Study Partner: Kevin Jungho Lee

2 Overfitting

Overfitting is a common problem when doing datascience work.

(a) How can you tell if a model you have trained is overfitting?

(b) Why would it be bad to deploy a model that's been overfitted to your training data?

(c) Briefly describe how overfitting can occur.

(d) In lecture and discussion we've discussed multiple methods for dealing with overfitting. One such technique was regularization. Please describe two different regularization techniques and explain how they help to mitigate overfitting.

(a) If a model is performing very well on the training set, but very poorly on the test set, then the model is overfitting.

(b) If a model is overfitting, then it performs badly on the test set, which is a representation/subset of what the model would encounter in the real world (i.e. data the model has never seen).

(c) Overfitting happens when the model is very complex (e.g. has many parameters) and there is not much training data. In this case, our model can easily capture the bias of the training data and minimize the training error. Thus, it fails to generalize the model to test data.

(d)

1. L1 regularization

An example of L1 regularization is Lasso regression. Recall linear regression has cost function as follows:

$$\sum_{i=1}^{n}(y^{(i)} - \sum_{j=1}^{m} x^{(i)}{}_j w_j)^2$$

What L1 regularization does is to add penalty for the weights being too big. This prevents the model from being complex. The cost function of Lasso regression is follows:

$$\sum_{i=1}^{n}(y^{(i)} - \sum_{j=1}^{m} x^{(i)}{}_j w_j)^2 + \lambda \sum_{j=1}^{m} |w_j|$$

Note as the weights get bigger, the cost also increases. Thus, this cost function ensures that weights don't get too big. λ here is regularization term, where big λ means more regularization, small λ means only a little regularization.

2. L2 regularization

An example of L2 regularization is Ridge regression. Similar to L1 regression except the cost function of ridge regression is as follows:

$$\sum_{i=1}^{n}(y^{(i)} - \sum_{j=1}^{m} x^{(i)}{}_j w_j)^2 + \lambda \sum_{j=1}^{m} w^2{}_j$$

3 Overfitting Mitigations

For each of the below strategies, state whether or not it might help to mitigate overfitting and explain why.

(a) Using a smaller training dataset

(b) Restricting the maximum value any parameter can take on

(c) Training your neural network for longer (more iterations)

(d) Training a model with more parameters

(e) Randomly setting the outputs of 50% of the nodes in your neural network to zero

(f) Incorporating additional sparse features into your model

(g) Initializing your parameters randomly instead of to zero

(h) Training your model on a graphics processing unit or specialized accelerator chip instead of a CPU

(a) This would make it worse. The more training data we have, the harder the model can overfit the training data because there isn't likely a good set of parameters that work well for many data points.

(b) This would help because it's kind of what L1/L2 regularization does. Consider the linear regression model. Recall the cost function is

$$\sum_{i=1}^{n}(y^{(i)} - \sum_{j=1}^{m} x^{(i)}{}_j w_j)^2$$

Originally there was no constraint on the values of the weights. However, when we add max constraint on them, then the model can only express a less complex model. It can no longer express complicated models.

(c) This either won't change anything or even make it even overfit. The more iteration we train our model, the less the train error becomes. Thus, it overfits the training set.

(d) This would make it worse because more parameters mean that the algorithm can represent an even more complex model.

(e) This would help. This technique is known as dropout, a common regularization technique used for neural networks. The idea is that this enforces our network from relying too much on any particular node, and thus avoid overfitting.

(f) This doesn’t help much. Adding sparse features means adding features where most of the labels are 0 while only a few are 1. Notice although the dimension of our data increased, since most of the data in the new dimension is just 0, there is not much information added to the model.

(g) It depends. If we find a better local minimum, it would make it worse. Otherwise, if we find a worse local minimum, it helps because now the model is less fitting to the training set.

(h) This doesn’t change anything because it just speeds up the training time.

4 K-Nearest Neighbors

(a) Why is it important to normalize your data when using the k-nearest neighbors algorithm (KNN)?

(b) When doing k-nearest neighbors, an odd value for k is typically used. Why is this?

(c) Say you have the dataset in Table 1, where x and y are features and L is the label. Normalize the features by scaling them so that all values in a column lie in the range $[0, 1]$, so that they can be used with KNN.

(d) Use KNN with K=3 to make predictions for the data points in Table 2.

(e) Use KNN with $k = 1$ to make predictions for the data points in Table 2.

(f) Use KNN with $k = 5$ to make predictions for the data points in Table 2.

(g) What is a potential issue with using a low k for KNN?

(h) What is a potential issue with selecting a k that is too high when doing KNN?

(a) We want the model to treat each feature equally. Otherwise, the distance between two data points will mostly depend on the features with large values.

(b) We want to break the tie

(c) Normalizing data by dividing each column by its max. Note max(x) = 1.1, max(y) = 760, and max(L) = 1.

x	y	L
0.17	0.46	0
0.08	0.99	0
0.25	0.92	0
0.08	0.66	0
0.17	0.66	0
1	0.53	1
0.75	0.54	1
0.92	0.51	1
0.08	1	1

(d) k=3

Before Normalization

x	y	L_pred
0.1	760	
0.2	700	
0.6	200	

After Normalization (Note max(x) = 1.1, max(y) = 760, and max(L) = 1 from training set)

x	y	L_pred
0.09	1	
0.18	0.92	
0.55	0.26	

Distance to each data point in training set from x=0.09, y=1

(x - x_train)^2	(y - y_train)^2	distance	label
(0.09 - 0.17)^2=0.0064	(1 - 0.46)^2=0.2916	sqrt(0.0064+0.2916)=0.54589	0
(0.09 - 0.08)^2=0.0001	(1 - 0.99)^2=0.0001	sqrt(0.0001+0.0001)=0.014142	0
(0.09 - 0.25)^2=0.0256	(1 - 0.92)^2=0.0064	sqrt(0.0256+0.0064)=0.178885	0
(0.09 - 0.08)^2=0.0001	(1 - 0.66)^2=0.1156	sqrt(0.0001+0.1156)=0.34014	0
(0.09 - 0.17)^2=0.0064	(1 - 0.66)^2=0.1156	sqrt(0.0064+0.1156)=0.34928	0
(0.09 - 1)^2=0.8281	(1 - 0.53)^2=0.2209	sqrt(0.8281+0.2209)=1.0242	1
(0.09 - 0.75)^2=0.4356	(1 - 0.54)^2=0.2116	sqrt(0.4356+0.2116)=0.804487	1
(0.09 - 0.92)^2=0.6889	(1 - 0.51)^2=0.2401	sqrt(0.6889+0.2401)=0.9638	1
(0.09 - 0.08)^2=0.0001	(1 - 1)^2=0	sqrt(0.0001+0)=0.01	1

