Course: Data Mining

Name: Ahmed Khaled Saad Seif

ID: 06

Name: Hisham Osama Abd Al-Salam

ID: 82

Assignment 4 - Image Classification

Introduction:

In this assignment you will practice putting together a simple image classification pipeline, based on the k-Nearest Neighbor or the SVM/Softmax classifier. The goals of this assignment are as follows:

- understand the basic **Image Classification pipeline** and the data-driven approach (train/predict stages)
- understand the train/val/test **splits** and the use of validation data for **hyperparameter tuning**.
- develop proficiency in writing efficient **vectorized** code with numpy
- implement and apply a k-Nearest Neighbor (**kNN**) classifier
- implement and apply a Multiclass Support Vector Machine (**SVM**) classifier
- implement and apply a **Softmax** classifier
- implement and apply a **Two layer neural network** classifier
- understand the differences and tradeoffs between these classifiers
- get a basic understanding of performance improvements from using **higher-level representations** than raw pixels (e.g. color histograms, Histogram of Gradient (HOG) features)

Data Set :

The **CIFAR-10 dataset** (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research.[1][2] The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes.[3] The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.[4]
Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works. Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10.
CIFAR-10 is a labeled subset of the 80 million tiny images dataset. When the dataset was created, students were paid to label all of the images.[5]

Classes	10
Samples total	50,000
Dimensionality	32x32x3
Flattened Dimensionality	3072

K-NN :

Formulas:

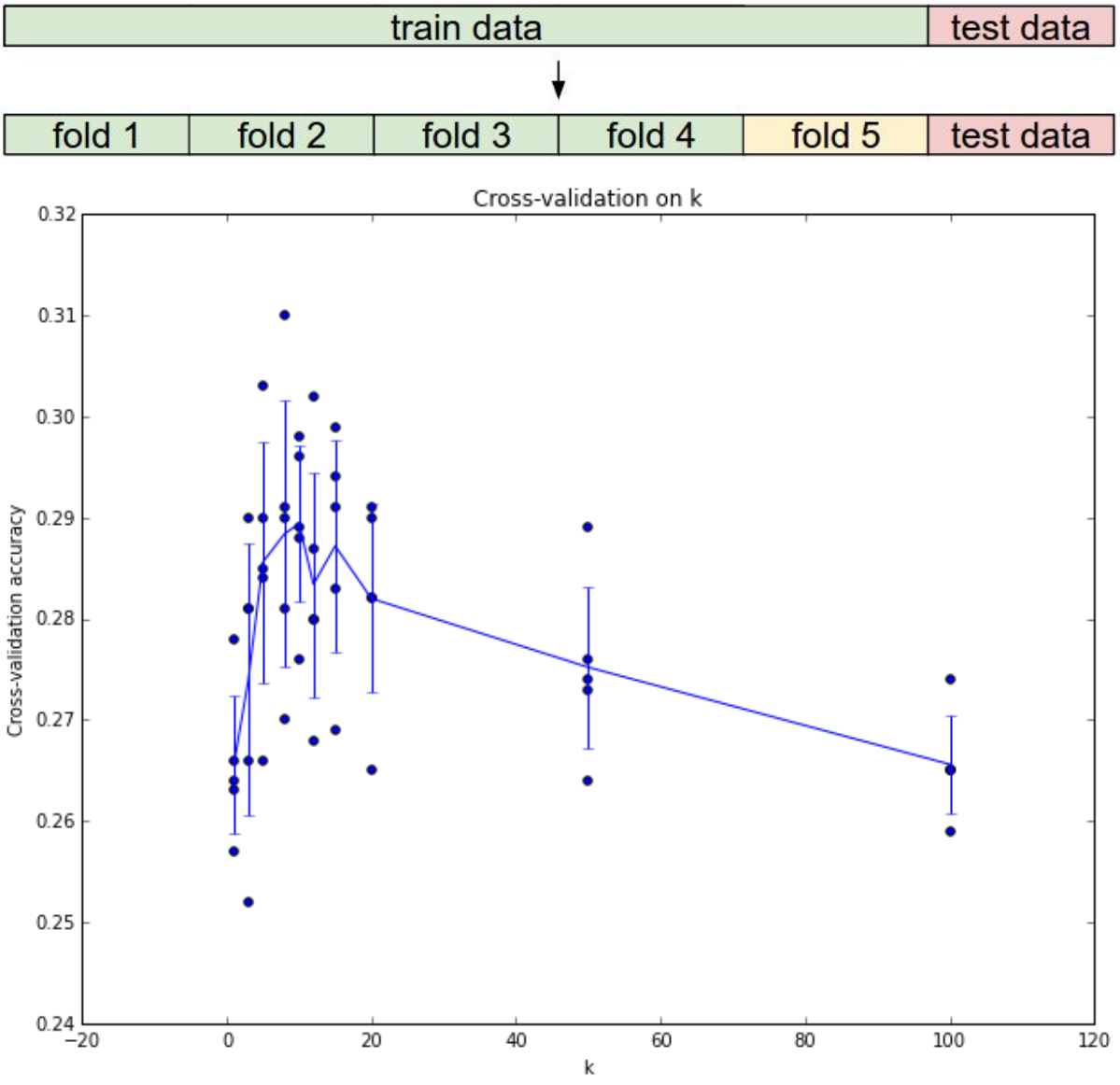
L1 distance (not used in the assignment) $d1(I1, I2) = \sum p|Ip1 - Ip2|d1(I1, I2) = \sum p|I1p - I2p|$

