CSCE 822 – Homework 3

Name: Sadman Sadeed Omee
ID: W45085628
Email: somee@email.sc.edu

Problem-1:

Hold-out is when we split the dataset into a train and test set. We only use one run in hold out and train the model with the train set and test it with the test set.

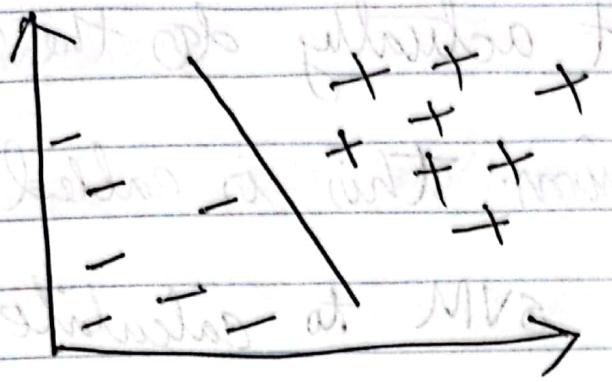
Cross validation is when we split the dataset into k different groups and at each run, we only use one partition as test set and the rest as train set. We do this until each partition is a test set at least once.

Holdout is used when the size of the dataset is really huge and we have low computational

resources and get results quickly.
CV has the advantage of being the better prediction method as each run we are using different test sets and so it learns to work better with unseen data. Also, as we are averaging the CV result, we are getting better estimate than holdout which only depend on one run.

Problem-2:

(i) Maximum-margin classifier: Here we find a decision boundary to separate objects of two different class such that the closest objects ~~of~~ from the decision boundary of both classes remain as far as possible.



Here we can see that we have placed the decision boundary to separate both classes objects as much as possible.

(ii) Map to high dimensions: ~~Some none non linearly class Some datasets can~~ Some cases, we can not separate classes based on linear classifier, in that cases we need to transform the points into higher dimensions. Doing so, we can separate the classes easily by putting a decision boundary. (costly).

(iii) Kernel trick: Kernel functions only calculate the relationships between every pair of points as if they are in the higher dimensions.

But they don't actually do the mapping to higher dimension. this is called kernel trick. this makes SVM to calculate decision boundary for even non-linearly separable classes even though it is a linear classifier.

DNA sequence data can be multidimensional. But SVM can put a decision boundary to classify that using kernel trick. Also tree/graph data can be used as input in SVM. that makes SVM to be useful in graph/network classification.

Problem-3.1:

1. AdaBoost.
2. XGBoost.
3. RandomForest.

Problem 3.2:

Ensemble models reduces the variance of prediction. Also when prediction of many models are averaged, it gives much better prediction results because it is tested on many models and standalone algorithms depend on just one models output.

Problem 3.3:

~~$^{7}C_{7}(0.7)^{7}_{0.} + {}^{7}C_{6}$~~

$${}^{7}C_{7}(0.7)^{7}(0.3)^{0} + {}^{7}C_{6}(0.7)^{6}0.3 + {}^{7}C_{5}(0.7)^{5}(0.3)^{2}$$

$$+ {}^{7}C_{4}(0.7)^{4}(0.3)^{3}$$

$$= 0.874$$

$$= 87.4\%$$

Problem -3.4:

Boosting can learn weak rule and assign more weights to ~~those~~ those points that were

misclassified in the previous models output so the new model can focus more on those misclassified samples. But bagging can not learn from past misclassification errors like boosting.

Problem-4:

1) Unbalanced data means that data of one class is much larger than the other class. For example, if in a dataset, data of one class is a total of ~~100~~ 998 and only 2 data ~~are~~ are of another class. The problem is, the classifier always classifies any new samples to the majority class. For example, ~~to~~ lets say we have 100 new samples where 97 data are in the first class and 3 data are

in the second class. The classifier classes all the samples ~~are~~ as of the first class. And ~~so, the~~ yet, the accuracy is still 97% even though it didn't classify any samples correctly of class B.

