**ENPM703- Assignment-1**

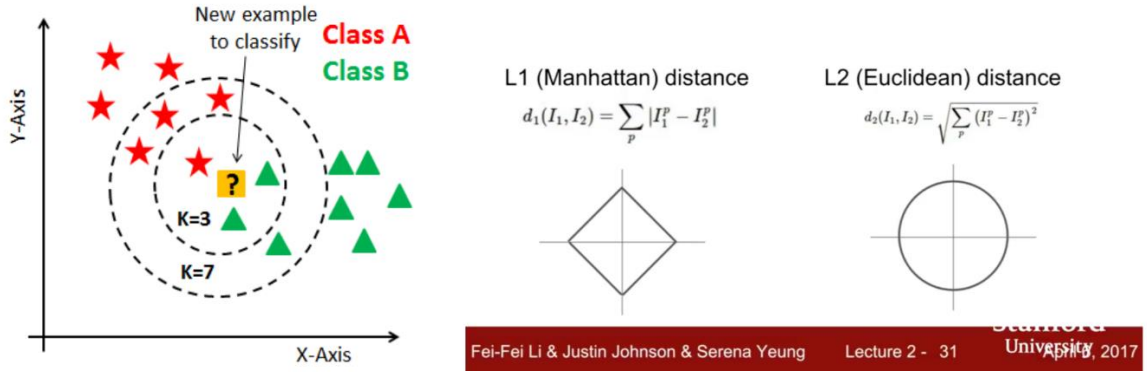
**Part1: KNN Classifier**

## CIFAR-10 Dataset

Contains collection of 60000 RGB images with dimensions 32x32. Thus, loading the training dataset forms an array of size (50000x32x32x3). This data is pre-processed to a row matrix of dimensions (50000x3072) by straightening up the pixel values. Only a sample of 5000 images for training and 500 images for testing are used.

## K-NN Classifier

K nearest neighbour is a classification algorithm which classifies dataset according to the similarity with other data points. This similarity can be found by l1-Manhattan or l2-Eucladian distance. The class of the test data is predicted based on the closest distance between the test and k nearest points.




## L1 and L2 Distance

L1 distance gives the absolute difference between the pixel values. This type of distance measurement is suited if the dimensional features have particular meaning such as make, number of seats, year of manufacturing, etc. of a car.

L2 distance gives the straight-line distance between two points. This can be used if the features do not have any specific meaning, like in this case where they are the pixel values of the images.

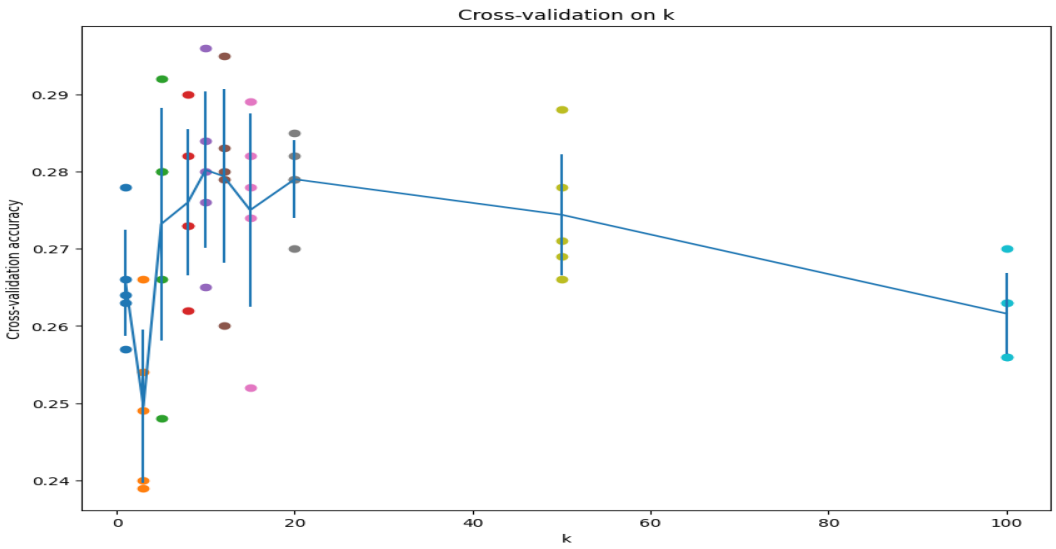
In this algorithm, we have computed l2 distance with two, one and zero loop. Since iterating once is an efficient vectorized code, by using basic matrix algebra, the computation of l2 distance becomes much quicker.

## Cross Validation

Cross Validation is used to determine the best k value such that the algorithm has maximum accuracy. In cross validation, we split the training data into five folds and out of five one of the folds is kept as validating set. The algorithm is trained using the rest of the four folds and the k value is validated using the validating set.