Distance to each data point in training set from x=0.18, y=0.92

(x - x_train)^2	(y - y_train)^2	distance	label
(0.18 - 0.17)^2=0.0001	(0.92 - 0.46)^2=0.2116	sqrt(0.0001+0.2116)=0.4601	0
(0.18 - 0.08)^2=0.01	(0.92 - 0.99)^2=0.0049	sqrt(0.01+0.0049)=0.12206	0
(0.18 - 0.25)^2=0.0049	(0.92 - 0.92)^2=0	sqrt(0.0049+0)=0.07	0
(0.18 - 0.08)^2=0.01	(0.92 - 0.66)^2=0.0676	sqrt(0.01+0.0676)=0.27856	0
(0.18 - 0.17)^2=0.0001	(0.92 - 0.66)^2=0.0676	sqrt(0.0001+0.0676)=0.2601	0
(0.18 - 1)^2=0.6724	(0.92 - 0.53)^2=0.1521	sqrt(0.6724+0.1521)=0.90801	1
(0.18 - 0.75)^2=0.3249	(0.92 - 0.54)^2=0.1444	sqrt(0.3249+0.1444)=0.6850	1
(0.18 - 0.92)^2=0.5476	(0.92 - 0.51)^2=0.1681	sqrt(0.5476+0.1681)=0.84599	1
(0.18 - 0.08)^2=0.01	(0.92 - 1)^2=0.0064	sqrt(0.01+0.0064)=0.1280	1

Distance to each data point in training set from x=0.55, y=0.26

(x - x_train)^2	(y - y_train)^2	distance	label
(0.55 - 0.17)^2=0.1444	(0.26 - 0.46)^2=0.04	sqrt(0.1444+0.04)=0.4294	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.99)^2=0.5329	sqrt(0.2209+0.5329)=0.8682	0
(0.55 - 0.25)^2=0.09	(0.26 - 0.92)^2=0.4356	sqrt(0.09+0.4356)=0.72498	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.66)^2=0.16	sqrt(0.2209+0.16)=0.61717	0
(0.55 - 0.17)^2=0.1444	(0.26 - 0.66)^2=0.16	sqrt(0.1444+0.16)=0.5517	0
(0.55 - 1)^2=0.2025	(0.26 - 0.53)^2=0.0729	sqrt(0.2025+0.0729)=0.52478	1
(0.55 - 0.75)^2=0.04	(0.26 - 0.54)^2=0.0784	sqrt(0.04+0.0784)=0.34409	1
(0.55 - 0.92)^2=0.1369	(0.26 - 0.51)^2=0.0625	sqrt(0.1369+0.0625)=0.44654	1
(0.55 - 0.08)^2=0.2209	(0.26 - 1)^2=0.5476	sqrt(0.2209+0.5476)=0.8766	1

Thus the predictions are follows:

x	y	L_pred
0.1	760	0
0.2	700	0
0.6	200	1

(e) k=1

Use the results from the previous question.

Distance to each data point in training set from x=0.09, y=1

(x - x_train)^2	(y - y_train)^2	distance	label
(0.09 - 0.17)^2=0.0064	(1 - 0.46)^2=0.2916	sqrt(0.0064+0.2916)=0.54589	0
(0.09 - 0.08)^2=0.0001	(1 - 0.99)^2=0.0001	sqrt(0.0001+0.0001)=0.014142	0
(0.09 - 0.25)^2=0.0256	(1 - 0.92)^2=0.0064	sqrt(0.0256+0.0064)=0.178885	0
(0.09 - 0.08)^2=0.0001	(1 - 0.66)^2=0.1156	sqrt(0.0001+0.1156)=0.34014	0
(0.09 - 0.17)^2=0.0064	(1 - 0.66)^2=0.1156	sqrt(0.0064+0.1156)=0.34928	0
(0.09 - 1)^2=0.8281	(1 - 0.53)^2=0.2209	sqrt(0.8281+0.2209)=1.0242	1
(0.09 - 0.75)^2=0.4356	(1 - 0.54)^2=0.2116	sqrt(0.4356+0.2116)=0.804487	1
(0.09 - 0.92)^2=0.6889	(1 - 0.51)^2=0.2401	sqrt(0.6889+0.2401)=0.9638	1
(0.09 - 0.08)^2=0.0001	(1 - 1)^2=0	sqrt(0.0001+0)=0.01	1

Distance to each data point in training set from x=0.18, y=0.92

(x - x_train)^2	(y - y_train)^2	distance	label
(0.18 - 0.17)^2=0.0001	(0.92 - 0.46)^2=0.2116	sqrt(0.0001+0.2116)=0.4601	0
(0.18 - 0.08)^2=0.01	(0.92 - 0.99)^2=0.0049	sqrt(0.01+0.0049)=0.12206	0
(0.18 - 0.25)^2=0.0049	(0.92 - 0.92)^2=0	sqrt(0.0049+0)=0.07	0

(0.18 - 0.08)^2=0.01	(0.92 - 0.66)^2=0.0676	sqrt(0.01+0.0676)=0.27856	0
(0.18 - 0.17)^2=0.0001	(0.92 - 0.66)^2=0.0676	sqrt(0.0001+0.0676)=0.2601	0
(0.18 - 1)^2=0.6724	(0.92 - 0.53)^2=0.1521	sqrt(0.6724+0.1521)=0.90801	1
(0.18 - 0.75)^2=0.3249	(0.92 - 0.54)^2=0.1444	sqrt(0.3249+0.1444)=0.6850	1
(0.18 - 0.92)^2=0.5476	(0.92 - 0.51)^2=0.1681	sqrt(0.5476+0.1681)=0.84599	1
(0.18 - 0.08)^2=0.01	(0.92 - 1)^2=0.0064	sqrt(0.01+0.0064)=0.1280	1

Distance to each data point in training set from x=0.55, y=0.26

(x - x_train)^2	(y - y_train)^2	distance	label
(0.55 - 0.17)^2=0.1444	(0.26 - 0.46)^2=0.04	sqrt(0.1444+0.04)=0.4294	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.99)^2=0.5329	sqrt(0.2209+0.5329)=0.8682	0
(0.55 - 0.25)^2=0.09	(0.26 - 0.92)^2=0.4356	sqrt(0.09+0.4356)=0.72498	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.66)^2=0.16	sqrt(0.2209+0.16)=0.61717	0
(0.55 - 0.17)^2=0.1444	(0.26 - 0.66)^2=0.16	sqrt(0.1444+0.16)=0.5517	0
(0.55 - 1)^2=0.2025	(0.26 - 0.53)^2=0.0729	sqrt(0.2025+0.0729)=0.52478	1
(0.55 - 0.75)^2=0.04	(0.26 - 0.54)^2=0.0784	sqrt(0.04+0.0784)=0.34409	1
(0.55 - 0.92)^2=0.1369	(0.26 - 0.51)^2=0.0625	sqrt(0.1369+0.0625)=0.44654	1
(0.55 - 0.08)^2=0.2209	(0.26 - 1)^2=0.5476	sqrt(0.2209+0.5476)=0.8766	1

Thus the predictions are follows:

x	y	L_pred
0.1	760	1
0.2	700	0
0.6	200	1

(f) k=5

Use the results from the previous question.

