Yerbol Aussat

k-NN algorithm on MNIST data set

k-NN algorithm was implemented in Python as follows.

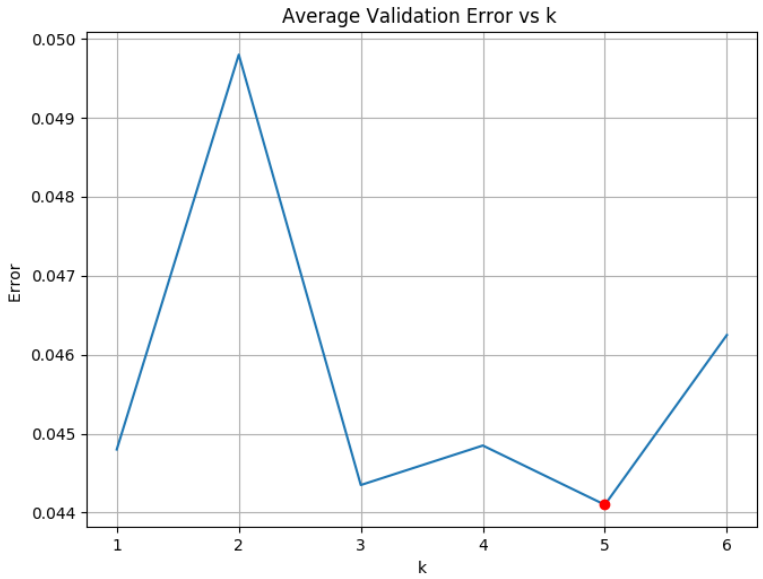
```
def kNN(X_train, y_train, X_test, k):
    y_pred = []
    start_time = time.time()

    for t in range(X_test.shape[0]):
        x0 = X_train - X_test[t]
        D_t = np.sum(x0**2,axis=-1)**(1./2)
        k_smallest = np.argpartition(D_t, k-1)[:k]  # linear time at the worst case
        y_k_smallest = y_train[k_smallest]
        y_prediction = collections.Counter(y_k_smallest).most_common()[0][0]
        y_pred.append(y_prediction)
    print "prediction done in ", time.time() - start_time, "sec"
    return y_pred
```

Matrix-matrix multiplication was used instead of using for-loops for iterating through the training samples. Also, for finding k smallest distances to the test sample, *numpy.argpartition* function was used. It's complexity is $O(n)$.

The complexity of cross-validation is $O(n^2)$, because the size of validation sets is a function of n (specifically it is $\frac{n}{10}$). It was confirmed experimentally be measuring the amount of time kNN takes.

Due to the lack of time I used only one third of the training data for cross-validation for selecting the best k.



The plot above shows average validation error for the values of k from 1 to 6. Based on these results, **k=5** was selected for making prediction on the test set.

Test error for k = 5 turned out to be `0.0306`, or **3.06%.**