Also, the validating set get changed in each iteration, so that there is no bias in the training. Thus, the training and prediction is done for all five iterations and the accuracy values are recorded in a dictionary.

Finaly, the accuracy values for each k are plotted and the k with maximum accuracy is chosen.




Here both k=10 and k=12 has similar accuracy values, but k=10 has the maximum accuracy and is the suitable value of k for this data set.

**Is K-NN best?**

The maximum accuracy in this classification algorithm is 28.7%, which is slightly higher than the baseline accuracy of 10%. One of the reasons for the minimal performance is that knn is sensitive to redundant features such as background of an image. For example,

in these images the background is mostly similar and the distance value will be minimal, since there is no much variation un the pixel intensities.

Moreover, knn does not perform well if the data is not well sampled. Since knn predicts the class based on the neighbours, if the data is not sampled properly the algorithm might predict the class whichever is closer to the test data. Thus, might not be suitable for minority class samples.

**ENPM703- Assignment-1**

**Part1: SVM**

## CIFAR-10 Dataset

Contains collection of 60000 RGB images with dimensions 32x32. Thus, loading the training dataset forms an array of size (50000x32x32x3). This data is pre-processed to a row matrix of dimensions (50000x3072) by straightening up the pixel values. We use a sample of 49000 images for training, 1000 images for validation and 1000 images for testing are used.

## SVM

SVM is a classification technique which transforms the input data (not linearly separable) to a high dimensional space to make them a linearly separable data. Then an optimal separating plane is found by optimizing the weights of different parameters.

## SVM Loss

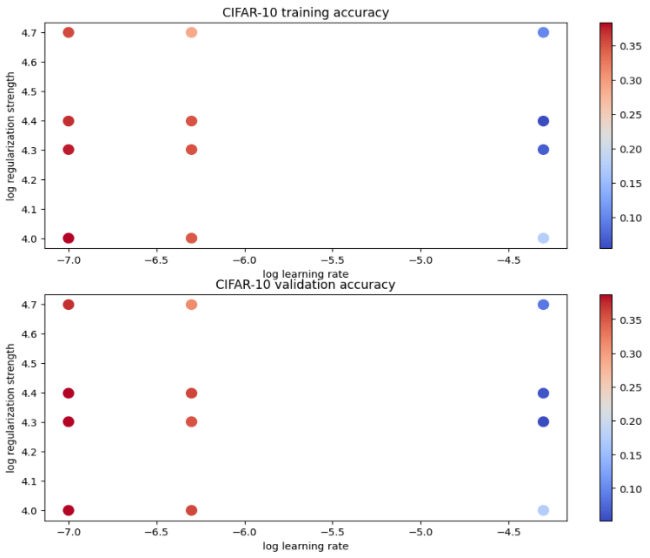
The optimal separable plane is found by a parametric approach, where in we project the flattened to the weights of the plane (ie. taking dot product) and get the scores for all individual class. Then the scores are compared with the ground truth through a loss function and this loss is minimized using gradient descent algorithm.

In this assignment we have computed SVM loss by iterating through each training data and each class, but this approach is time complex. Thus, we use a vectorized approach for finding the loss and its corresponding gradient with respect to W.

## Stochastic Gradient Descent

This gradient which we found is used to find the global minima of the loss function, thus minimizing the loss. Here we sample mini batches from the training set and for each iterations unique batches are used to find the loss and gradient. At the end of each iterations the weights are adjusted based on the leaning rate, which is one of the hyperparameter.

## Cross Validation



The hyperparameter that needs to be optimized are learning rate and regularization strength. Here we initialize few values for the hyperparameters and we iterate through different combinations to find the optimal value. We also store the accuracy values of each combination in a dictionary and visualize to get a clear view of the combinations.

The combination lr = 1.000000e-07 reg = 2.000000e+04 has the maximum accuracy.

**Visualizing Hyperplanes**

When the weights are reshaped and transformed in RGB images, it resembles the visual characters of each class. So, when a test image is projected on these hyperplanes, the plane with maximum resemblance is chosen as the predicted class (ie. image having the maximum dot product with the weights)

**ENPM703- Assignment-1**

**Part3: Softmax**

**Softmax Classifier**

Softmax is used as a multiclass classifier where the scores of each class are transformed into probabilities which are used to predict the class.

**Softmax Loss**

The Softmax loss function takes the output scores of individual classes and transform them to probability values. The scores are the dot product of the image features and the weight that we use for each class. Here the scores are logits and the loss function takes the log of the probabilities of the true class. Since we are dealing with the log of the probabilities, the minimum and maximum value of the output is zero to infinity.

$$\sigma = -\log(e^{sj}/ \Sigma e^{si})$$

Since the model is predicting the probability of Y given X, at initialization the model is not given with the input features the probability becomes the baseline probability. In this case, that is 0.1 thus, the loss at initialization is -log (0.1).

