Final Project for DSCI320, Fall 2024

In this project we will

(1) Implement the very basic version of support vector machine.

(2) Apply support vector machine to do a classification problem.

(3) Apply neural network to do the same classification problem.

We have three classification problems for you to choose. The first is the identification of fake news, with dataset available at

`https://www.kaggle.com/competitions/fake-news/data`,

and the second is the classification of fetal cardiotocography, with dataset available at

`https://www.kaggle.com/datasets/akshat0007/fetalhr`,

and the third is the brain tumor classification, with dataset available at

`https://github.com/sartajbhuvaji/brain-tumor-classification-dataset`.

The last problem is computationally demanding and will take long time to train, but it will be very rewarding if you are able to complete the implementation.

For any chosen problem, data normalization will always be helpful. Those who understand the singular value decomposition (SVD) well could try to use SVD first to reduce the dimensionality before moving to the classification using SVM or neural network.

1 Part 1: Naïve implementation of support vector machine

Here we will create the a small 2-D dataset and implement a naïve algorithm to find the hyperplane that separate the two clusters in the dataset. Let the hyperplane be $y = \mathbf{w}^T\mathbf{x}$ where x is the extended variable padded with 1s, the algorithm goes as

A simple way to generate small datasets is using `make_blobs` in `sklearn`. This way one can generate points of a given number in 2-D plane clustered around given centers, for example:

```
from sklearn.datasets import make_blobs
centers = [(1,1),(-1,-1])] # centers of the two clusters
cluster_std = [0.2,0.2] # variance of the two clusters
X, Y=make_blobs(n_samples=10, cluster_std=cluster_std, centers=centers,
                n_features=2, random_state=1) # two clusters, 10 points in total.
```

Please note that the labels of the two clusters will be 0 and 1 in this case, and you will have to change them to be 1 and -1, so they are consistent with the algorithm. For this part, please make a few figures showing the improved separation of clusters as the iterations continue and the final state of the separation. Attach these figures to your project report.

```
1: Define learning rate $l_r = 0.1$
2: Define expand factor $f_e = 0.9$
3: Define reduce factor $f_r = 1.1$
4: Pick an arbitrary data point $(\mathbf{x}, y)$ and determine whether it is misclassified
5: if Classified correctly then
6:     if Margin too small then
7:         $\mathbf{w} \leftarrow \mathbf{w} + l_r \cdot f_r \cdot y\mathbf{x}$
8:     else if Margin too wide then
9:         $\mathbf{w} \leftarrow f_e \cdot \mathbf{w}$
10:    end if
11: else
12:     $\mathbf{w} \leftarrow \mathbf{w} + l_r \cdot y\mathbf{x}$
13: end if
14: Goto 4 and continue the process until convergence, or a preset number of iterations is reached.
```

2 Part 2: Classification using support vector machine

In this step we will implement the support vector machine modules in `sklearn` (`https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`) which also allows you to use various type of kernels. You can then compare results with and without kernels. You are expected to

(a) Load and understand the dataset.

(b) Implement SVM in sklearn for the multi-class classification.

(c) Understand the functionality of various kernels for SVM and compare their performance for the problem. Quantify the accuracy of your classifications.

3 Part 3: Classification using neural network

In this step you will implement neural network for classification. You are not required to build your own neural network and its training. Instead, you may use the Multilayer Perception Classifier in sklearn. More details information about the classification function can be found at `https://scikit-learn.org/stable/modules/neural_networks_supervised.html` and `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`. Alternatively, you could work with PyTorch (`https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`). In either case, we might want to walk through the sample given in these introductory pages and make sure they are pruning as expected. Then read the sample code very carefully, understand each line and every single feature or parameter of the method, before moving to build our own implementation and applications. These features or parameter include

(a) Depth and width of the network.

(b) Solver of the minimization problem.

(c) Target function and strength of regularization

(d) Batch size.

(e) Initial and adaptive earning rate.

(f) Initialization of the network.

(g) Momentum method.

(h) Termination of the training.

(i) Use of well-trained network for prediction on test dataset.

Your datasets contain both train and test sets, so you will be able to quantify your classification.

At the end of these three parts, each group is expected to

(a) Describe the problem and your algorithm.

(b) Describe your implementation of the algorithm, major steps, and key parameters.

(c) Describe the training process. Quantify the accuracy of your classifications.

(d) Wrap up the results and finish a project report with eight or more pages (excluding your code), including diagrams, tables, or figures.

(e) Python code will be submitted separately.