**2.1 Distance-Weighted K-Nearest Neighbors**

K-nearest neighbors classifier is a very intuitive algorithm and easy to implement. In training phase, we save both features *xTrain* and labels *yTrain* of the entire training set as our model. In classification phase, we assign the label by looking the most frequent class of the top *k* closest neighbors. Sometimes the frequencies of different classes could be tied, and one way to resolve this issue is to use the summation of “voting power” in replace of direct counting. Here, we use the Euclidean distance as our distance measure.

Consider the *k* nearest neighbors of test data **x**:

We define the “voting power” of each class *j* on test data **x** as the summation of the distance inverse of *k* nearest neighbors that have label *j*:

The class that possesses the highest voting power is chosen to be the label of the test data.

**2.2 Multinomial Logistic Regression**

In multinomial logistic regression, we want to classify the data into *k* classes with discrete label . For the case of CIFAR-10 *k* equals 10, and the likelihood of the particular class given specific weight is determined by:

In our implementation, we prepend the training and test data matrix with an “ones” column so that we can simplify

The conditional log likelihood can be expressed as

To maximize the conditional log likelihood of the weight vector **w**, we take partial derivative on each dimension *j* and each class c:

where

By gradient ascent, we can approximate the optimal solution by iteration:

**3.1 Soft-Margin Dual Support Vector Machine**

The SVM problem can be solved in primal form or dual form using quadratic programming technique. Here, we exploit the function *quadprog* in MATLAB, which has the form:

The optimal solution can be obtained by calling

In this report, we use soft-margin SVM to circumvent the cases that are not linearly separable. Compared to the original SVM, the soft-margin SVM is to allow mis-classification within a small error . The error will be regularized by a user-defined parameter C.

In the form of primal SVM:

In the form of dual SVM:

Since it uses less parameter in dual SVM, plus the kernel trick is applicable here, we choose soft-margin dual SVM as our main algorithm. In training phase, we use *quadprog* to find the optimal value . In classification phase, we classify new data by computing

Where b is a bias vector, which serves to translate the hyperplane and is computed as:

For multi-class SVM, we use the “*on-vs-rest,*” which independently trains SVM for each class, do the classification for “is this class” and “not this class.” For 10 classes, we will have 10 different sets of support vectors, corresponding to each different class.