**Gradient Calculation**

Rewriting the loss function as,

$$\sigma = - S_{correct} + \log(\Sigma e^{si})$$

Differentiating individually,

$$D(- S_{correct}) = -x_i$$

$$D(\log(\Sigma e^{si})) = \Sigma e^{si}. x_i$$

Finally, using cross validation the optimal values of learning rate and regularization strength are found.

**SVM vs Softmax**

SVM:


Softmax:


- The visualization of weights is similar in both cases, since both weights resembles the base features of the class images. But, Softmax weights are less noisy because of its probabilistic nature and leading to smooth transition between classes.

- Since SVM uses Hinge loss to make prediction, the output is binary when compared with the probabilistic output of Softmax. Thus, Softmax can be used for multiclass classification which is often the case in real world problems.
- In SVM, the loss can approach to zero and it will not affect the solution after that point but, Softmax on the other hand will make minimal updates and will try to make the probability close to one. Thus, Softmax is more sensitive to changes.

**ENPM703- Assignment-1**

**Part4: Two Layer Neural Network**

## Neural Network

Neural Network consists of layers of interconnected nodes where each node performs a mathematical operation. In a 2-layer neural network, there are two layers of weights: one from the input layer to a hidden layer, and one from the hidden layer to the output layer. The network learns by adjusting these weights to minimize the error between predicted and the ground truth data. Neural networks are powerful for tasks like image classification, speech recognition, since they are able to perform non-linear mappings from input to output.

## Forward Pass

In forward pass input data passes through the network layers to produce the output prediction. In first layer of NN, input data is transformed by multiplying it with the weights and adding biases at each layer, followed by applying activation functions like ReLU. In the second layer the output is computed layer by layer, moving from the input to the output layer, and is used to calculate the loss through loss functions like softmax loss.

## Back Propagation

Backpropagation is the process of computing gradients for each weight in the network and used to minimize the loss function during training. Here we compute the gradient of the loss with respect to the weights using the chain rule.

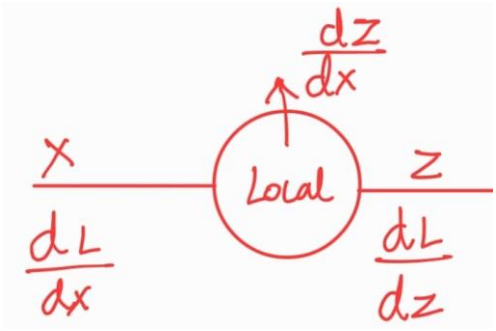
The backward pass starts from the output layer and propagates back through each layer, updating the weights proportional to the negative gradient. Thus, the weights are updates at each iteration to mimic the features of each class.

By using the chain rule, we get,

**Downstream gradient(dL/dx) =**

Local gradient(dz/dx) *

Upstream gradient(dL/dz)



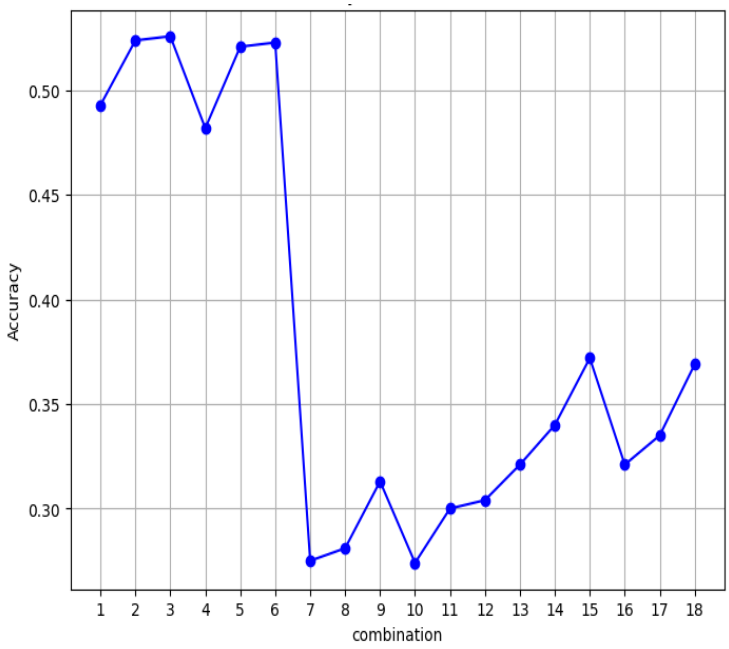

## Backprop with Matrices

Since all the weights and the gradient of loss are all matrices, we are dealing with both upstream and downstream gradients as matrices. Since matrix multiplication is a computationally expensive task, we find patterns in these matrix multiplications and implement those patterns.