Distance to each data point in training set from x=0.09, y=1

(x - x_train)^2	(y - y_train)^2	distance	label
(0.09 - 0.17)^2=0.0064	(1 - 0.46)^2=0.2916	sqrt(0.0064+0.2916)=0.54589	0
(0.09 - 0.08)^2=0.0001	(1 - 0.99)^2=0.0001	sqrt(0.0001+0.0001)=0.014142	0
(0.09 - 0.25)^2=0.0256	(1 - 0.92)^2=0.0064	sqrt(0.0256+0.0064)=0.178885	0
(0.09 - 0.08)^2=0.0001	(1 - 0.66)^2=0.1156	sqrt(0.0001+0.1156)=0.34014	0
(0.09 - 0.17)^2=0.0064	(1 - 0.66)^2=0.1156	sqrt(0.0064+0.1156)=0.34928	0
(0.09 - 1)^2=0.8281	(1 - 0.53)^2=0.2209	sqrt(0.8281+0.2209)=1.0242	1
(0.09 - 0.75)^2=0.4356	(1 - 0.54)^2=0.2116	sqrt(0.4356+0.2116)=0.804487	1
(0.09 - 0.92)^2=0.6889	(1 - 0.51)^2=0.2401	sqrt(0.6889+0.2401)=0.9638	1
(0.09 - 0.08)^2=0.0001	(1 - 1)^2=0	sqrt(0.0001+0)=0.01	1

Distance to each data point in training set from x=0.18, y=0.92

(x - x_train)^2	(y - y_train)^2	distance	label
(0.18 - 0.17)^2=0.0001	(0.92 - 0.46)^2=0.2116	sqrt(0.0001+0.2116)=0.4601	0
(0.18 - 0.08)^2=0.01	(0.92 - 0.99)^2=0.0049	sqrt(0.01+0.0049)=0.12206	0
(0.18 - 0.25)^2=0.0049	(0.92 - 0.92)^2=0	sqrt(0.0049+0)=0.07	0
(0.18 - 0.08)^2=0.01	(0.92 - 0.66)^2=0.0676	sqrt(0.01+0.0676)=0.27856	0
(0.18 - 0.17)^2=0.0001	(0.92 - 0.66)^2=0.0676	sqrt(0.0001+0.0676)=0.2601	0
(0.18 - 1)^2=0.6724	(0.92 - 0.53)^2=0.1521	sqrt(0.6724+0.1521)=0.90801	1
(0.18 - 0.75)^2=0.3249	(0.92 - 0.54)^2=0.1444	sqrt(0.3249+0.1444)=0.6850	1
(0.18 - 0.92)^2=0.5476	(0.92 - 0.51)^2=0.1681	sqrt(0.5476+0.1681)=0.84599	1
(0.18 - 0.08)^2=0.01	(0.92 - 1)^2=0.0064	sqrt(0.01+0.0064)=0.1280	1

Distance to each data point in training set from x=0.55, y=0.26

(x - x_train)^2	(y - y_train)^2	distance	label
(0.55 - 0.17)^2=0.1444	(0.26 - 0.46)^2=0.04	sqrt(0.1444+0.04)=0.4294	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.99)^2=0.5329	sqrt(0.2209+0.5329)=0.8682	0
(0.55 - 0.25)^2=0.09	(0.26 - 0.92)^2=0.4356	sqrt(0.09+0.4356)=0.72498	0
(0.55 - 0.08)^2=0.2209	(0.26 - 0.66)^2=0.16	sqrt(0.2209+0.16)=0.61717	0
(0.55 - 0.17)^2=0.1444	(0.26 - 0.66)^2=0.16	sqrt(0.1444+0.16)=0.5517	0
(0.55 - 1)^2=0.2025	(0.26 - 0.53)^2=0.0729	sqrt(0.2025+0.0729)=0.52478	1
(0.55 - 0.75)^2=0.04	(0.26 - 0.54)^2=0.0784	sqrt(0.04+0.0784)=0.34409	1
(0.55 - 0.92)^2=0.1369	(0.26 - 0.51)^2=0.0625	sqrt(0.1369+0.0625)=0.44654	1
(0.55 - 0.08)^2=0.2209	(0.26 - 1)^2=0.5476	sqrt(0.2209+0.5476)=0.8766	1

Thus the predictions are follows:

x	y	L_pred
0.1	760	0
0.2	700	0
0.6	200	1

(g) Using small k means that noise will have a higher influence on the result. (i.e. overfitting)

(h) Using large k means that it could be influenced by things that are not very close in terms of distance. Eventually when k is at maximum, then the algorithm simply returns the majority class. (i.e. underfitting)

x	y	L
0.2	350	0
0.1	750	0
0.3	700	0
0.1	500	0
0.2	500	0
1.2	400	1
0.9	410	1
1.1	390	1
0.1	760	1

Table 1: Raw KNN data for training

x	y	L_{pred}
.1	760	
.2	700	
.6	200	

Table 2: Raw KNN data for evaluation

(i) Say you had a dataset and wanted to understand how well KNN with a k of 3 performed on it. How could you quantify its performance? Assume you do not have access to any samples beyond those in your dataset.

(j) Say you had a dataset consisting of 200 samples. How might you select the optimal k to use for KNN?

(i) One way to measure the performance is to split the dataset into training set and test set. Then we train our model with k=3 on the training set and see the performance on the test set. However, since we don't have much dataset (we have only 9 tuples!), it's probably a good idea to use cross validation and check the average of the test performance. Finally, you can quantify the test score using accuracy, precision, recall, or F1 score depending on the domain of the problem (i.e. whether you care about false positive or false negative or both or neither)

(j) We perform the step taken at (i) for different values of k. We pick the k that gives us the best test set performance.

5 Principal Component Analysis

For each of the below situations, state whether or not PCA would work well, and briefly explain why.