L2 distance: $d2(I1, I2) = \sum p\sqrt{(Ip1 - Ip2)2}$

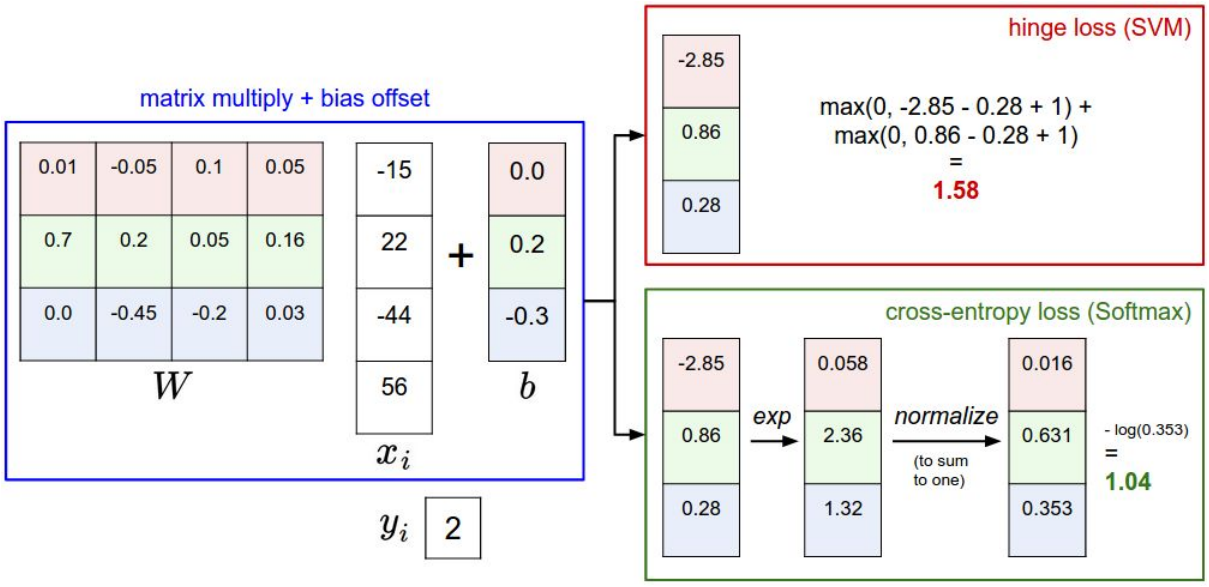
Sample Code:

```
distances = np.sqrt(np.sum(np.square(self.Xtr - X[i,:]), axis = 1))
```

Cross Validation:



SVM :



Loss function

Quality of weights is often expressed by a loss function, our unhappiness with classification result, and we want its value to be as small as possible. To minimize the loss, we have to define a loss function and find their partial derivatives with respect to the weights to update them iteratively.

SVM loss (a.k.a. hinge loss) function can be defined as:

$$Li=\sum j\neq yi[\max(0,xiwj-xiwyi+\Delta)](1)$$

where

- i iterates over all N examples,
- j iterates over all C classes,
- L_i is loss for classifying a single example x_i (row vector),
- w_j is the weights (column vector) for computing the score of class j ,
- y_i is the index of the correct class of x_i , and
- Δ is a margin parameter

Intuitively, SVM wants score, $x_i w_{y_i}$, of the correct class, y_i , to be greater than any other classes, $x_i w_j$, by at least Δ such that the loss becomes zero (clamped with the max operation).

Gredient Descent

$$Li=\sum j\neq yi[\max(0,wTjxi-wTyixi+\Delta)]\text{Li}=\sum\text{j}\neq\text{yi}[\max(0,\text{wjTxi}-\text{wyiTxi}+\Delta)]$$

We can differentiate the function with respect to the weights. For example, taking the gradient with respect to w_{yi}wyi we obtain:

$$\nabla wyiLi=-(\sum j\neq yi 1(wTjxi-wTyixi+\Delta>0))xi\nabla\text{wyiLi}=-(\sum\text{j}\neq\text{yi}1(\text{wjTxi}-\text{wyiTxi}+\Delta>0))\text{xi}$$