2) Using different evaluation matrices: We can use evaluation matrix like Precision, Recall, F_1 Score, Mcc Score, AUC for this. These actually take account of the False positive and False negative values of each class and so ~~it~~ these matrix will penalize any misclassification of the minority class more than othe matrix like MSE, MAE etc.

Resampling the training set: For training we can either over sample minority class or undersample majority class. This will help to reduce the problem of unbalanced data and this approach actually achieves better AUC value.

Changing classification algorithm: This approach work well when one classification algorithm does not fit in the scenario. For example, Random forest doesn't work well when there are too many irrelavant features. So, we can shift to SVM to get better results.

Problem – 5.1

The problem is vanishing and exploding gradient problem. Vanishing gradient can be solved using skip connections and / ReLU activation function. Skip connection adds previous ~~weights~~ layer's weight to future layers which stops gradients to vanish. Also, ReLU does not cause a small derivative, i.e, gradient does not vanish so soon. Gradient Clipping can take care of the exploding gradient problem which cuts gradients after a certain threshold.

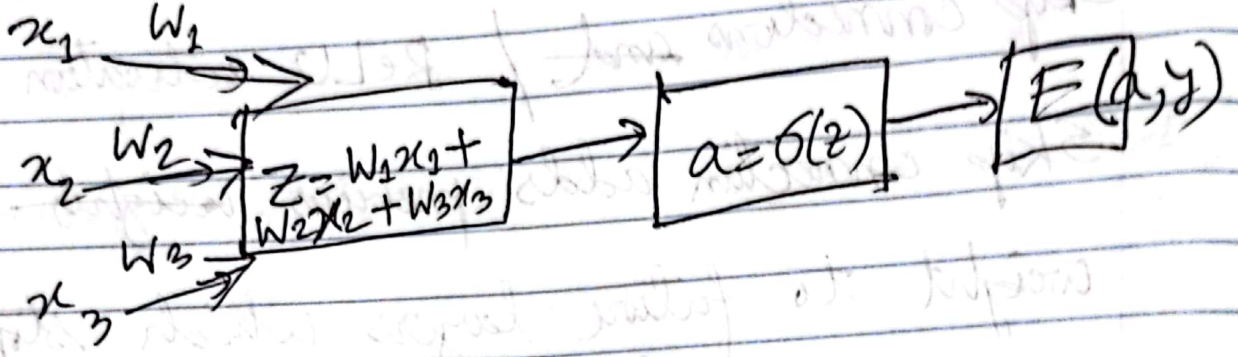
Problem – 5.2:

$$W_1 = W_1 - \frac{dE}{dW_1}$$

$$W_2 = W_2 - \frac{dE}{dW_2}$$

$$W_3 = W_3 - \frac{dE}{dW_3}$$

$\frac{dE}{dW_1} = \frac{dE}{d}$ Redrawing the neural network computation graph,



$$\frac{dE}{dW_1} = \frac{dE}{da} \frac{da}{dz} \frac{dz}{dW_1}$$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)\left(a(1-a)\right)x_1$$

$$= (a-y)\,x_1.$$

Problem 5.3:

Dimension$(C_1) = (60-5+1) \times (60-5+1) \times 10$

$= 56 \times 56 \times 10$

Dimension$(S_2) = (56-2+1) \times (56-2+1) \times 10$

$= 55 \times 55 \times 10$

Dimension $(C_3) = \left(\frac{55-3}{+1}\right) \times (55-3+1) \times 5$

$= 53 \times 53 \times 5$

Dimension $(S_4) = \left(\frac{53}{2}-2+1\right) \times (53-2+1) \times 5$

$= 52 \times 52 \times 5$.

Problem-6:

The main difference of RNN and fully connected FC Network is that RNN is specially designed to handle sequential data, ~~text~~, where each output token $\langle y_i \rangle$ depends on current input and previous output aswell. In, fed forward neural network, as such utilization is done and all input are passed to the network at once.

The main idea behind autoencoder is to efficiently compress and encode data and then learn how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.

Autoencoders are used for tasks like anomaly detection, image denoising, dimensionality reduction, image compression, feature extraction, etc.