(a) Data with a linear structure

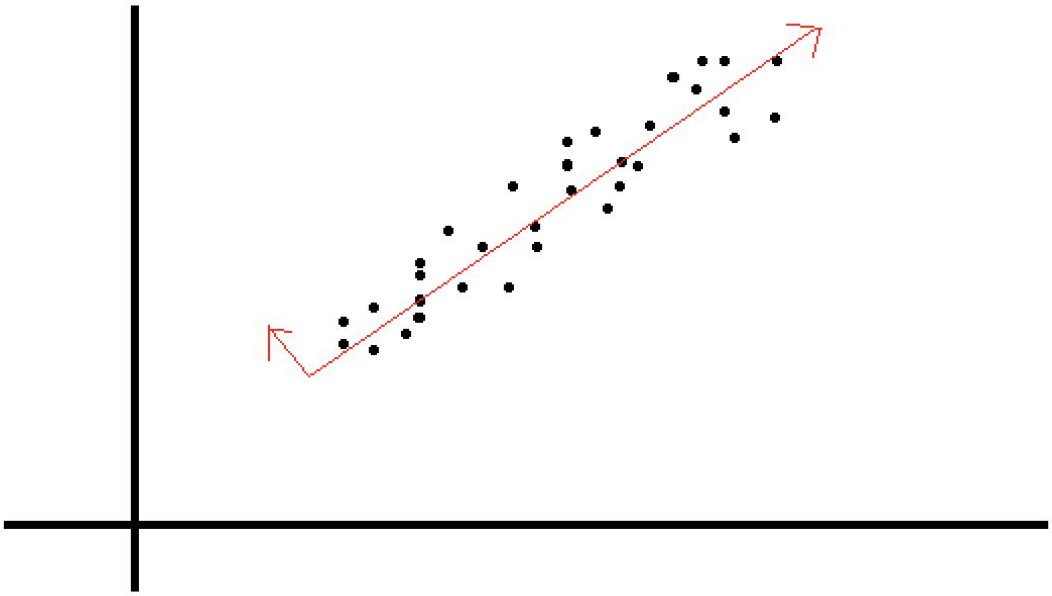
(b) Data lying on a hyperbolic plane

(c) A dataset containing non-normalized features

(d) A dataset where each feature is statistically independent of all others

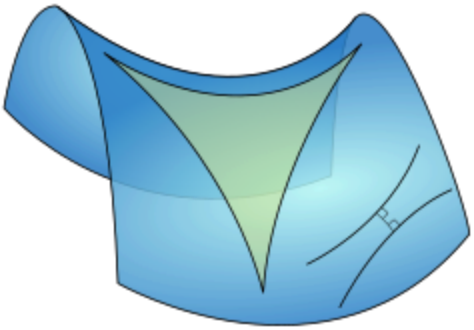
(a)

It will work well. For example, consider the 2D dataset below with linear structure. Then PCA would find the red axes shown below as principal components.

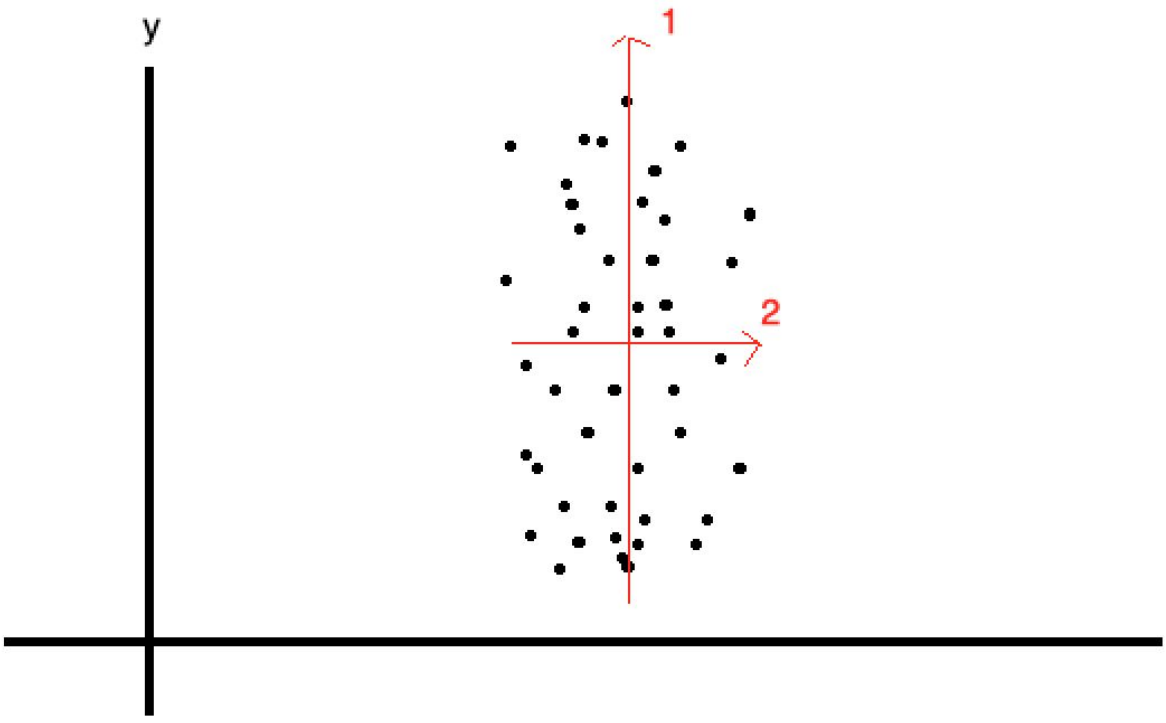


(b)

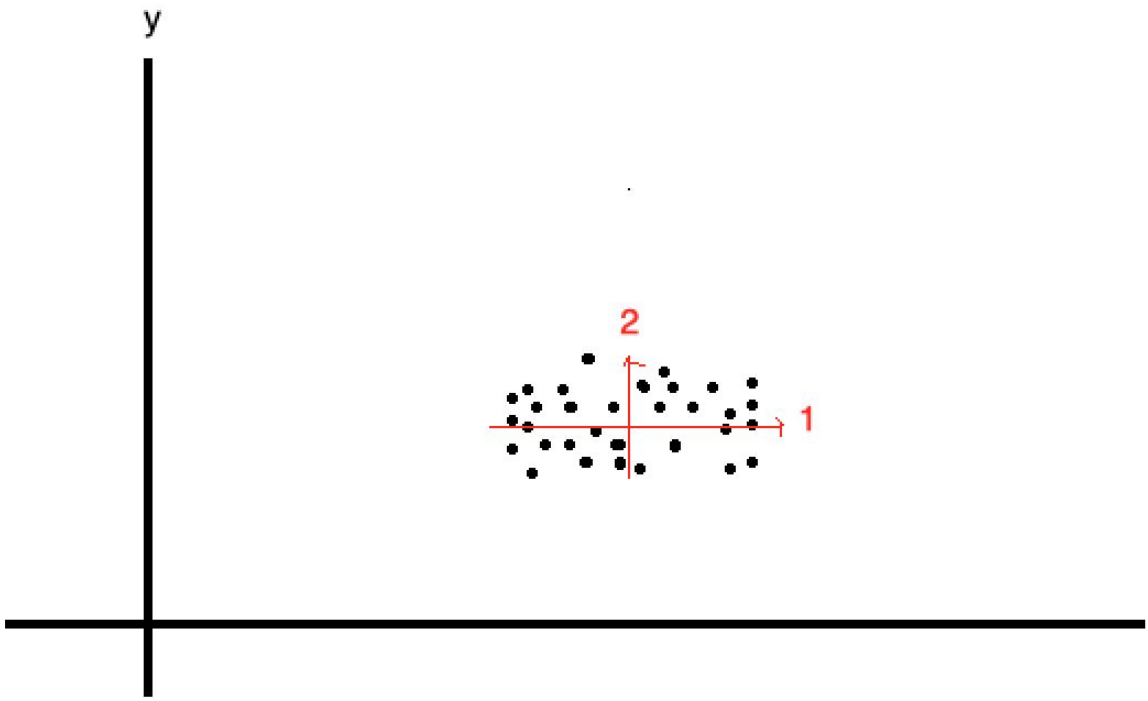
It would not work. Consider the hyperbolic plane in 3D below. PCA would not be able to find nice principal components because every principal component must be orthogonal to each other.



