(W) after the weight updating for the first round.

The First Round

Input	Weight	v	Desired	Output	ΔW
(1,-1,-1)					
(1,1,1)					
(1,0,-1)					
(1,0,1)					
(1,-1,0)					
(1,1,0)					

Input	Weight	v	Desired	Output	ΔW
(1,-1,-1)	(1,1,1)	-1	0	-1	(0.2, -0.2, -0.2)
(1,1,1)	(1.2, 0.8, 0.8)	2.8	1	2.8	(-0.36, -0.36, -0.36)
(1,0,-1)	(0.84, 0.44, 0.44)	0.4	1	0.4	(0.12, 0, -0.12)
(1,0,1)	(0.96, 0.44, 0.32)	1.28	0	1.28	(-0.256, 0, -0.256)
(1,-1,0)	(0.704, 0.44, 0.064)	0.264	0	0.264	(-0.0528, 0.0528, 0)
(1,1,0)	(0.6512, 0.4928, 0.064)	1.144	1	1.144	(-0.0288,-0.0288,0)

Final Weight: (0.6224, 0.464,0.064)= (0.6512, 0.4928, 0.064)+ (-0.0288,-0.0288,0)

In order to calculate mean squared errors after the first round, we have to use the new weight (0.6224, 0.464,0.064) to calculate output of all instances, and then find mean squared errors.

Input	Weight	v	Desired	Output	
(1,-1,-1)	(0.6224, 0.464,0.064)	**0.094**	**0**	**0.094**	
(1,1,1)	(0.6224, 0.464,0.064)	**1.150**	**1**	**1.150**	
(1,0,-1)	(0.6224, 0.464,0.064)	**0.558**	**1**	**0.558**	
(1,0,1)	(0.6224, 0.464,0.064)	**0.686**	**0**	**0.686**	
(1,-1,0)	(0.6224, 0.464,0.064)	**0.158**	**0**	**0.158**	
(1,1,0)	(0.6224, 0.464,0.064)	**1.086**	**1**	**1.086**	

Therefore, total squared Error is:

$E(W)=\{(0-0.094)^2+(1-1.15)^2+(1-0.558)^2+(0-0.686)^2+(0-0.158)^2+(1+1.086)^2\}/2=$ **0.365**

The mean squared error is

0.365/6=0.0608

Programming Tasks: For all programming tasks, please submit the Notebook (or Markdown) as html files for grading (**the notebook must include scrips/code and the results of the script**)

Question 11 [3 pts]

Class1.txt and Class2.txt files in the Canvas contain two-dimensional instances in two classes (C_1 and C_2 respectively, with 100 instances in each class). In these two files, the first column denotes the *x* values and the second column represents the *y* values (separated by comma). Please use skeleton code in Gradient Descent Learning [Notebook, pdf] in Canvas to implement following tasks

- Combine instances in the Class1.txt and Class2.txt files as one dataframe, and label instances in Class1 as 2, and instances in Class2 as -2. [0.25 pt]
- Report the scatter plot of all 200 instances in the same plot, using "Blue" color to show instances in Class1 and uses "Green" to show instances in Class 2 [0.25 pt]
- Randomly select 80% instances from class1.txt and 80% instances from class2.txt to train a perceptron classifier (using gradient descent learning rule), and use the classifier to classify remaining 20% instances in class1.txt and class2.txt. Report the classification accuracy of the perceptron classifier on the 20% test instances (using learning rate=1/(# of training samples), error threshold 0.1, and iteration numbers 2000) [0.5 pt]
- Report training errors and test errors of the perceptron classifier with respect to each iteration. Show the two error rates on the same chart, where the x-axis denotes the iteration and the y-axis denotes the mean classification errors [0.5 pt]
- Report the final decision surface on the same scatter plot which shows the 200 instances [0.5 pt]
- Please use Delta rule function in the Gradient Descent Learning [Notebook, pdf] to learn from the same 80% of training samples (using learning rate =0.01, error threshold 0.1, and iteration numbers 2000).

- Report the training errors as a plot, where the x-axis denotes the iteration and the y-axis denotes the mean classification errors [0.5 pt]
- Compare Delta learing rule and Gradient Descent Learning rule, explain the advantage and disadvantage of each of them, respectively [0.5 pt]

Question 12 [1.5 pts]

Please download review100.csv dataset from Canvas, and use a programming language (Python, R, etc.) to implement tasks below. The review100.csv includes 100 movie reviews (from IMDB) and the sentiment (positive vs. negative) of the reviewer (there are 50 positive reviews and 50 negative reviews). Each review (each row) contains two parts. The first column is the reviews (text), and the second column is the sentiment (positive or negative).

- Read reviews from the review100.csv. Tokenize each review using space key, so each review is represented as a set of tokens (words). Use the top 1,000 most frequent tokens (words) as features to represent each review (so each review is represented as an instance with 1000 features). The value of each feature is 1 if the view has the corresponding token/word, or 0 otherwise. Use .head() to show the first several rows of the data frame. [0.5 pt]
- Use 1 as the label of the positive review, and 0 as the label of the negative review (so each review is represented using 1,000 features and a class label). Use .head() to show the first several rows of the data frame. [0.5 pt]
- Calculate conditional entropy and information gain of each token/word, and report the top-50 tokens/words with the highest information gain values (as a bar plot, where x-axis shows the word and y-axis shows the information gain values) [0.5 pt]