Deep Learning for Advanced Robot Perception

Assignment 1: Linear Classification

Due Sunday, September 10th

This homework is a warm up for the rest of the course. As part of this homework you will:

- Implement a Multi-Class Support Vector Machine (SVM)
  - vectorized loss function
  - vectorized gradient computation

You will train the classifiers on images in the CIFAR-10 dataset. The CIFAR-10 is a toy dataset with 60000 images of size 32 X 32, belonging to 10 classes. You need to start with `svm.ipynb` first to implement the SVM

TODO

Download the starter code from week 1/homework1

Getting the dataset

Make sure you are connected to the internet. Navigate to the `f16RBE595/data` folder and run the following:

TODO

```
./get_datasets.sh
```

This script will download the python version of the database for you and put it in `f16RBE595/data/cifar-10-batches.py` folder

As you might already know, SVM classifier is a linear model similar to other classifiers such as SofMax or Stochastic Regression but they use different loss functions.

Here is a brief summary of the classifiers and if you need a detailed tutorial to brush up your knowledge, Stanford CS231 course is a nice place (http://cs231n.github.io/linear-classify/).

Before we go into the details of a classifier, let us assume that our training dataset consists of N instances $x_i \in R^D$ of dimensionality D. Corresponding to each of the training instances, we have labels $y_i \in 1,2,\ldots,K$ where K is the number of classes. In this homework, we are

using the CIFAR-10 database where $N=50,000$, $K=10$, $D=32\times32\times3$ (image of size 32×32 with 3 channels - Red, Green, and Blue).

Classification is the task of assigning a label to the input from a fixed set of categories or classes. A classifier consists of two important components:

Score function: This maps every instance x_i to a vector p_i of dimensionality K. Each of these entries represent the class scores for that image. Both SVM and Logistic Regression have a linear score function given by:

$$p_i=f(x_i;\ W,\ b)$$

where,

$$f(x;\ W,\ b)=Wx\ +\ b$$

Here, W is a matrix of weights of dimensionality $K\times D$ and b is a vector of bias terms of dimensionality $K\times1$. The process of training is to find the appropriate values for W and b such that the score corresponding to the correct class is high. In order to do this, we need a function that evaluates the performance. Using this evaluation as feedback, the weights can be updated in the right 'direction' to improve the performance of the classifier.
We make a minor modification to the notation before proceeding further. The bias term can be incorporated within the weight matrix W making it of dimensionality $K\times(D+1)$. The i^{th} row of the weight matrix W is represented as a column vector w_i so that $p^j_i=w^T_jx_i$. The superscript j denotes the j^{th} element of p_i, the score vector corresponding to x_i.

Loss function: This function quantifies the correspondence between the predicted scores and ground truth labels. The loss of the SVM is given by:

$$L=\frac{1}{n}\sum_{i=1}^{N}\sum_{j\neq y_i}\left[max\left(0,\ p_i^j-\ p_i^{y_i}+\ \Delta\ \right)\right]$$

Here, Δ is the margin. The loss function penalizes when the correct class is not greater than all the other scores by at least Δ.

If the weights are allowed to take values as high as possible, the model can overfit to the training data. To prevent this from happening a regularization term $R(W)$ is added to the

loss function. The regularization term is the squared some of the weight matrix W. Mathematically,

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

The regularization term $R(W)$ is usually multiplied by the regularization strength λ before adding it to the loss function. λ is a hyper parameter which needs to be tuned so that the classifier generalizes well over the training set.

The next step is to update the weight parts such that the loss is minimized. This is done by Stochastic Gradient Descent (SGD). The weight update is done as:

$$W := W - \eta \nabla L$$

Here, ∇L is the gradient of the loss function and the factor η is the learning rate. SGD is usually performed by computing the gradient w.r.t. a randomly selected batch from the training set. This method is more efficient than computing the gradient w.r.t the whole training set before each update is performed.