(c)
It would not work as expected. Consider the 2D dataset below with y features non-normlized.

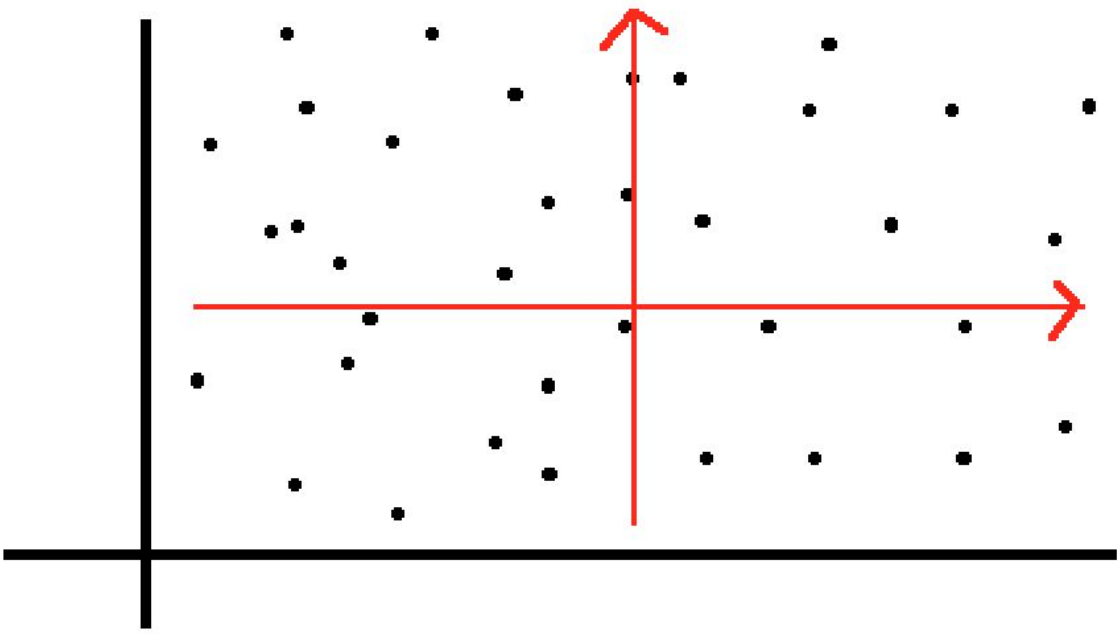


It may see that y axis is the first principal component and x axis is the second principal component. Let's say when we normalize values in y-axis, it looks as follows:



Now x axis is the first principal component and y axis is the second principal component.

(d) It would not work. The idea of PCA is to transform correlated variables into a set of values of linearly uncorrelated variables called principal components. Consider the dataset that is uncorrelated as follows:



Then principal components are just the same as the original axises.

6 Artificial Neural Networks

Consider the following computation graph for a simple neural network for binary classification. Here x_1 and x_2 are input features and y is their associated class. The network has multiple parameters, including weights w and biases b, as well as non-linearity function g. The network will output a value y_{pred}, representing the probability that a sample belongs to class 1. We use a loss function $Loss$ to help train our model. The network is initialized with the parameters in Table 3.

You will first train the model using some sample datapoints and then evaluate its performance. For any questions that ask for performance metrics, generate them using the samples in Table 4.

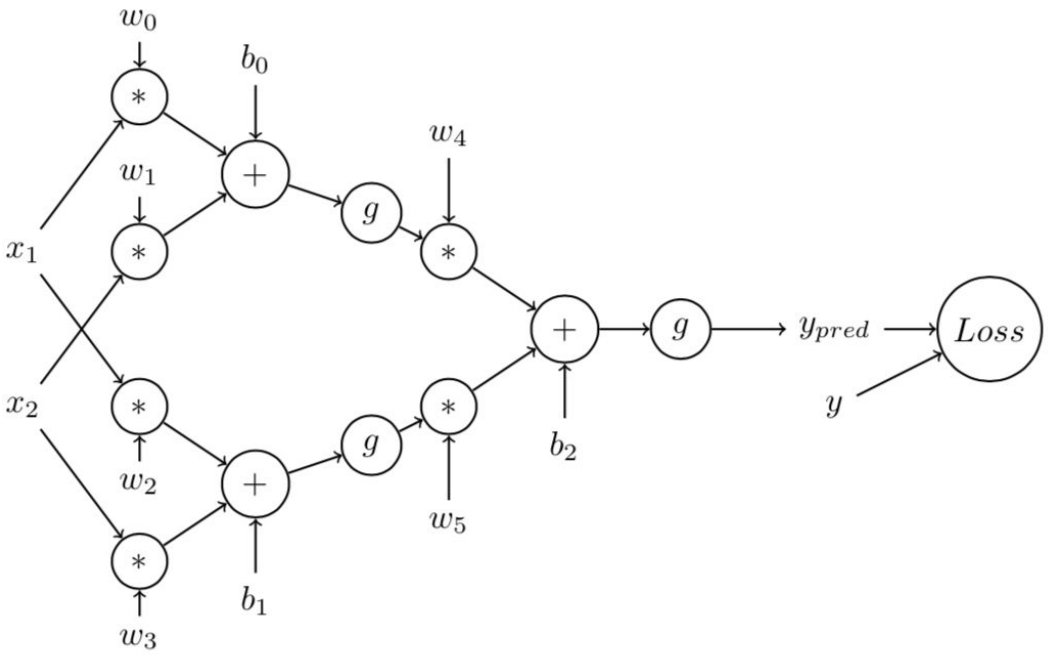


Figure 1: Neural network architecture

(a) Initialize the neural network with the parameters in Table 3 and then train it using the samples in Table 4. Use gradient descent (forward pass followed by backpropagation) to update the parameters.

Suppose the loss function is quadratic, $Loss(y, y_{pred}) = \frac{1}{2}(y - y_{pred})^2$, and g is the sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ (note: it's typically better to use a different type of loss, cross-entropy, for classification problems, but using a quadratic loss function keeps the math easier for this assignment).

Assume that your learning rate $\alpha = .1$.

Pass the samples through the network one at a time in order from top to bottom. Report the final values of all parameters. Show all work.

(b) What is your model's accuracy?

(c) What is your model's precision?

(d) What is your model's recall?

(e) What is your model's F_1 score?