Thus, we boil down to,

1. **Downstream gradient(dL/dx) = Upstream gradient(dL/dz) * Transpose of weight($W^T$)**
2. **Downstream gradient(dL/dw) = Transpose of weight($X^T$) * Upstream gradient(dL/dz)**

**Hyperparameters Tuning**




1: [‘lr:0.001', ‘reg:1e-05', ‘hid:32'],
2: [‘lr:0.001', ‘reg:1e-05', ‘hid:64'],
3: [‘lr:0.001', ‘reg:1e-05', ‘hid:128'],
4: [‘lr:0.001', ‘reg:1e-06', ‘hid:32'],
5: [‘lr:0.001', ‘reg:1e-06', ‘hid:64'],
6: [‘lr:0.001', ‘reg:1e-06', ‘hid:128'],
7: [‘lr:1e-05', ‘reg:1e-05', ‘hid:32'],
8: [‘lr:1e-05', ‘reg:1e-05', ‘hid:64'],
9: [‘lr:1e-05', ‘reg:1e-05', ‘hid:128'],
10: [‘lr:1e-05', ‘reg:1e-06', ‘hid:32'],
11: [‘lr:1e-05', ‘reg:1e-06', ‘hid:64'],
12: [‘lr:1e-05', ‘reg:1e-06', ‘hid:128'],
13: [‘lr:2e-05', ‘reg:1e-05', ‘hid:32'],
14: [‘lr:2e-05', ‘reg:1e-05', ‘hid:64'],
15: [‘lr:2e-05', ‘reg:1e-05', ‘hid:128'],
16: [‘lr:2e-05', ‘reg:1e-06', ‘hid:32'],
17: [‘lr:2e-05', ‘reg:1e-06', ‘hid:64'],
18: [‘lr:2e-05', ‘reg:1e-06', ‘hid:128']

**Learning Rate:** This controls how fast the model updates its weights during training. If learning rate is too small, the model will be very slow and if its is too high it may result in overshooting.

**Regularization Strength:** This controls how the weights should be with respect to the image features. If this is too small, the model might overfit and if it is too high, the model might be generalized and result in underfitting

**No. Hidden Layer:** This is used in neural network to learn complex patterns from the data. If it is too high, the model might overfit and if it is too low the model might be inefficient,

**Why NN is Better?**

Neural networks outperform traditional machine learning models because they can automatically learn complex features from raw data. They are better at handling high dimensional and non-linear data, and they can generalize better when trained with large datasets.

Neural networks also enable deep learning, where multiple hidden layers allow for learning more abstract and hierarchical patterns, leading to state-of-the-art results in fields like computer vision and natural language processing.

# ENPM703- Assignment-1

## Part5: Neural Network on Image Features

### Feature Extraction

Feature extraction is the process of converting raw image data into a format that can be easily interpreted by machine learning models. In context of image classification, features represent distinctive characteristics like edges, textures, shapes, or colors that help training process to be more efficient.

It makes model performance by reducing noise and irrelevant data, and allows algorithms to focus on the most relevant parts of the image.

### Methods

**Histogram of Oriented Gradients (HOG)**: Captures the structure and shape of objects by calculating the gradient orientation in localized portions of the image. Since, HOG captures texture of the image, it is robust to change in pose and color of the image contents.

**Color Histogram (HSV)**: Represents the distribution of colors in an image using the Hue, Saturation, and Value (brightness) components, which helps in distinguishing objects based on their colors.

### Overview of SVM and Neural Networks

SVM is a classification method that takes input data, which may not be linearly separable, and projects it into a higher-dimensional space to make it linearly separable. An optimal separating hyperplane is then determined by adjusting the weights of various parameters.

In a 2-layer NN, there are two layers of weights: one connecting the input layer to a hidden layer, and another linking the hidden layer with the output layer. The network learns by modifying these weights to reduce the error between the predicted values and the actual ground truth data.

### SVM vs NN

- SVM performs well on smaller datasets with clear class boundaries, while neural networks are better suited for larger datasets with complex patterns and SVM is faster for training on smaller datasets.
- SVM is less prone to overfitting while there are many features, whereas neural networks need careful tuning to avoid overfitting by adjusting different hyperparameters.
- Neural networks scale effectively with large datasets and benefit from GPU acceleration, whereas SVM may struggle with larger datasets due to higher time complexity.

### Raw Image Data vs Feature Extracted Data

Raw image data consists of pixel values that represent the image in its original form but it is too large and complex for efficient model training. Feature extracted data simplifies the raw image by focusing on important characteristics like edges, textures, or colors, reducing its complexity. Using feature extracted data helps models like SVM or neural networks to learn faster and perform better by removing irrelevant and redundant information. While raw data

might contain more detailed information, feature extraction makes it easier for ML models to focus on patterns relevant to the task.

Both validation and test accuracies are higher with feature data when compared with the raw image data. Thus, preprocessing data by extracting features is a reliable method to increase the efficiency of the model.