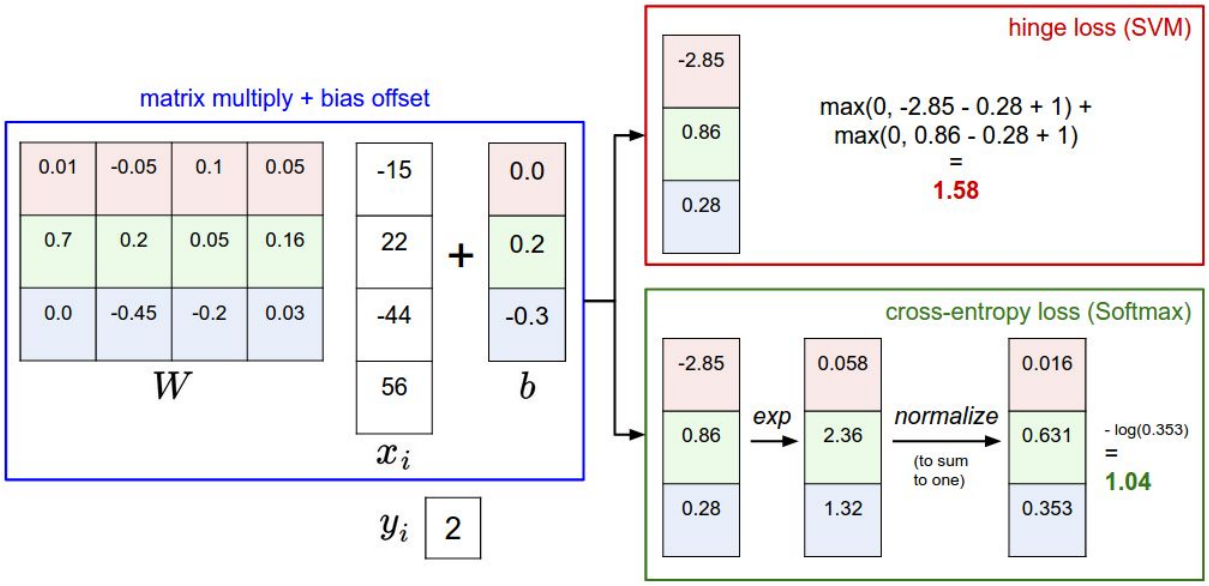
where 11 is the indicator function that is one if the condition inside is true or zero otherwise. While the expression may look scary when it is written out, when you're implementing this in code you'd simply count the number of classes that didn't meet the desired margin (and hence contributed to the loss function) and then the data vector $x_i$xi scaled by this number is the gradient. Notice that this is the gradient only with respect to the row of WW that corresponds to the correct class. For the other rows where $j\neq y_i$j≠yi the gradient is:

$$\nabla wjLi=1(wTjxi-wTyixi+\Delta>0)xi$$

dl/dw = x in case of any class

dl/dw = -x in case of the correclty classified class

SOFT MAX :



It turns out that the SVM is one of two commonly seen classifiers. The other popular choice is the **Softmax classifier**, which has a different loss function. If you've heard of the binary Logistic Regression classifier before, the Softmax classifier is its generalization to multiple classes. Unlike the SVM which treats the outputs $f(x_i,W)$f(xi,W) as (uncalibrated and possibly difficult to interpret) scores for each class, the Softmax classifier gives a slightly more intuitive output (normalized class probabilities) and also has a probabilistic interpretation that we will describe shortly. In the Softmax classifier, the function mapping $f(x_i;W)=Wx_i$f(xi;W)=Wxi stays unchanged, but we now interpret these scores as the unnormalized log probabilities for each class and replace the *hinge loss* with a **cross-entropy loss** that has the form:

$$L_i=-\log(e^{f_{y_i}}/\sum_j e^{f_j})$$

or equivalently$L_i=-f_{y_i}+\log\sum_j e^{f_j}$Li=−log(efyi∑jefj)or equivalently

Li=−fyi+log∑jefj

where we are using the notation $f_j$fj to mean the j-th element of the vector of class scores ff . As before, the full loss for the dataset is the mean of L_iLi over all training examples together with a regularization term $R(W)$R(W) . The function $f_j(z)=\frac{e^{z_j}}{\sum_k e^{z_k}}$fj(z)=ezj∑kezkfj(z)=ezj∑kezk is called the **softmax**

function: It takes a vector of arbitrary real-valued scores (in *z*z) and squashes it to a vector of values between zero and one. The full cross-entropy loss that involves the softmax function might look scary if you're seeing it for the first time but it is relatively easy to motivate.

Neural Networks:

- Model is two layer NN :
  Input Layer → First layer → relu → second layer → sigmoid → Output
- gradients with respect to loss is computed during backward propagation:

Extra Features:

Histogram of colours is used to add extra features for the classifiers we have already trained previously

In this histogram, we find the frequency of the ouccurences of the colors