(a)

Forward propagation (x1=0, x2=0, y=0)

Current parameters

b_0	1
b_1	-6
b_2	-3.93
w_0	3
w_1	4
w_2	6
w_3	5
w_4	2
w_5	4

Note

$$h_{11} = g(w_0 x_1 + w_1 x_2 + b_0)$$
$$h_{12} = g(w_2 x_1 + w_3 x_2 + b_1)$$
$$y_{pred} = g(w_4 h_{11} + w_5 h_{12} + b_2)$$
$$L = \tfrac{1}{2}(y - y_{pred})^2$$
$$g(z) = \tfrac{1}{1+e^{-z}}$$

Since x1=0 and x2=0,

h11	0.7310585786
h12	0.002472623157
y_pred	0.07885604513

Back propagation (x1=0, x2=0, y=0)

Now we want to back propagate the error.

Note

$$g'(z) = g(z)(1 - g(z))$$
$$L = \tfrac{1}{2}(y - y_{pred})^2$$
$$\frac{dL}{dy_{pred}} = -(y - y_{pred}) = y_{pred} - y$$
$$\frac{dL}{db_2} = \frac{dy_{pred}}{db_2}\frac{dL}{dy_{pred}} = y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dw_4} = \frac{dy_{pred}}{dw_4}\frac{dL}{dy_{pred}} = h_{11} y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dw_5} = \frac{dy_{pred}}{dw_5}\frac{dL}{dy_{pred}} = h_{12} y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dh_{11}} = \frac{dy_{pred}}{dh_{11}}\frac{dL}{dy_{pred}} = w_4 y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dh_{12}} = \frac{dy_{pred}}{dh_{12}}\frac{dL}{dy_{pred}} = w_5 y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{db_1} = \frac{dh_{12}}{db_1}\frac{dL}{dh_{12}} = h_{12}(1 - h_{12})\frac{dL}{dh_{12}}$$
$$\frac{dL}{db_0} = \frac{dh_{11}}{db_0}\frac{dL}{dh_{11}} = h_{11}(1 - h_{11})\frac{dL}{dh_{11}}$$
$$\frac{dL}{dw_0} = \frac{dh_{11}}{dw_0}\frac{dL}{dh_{11}} = x_1 h_{11}(1 - h_{11})\frac{dL}{dh_{11}}$$

$$\frac{dL}{dw_1} = \frac{dh_{11}}{dw_1}\frac{dL}{dh_{11}} = x_2 h_{11}(1-h_{11})\frac{dL}{dh_{11}}$$
$$\frac{dL}{dw_2} = \frac{dh_{12}}{dw_2}\frac{dL}{dh_{12}} = x_1 h_{12}(1-h_{12})\frac{dL}{dh_{12}}$$
$$\frac{dL}{dw_3} = \frac{dh_{12}}{dw_3}\frac{dL}{dh_{12}} = x_2 h_{12}(1-h_{12})\frac{dL}{dh_{12}}$$
$$\Delta b_0 = -\alpha\frac{dL}{db_0}$$
$$\Delta b_1 = -\alpha\frac{dL}{db_1}$$
$$\Delta b_2 = -\alpha\frac{dL}{db_2}$$
$$\Delta w_0 = -\alpha\frac{dL}{dw_0}$$
$$\Delta w_1 = -\alpha\frac{dL}{dw_1}$$
$$\Delta w_2 = -\alpha\frac{dL}{dw_2}$$
$$\Delta w_3 = -\alpha\frac{dL}{dw_3}$$
$$\Delta w_4 = -\alpha\frac{dL}{dw_4}$$
$$\Delta w_5 = -\alpha\frac{dL}{dw_5}$$

dL/dy_pred	0.07885604513
dL/db2	0.005727927213
dL/dw4	0.004187450327
dL/dw5	0.000014163000547
dL/dh11	0.01145585443
dL/dh12	0.022911708885
dL/db1	0.00005651194276
dL/db0	0.002252357686
dL/dw0	0
dL/dw1	0
dL/dw2	0
dL/dw3	0
db0	-0.0002252357

	686
db1	-0.000005651194276
db2	-0.000572792721 3
dw0	0
dw1	0
dw2	0
dw3	0
dw4	-0.000418745032 7
dw5	-0.000001416300547

Updated parameters

b0	0.9997747642
b1	-6.000005651
b2	-3.930572793
w0	3
w1	4
w2	6
w3	5
w4	1.999581255
w5	3.999998584

Forward propagation (x1=1, x2=1, y=1)

Current parameters

b0	0.9997747642
b1	-6.000005651

b2	-3.930572793
w0	3
w1	4
w2	6
w3	5
w4	1.999581255
w5	3.999998584

Note

$$h_{11} = g(w_0 x_1 + w_1 x_2 + b_0)$$
$$h_{12} = g(w_2 x_1 + w_3 x_2 + b_1)$$
$$y_{pred} = g(w_4 h_{11} + w_5 h_{12} + b_2)$$
$$L = \tfrac{1}{2}(y - y_{pred})^2$$
$$g(z) = \tfrac{1}{1+e^{-z}}$$

Since x1=1 and x2=1,

h11	0.9996645744
h12	0.9933071115
y_pred	0.885092509

Back propagation (x1=1, x2=1, y=1)

Now we want to back propagate the error.

Note

$$g'(z) = g(z)(1 - g(z))$$
$$L = \tfrac{1}{2}(y - y_{pred})^2$$
$$\frac{dL}{dy_{pred}} = -(y - y_{pred}) = y_{pred} - y$$
$$\frac{dL}{db_2} = \frac{dy_{pred}}{db_2}\frac{dL}{dy_{pred}} = y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{dw_4} = \frac{dy_{pred}}{dw_4}\frac{dL}{dy_{pred}} = h_{11}y_{pred}(1-y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dw_5} = \frac{dy_{pred}}{dw_5}\frac{dL}{dy_{pred}} = h_{12}y_{pred}(1-y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dh_{11}} = \frac{dy_{pred}}{dh_{11}}\frac{dL}{dy_{pred}} = w_4y_{pred}(1-y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{dh_{12}} = \frac{dy_{pred}}{dh_{12}}\frac{dL}{dy_{pred}} = w_5y_{pred}(1-y_{pred})\frac{dL}{dy_{pred}}$$
$$\frac{dL}{db_1} = \frac{dh_{12}}{db_1}\frac{dL}{dh_{12}} = h_{12}(1-h_{12})\frac{dL}{dh_{12}}$$
$$\frac{dL}{db_0} = \frac{dh_{11}}{db_0}\frac{dL}{dh_{11}} = h_{11}(1-h_{11})\frac{dL}{dh_{11}}$$
$$\frac{dL}{dw_0} = \frac{dh_{11}}{dw_0}\frac{dL}{dh_{11}} = x_1h_{11}(1-h_{11})\frac{dL}{dh_{11}}$$
$$\frac{dL}{dw_1} = \frac{dh_{11}}{dw_1}\frac{dL}{dh_{11}} = x_2h_{11}(1-h_{11})\frac{dL}{dh_{11}}$$
$$\frac{dL}{dw_2} = \frac{dh_{12}}{dw_2}\frac{dL}{dh_{12}} = x_1h_{12}(1-h_{12})\frac{dL}{dh_{12}}$$
$$\frac{dL}{dw_3} = \frac{dh_{12}}{dw_3}\frac{dL}{dh_{12}} = x_2h_{12}(1-h_{12})\frac{dL}{dh_{12}}$$
$$\Delta b_0 = -\alpha\frac{dL}{db_0}$$
$$\Delta b_1 = -\alpha\frac{dL}{db_1}$$
$$\Delta b_2 = -\alpha\frac{dL}{db_2}$$
$$\Delta w_0 = -\alpha\frac{dL}{dw_0}$$
$$\Delta w_1 = -\alpha\frac{dL}{dw_1}$$
$$\Delta w_2 = -\alpha\frac{dL}{dw_2}$$
$$\Delta w_3 = -\alpha\frac{dL}{dw_3}$$
$$\Delta w_4 = -\alpha\frac{dL}{dw_4}$$
$$\Delta w_5 = -\alpha\frac{dL}{dw_5}$$

