Homework 3: Multi-Layer Perceptron (MLP)

In this exercise, you will implement a **two-layer neural network** to perform classification and test it on the CIFAR-10 dataset. [**20 points**]

1. Load the CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

We load the CIFAR-10 dataset from disk and perform preprocessing to prepare it for the two-layer neural network classifier.

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from load_data import get_CIFAR10_data

X_train, y_train, X_val, y_val, X_test, y_test = get_CIFAR10_data()
print('Training data shape:\t', X_train.shape)
print('Training labels shape:\t', y_train.shape)
print('Validation data shape:\t', X_val.shape)
print('Validation labels shape:', y_val.shape)
print('Test data shape:\t', X_test.shape)
print('Test labels shape:\t', y_test.shape)
```

2. Implement a two-layer neural network

The two-layer neural network has the following architecture:

```
input - fully connected layer - ReLU - fully connected layer - softmax
```

The outputs of the second fully-connected layer are the scores for each class.

We use the class `TwoLayerNet` in the file `two_layer_net.py` to represent instances of our network.

You need to fill the functions `loss`, `train`, and `predict` in the class `TwoLayerNet`. [**15 points**]

In [ ]:
```
from two_layer_net import TwoLayerNet

input_size = 3072
hidden_size = 80
num_classes = 10
net = TwoLayerNet(input_size, hidden_size, num_classes)
```

3. Train a network

We use SGD to train our network.

NOTE：Do not change preset parameters.

In [ ]:
```
# Train the network
stats = net.train(X_train, y_train, X_val, y_val,
                  learning_rate=1e-3, learning_rate_decay=0.95,
                  reg=1e-1, num_iters=1500, batch_size=490, verbose=True)

# Predict on the validation set
val_acc = (net.predict(X_val) == y_val).mean()
print('Validation accuracy: ', val_acc)
```

In [ ]:
```
# Plot the loss function
plt.plot(stats['loss_history'])
plt.title('Loss history')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.show()
```

In [ ]:
```
# Plot train / validation accuracies
plt.plot(stats['train_acc_history'], label='train')
plt.plot(stats['val_acc_history'], label='val')
plt.title('Classification accuracy history')
plt.xlabel('Epoch')
plt.ylabel('Classification accuracy')
plt.legend()
plt.show()
```

Question:

What is the relationship between epoch, batch size, sample size and iteration? [**5 points**]

Answer:

[fill this]

4. Run on the test set

In [ ]:
```
test_acc = (net.predict(X_test) == y_test).mean()
print('Test accuracy: ', test_acc)
correct_test_acc = 0.502
print('difference: ', correct_test_acc-test_acc)
```

5. Export as PDF

Click `File` -> `Print`

NOTE: Please make sure that the submitted notebook has been run and the cell outputs are visible.