dL/dy_pred	-0.114907491
dL/db2	-0.011686523844
dL/dw4	-0.011682603 88
dL/dw5	-0.011608307 24
dL/dh11	-0.023368154

dL/dh12	-0.0467460788
dL/db1	-0.000310772313 7
dL/db0	-0.000007835649003
dL/dw0	-0.000007835649003
dL/dw1	-0.000007835649003
dL/dw2	-0.000310772313 7
dL/dw3	-0.000310772313 7
db0	0.000007835649003
db1	0.000310772313 7
db2	0.001168652384
dw0	0.000007835649003
dw1	0.000007835649003
dw2	0.000310772313 7
dw3	0.000310772313 7
dw4	0.001168260388
dw5	0.001160830724

Updated parameters

b0	0.9997755478
b1	-5.999974574

b2	-3.92940414
w0	3.000000784
w1	4.000000784
w2	6.000031077
w3	5.000031077
w4	2.000749515
w5	4.001159414

Forward propagation (x1=0, x2=1, y=1)

Current parameters

b0	0.9997755478
b1	-5.999974574
b2	-3.92940414
w0	3.000000784
w1	4.000000784
w2	6.000031077
w3	5.000031077
w4	2.000749515
w5	4.001159414

Note

$$h_{11} = g(w_0 x_1 + w_1 x_2 + b_0)$$
$$h_{12} = g(w_2 x_1 + w_3 x_2 + b_1)$$
$$y_{pred} = g(w_4 h_{11} + w_5 h_{12} + b_2)$$
$$L = \tfrac{1}{2}(y - y_{pred})^2$$
$$g(z) = \tfrac{1}{1+e^{-z}}$$

Since x1=0 and x2=1,

h11	0.9933056619
h12	0.2689525307
y_pred	0.2961026464

Back propagation (x1=0, x2=1, y=1)

Now we want to back propagate the error.

Note

$$g'(z) = g(z)(1 - g(z))$$

$$L = \frac{1}{2}(y - y_{pred})^2$$

$$\frac{dL}{dy_{pred}} = -(y - y_{pred}) = y_{pred} - y$$

$$\frac{dL}{db_2} = \frac{dy_{pred}}{db_2}\frac{dL}{dy_{pred}} = y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{dw_4} = \frac{dy_{pred}}{dw_4}\frac{dL}{dy_{pred}} = h_{11}y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{dw_5} = \frac{dy_{pred}}{dw_5}\frac{dL}{dy_{pred}} = h_{12}y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{dh_{11}} = \frac{dy_{pred}}{dh_{11}}\frac{dL}{dy_{pred}} = w_4 y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{dh_{12}} = \frac{dy_{pred}}{dh_{12}}\frac{dL}{dy_{pred}} = w_5 y_{pred}(1 - y_{pred})\frac{dL}{dy_{pred}}$$

$$\frac{dL}{db_1} = \frac{dh_{12}}{db_1}\frac{dL}{dh_{12}} = h_{12}(1 - h_{12})\frac{dL}{dh_{12}}$$

$$\frac{dL}{db_0} = \frac{dh_{11}}{db_0}\frac{dL}{dh_{11}} = h_{11}(1 - h_{11})\frac{dL}{dh_{11}}$$

$$\frac{dL}{dw_0} = \frac{dh_{11}}{dw_0}\frac{dL}{dh_{11}} = x_1 h_{11}(1 - h_{11})\frac{dL}{dh_{11}}$$

$$\frac{dL}{dw_1} = \frac{dh_{11}}{dw_1}\frac{dL}{dh_{11}} = x_2 h_{11}(1 - h_{11})\frac{dL}{dh_{11}}$$

$$\frac{dL}{dw_2} = \frac{dh_{12}}{dw_2}\frac{dL}{dh_{12}} = x_1 h_{12}(1 - h_{12})\frac{dL}{dh_{12}}$$

$$\frac{dL}{dw_3} = \frac{dh_{12}}{dw_3}\frac{dL}{dh_{12}} = x_2 h_{12}(1 - h_{12})\frac{dL}{dh_{12}}$$

$$\Delta b_0 = -\alpha\frac{dL}{db_0}$$

$$\Delta b_1 = -\alpha\frac{dL}{db_1}$$

$$\Delta b_2 = -\alpha\frac{dL}{db_2}$$

$$\Delta w_0 = -\alpha\frac{dL}{dw_0}$$

$$\Delta w_1 = -\alpha\frac{dL}{dw_1}$$

$$\Delta w_2 = -\alpha \frac{dL}{dw_2}$$
$$\Delta w_3 = -\alpha \frac{dL}{dw_3}$$
$$\Delta w_4 = -\alpha \frac{dL}{dw_4}$$
$$\Delta w_5 = -\alpha \frac{dL}{dw_5}$$

dL/dy_pred	-0.7038973536
dL/db2	-0.1467104178
dL/dw4	-0.1457282886
dL/dw5	-0.03945813814
dL/dh11	-0.2935307972
dL/dh12	-0.5870117692
dL/db1	-0.1154165323
dL/db0	-0.001951840048
dL/dw0	0
dL/dw1	-0.001951840048
dL/dw2	0
dL/dw3	-0.1154165323
db0	0.001951840048
db1	0.01154165323
db2	0.01467104178
dw0	0
dw1	0.001951840048
dw2	0
dw3	0.01154165323
dw4	0.01457282886
dw5	0.003945813814

Update parameters

b0	0.9999707318
b1	-5.988432921
b2	-3.914733099
w0	3.000000784
w1	4.000195968
w2	6.000031077
w3	5.01157273
w4	2.015322344
w5	4.005105228

(b) accuracy

Current parameters

b0	0.9999707318
b1	-5.988432921
b2	-3.914733099
w0	3.000000784
w1	4.000195968
w2	6.000031077
w3	5.01157273
w4	2.015322344
w5	4.005105228

Our data

x1	x2	y
0	0	0
1	1	1

0	1	1

So the predictions are the follows:

Note

$$h_{11} = g(w_0 x_1 + w_1 x_2 + b_0)$$
$$h_{12} = g(w_2 x_1 + w_3 x_2 + b_1)$$
$$y_{pred} = g(w_4 h_{11} + w_5 h_{12} + b_2)$$
$$L = \frac{1}{2}(y - y_{pred})^2$$
$$g(z) = \frac{1}{1+e^{-z}}$$

[x1=0, x2=0]

h11	0.7310528241
h12	0.002501318284
y_pred	0.08080882363

[x1=1, x2=1]

h11	0.999664706
h12	0.9934594429
y_pred	0.8888277964

[x1=0, x2=1]

h11	0.9933082572
h12	0.2735152346
y_pred	0.306307458

[prediction summary]

x1	x2	y	y_pred	classified as
0	0	0	0.08080882363	FALSE
1	1	1	0.8888277964	TRUE
0	1	1	0.306307458	FALSE

Let’s build a confusion matrix first

TP	1
TN	1
FP	0
FN	1

Confusion matrix		Predicted	
		N	P
Actual	N	1	0
	P	1	1

Since accuracy is (TP + TN) / (TP + TN + FP + FN)

accuracy	0.6666666667

(c) precision

Since precision is TP / (TP + FP)

precision	1

(d) recall

Since recall is TP / (TP + FN)

recall	0.5

(e) f1 score

Since f1 score is 2 * (precision * recall) / (precision + recall)

f1	0.6666666667

Parameter	Initial value
b_0	1
b_1	-6
b_2	-3.93
w_0	3
w_1	4
w_2	6
w_3	5
w_4	2
w_5	4

Table 3: ANN initial state

x_1	x_2	y
0	0	0
1	1	1
0	1	1

Table 4: ANN training data

(f) Plot the ROC curve.

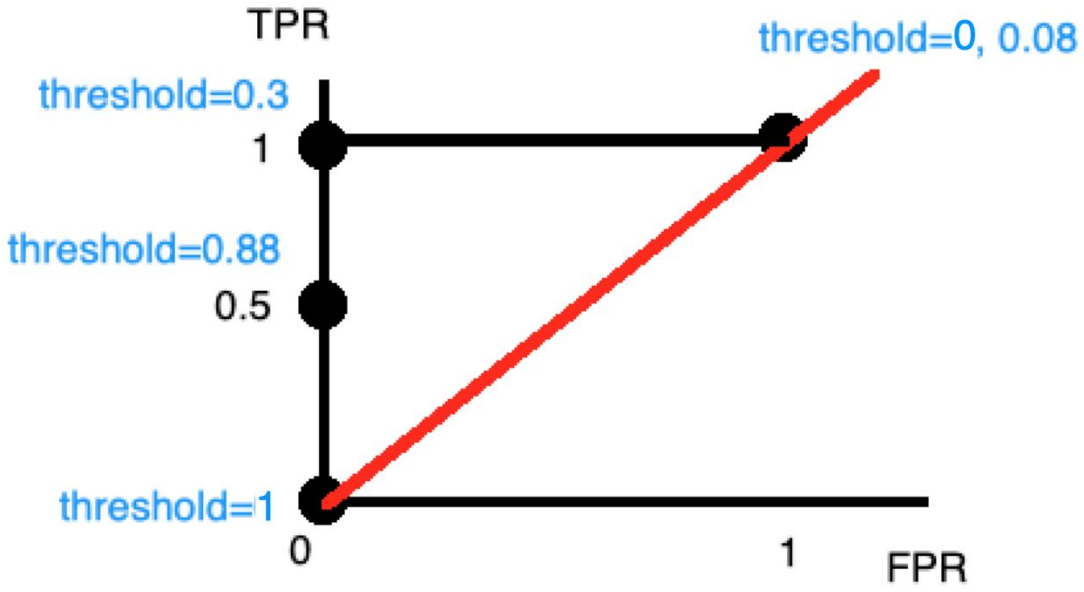
(g) What is your model's AUC-ROC?

(f) Note ROC curve is TPR / FPR where TPR = TP / (TP + FN) and FPR = FP / (TN + FP)

threshold	TP	TN	FP	FN	TPR	FPR
0	2	0	1	0	1	1
0.08080882363	2	0	1	0	1	1
0.306307458	2	1	0	0	1	0
0.8888277964	1	1	0	1	0.5	0

1	0	1	0	2	0	0

Thus the ROC curve is drawn as follows:



(g) 1

7 Reinforcement Learning

Consider the following instance of reinforcement learning

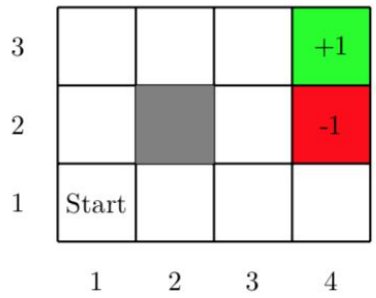


Figure 2: Reinforcement learning grid world

Assume the following:

- The agent lives in a grid
- The agent starts in the bottom left
- Walls (shaded grids) block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West
  - 10% of the time, North takes the agent East
  - If there is a wall in the direction the agent would have taken, the agent stays put
- $\gamma = .8$
- Big rewards come at the end
- Goal: maximize sum of rewards

Use Q-learning to find an optimal path in the grid. Show each step. State all your assumptions.

This question is removed

8 (Money)ball So Hard

Watch the 2011 movie *Moneyball* (directed by Bennett Miller and written by Steven Zaillian and Aaron Sorkin).

(a) Write a detailed formulation of the problem discussed in the movie and outline, step-by-step, how you would go about solving the problem (data collection, data cleaning, ML, and all other steps).

(b) Discuss what can go wrong when using this method in practice and how it can be improved.

(a)

1. The problem is this: we want to hire very talented baseball players with low budgets that can win the championship.

2. Side Note: Traditional analysis of baseball players is biased. They are affected by age, personality, and appearance for instance.

3. Data collection is the key here. What are the features that relate to winning the game? We will collect data of number of pitches, number of balls, number of strikes, age, and average run. It's okay to include features that might not be related because we can either remove them during data cleaning stage or ML algorithm simply won't use these features when predicting values.

4. For data cleaning, we can use things we learned from projects such as standard scaling and one hot encoding.

5. Now we could use something like neural network to see which player is likely to win a game

(b)

In practice, it is hard to tell if winning means a good player. For example, a bad player could still win in a good team, and a good player could lose in a bad team. To fix this problem, we can adjust the scores to be relative rather than absolute. For example, if the team won with 10 points and a player got 5 points in the game, then this player is good. If the team won with 100 points and a player got 5 points in the game, then this player is not as good.