

```
[info] ROOT=/home/kim1484/assignment1_colab/assignment1  
[info] DATA_DIR=/home/kim1484/assignment1_colab/assignment1/cs231n/datasets  
[ready] 환경 준비 완료.
```

k-Nearest Neighbor (kNN) exercise

Complete and hand in this completed worksheet (including its outputs and any supporting code outside of the worksheet) with your assignment submission. For more details see the [assignments page](#) on the course website.

The kNN classifier consists of two stages:

- During training, the classifier takes the training data and simply remembers it
- During testing, kNN classifies every test image by comparing to all training images and transferring the labels of the k most similar training examples
- The value of k is cross-validated

In this exercise you will implement these steps and understand the basic Image Classification pipeline, cross-validation, and gain proficiency in writing efficient, vectorized code.

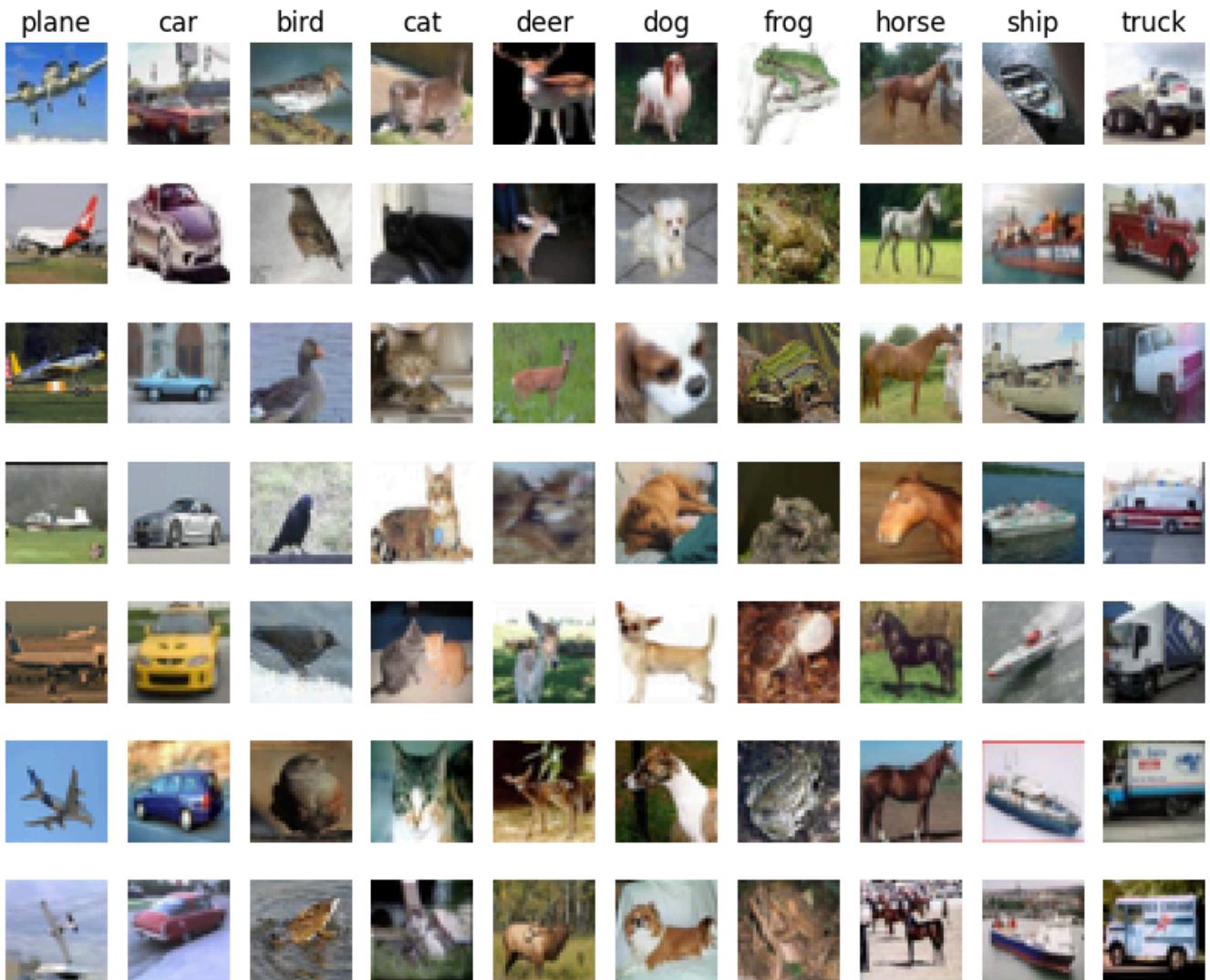
```
[ready] Visualization & autoreload settings complete.
```

```
Training data shape: (50000, 32, 32, 3)
```

```
Training labels shape: (50000,)
```

```
Test data shape: (10000, 32, 32, 3)
```

```
Test labels shape: (10000,)
```



(5000, 3072) (500, 3072)

We would now like to classify the test data with the kNN classifier. Recall that we can break down this process into two steps:

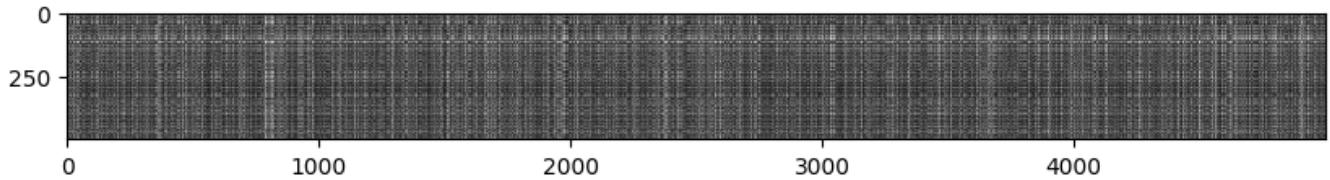
1. First we must compute the distances between all test examples and all train examples.
2. Given these distances, for each test example we find the k nearest examples and have them vote for the label

Lets begin with computing the distance matrix between all training and test examples. For example, if there are **Ntr** training examples and **Nte** test examples, this stage should result in a **Nte x Ntr** matrix where each element (i,j) is the distance between the i-th test and j-th train example.

Note: For the three distance computations that we require you to implement in this notebook, you may not use the np.linalg.norm() function that numpy provides.

First, open `cs231n/classifiers/k_nearest_neighbor.py` and implement the function `compute_distances_two_loops` that uses a (very inefficient) double loop over all pairs of (test, train) examples and computes the distance matrix one element at a time.

(500, 5000)



Inline Question 1

Notice the structured patterns in the distance matrix, where some rows or columns are visibly brighter. (Note that with the default color scheme black indicates low distances while white indicates high distances.)

- What in the data is the cause behind the distinctly bright rows?
- What causes the columns?

Your Answer : The bright rows indicate that the i -th test sample has large distance from most training samples. While bright columns mean that the j -th train sample is far from most testing samples.

Got 137 / 500 correct => accuracy: 0.274000

You should expect to see approximately 27% accuracy. Now lets try out a larger `k`, say `k = 5`:

Got 139 / 500 correct => accuracy: 0.278000

You should expect to see a slightly better performance than with `k = 1`.

Inline Question 2

We can also use other distance metrics such as L1 distance. For pixel values $p_{ij}^{(k)}$ at location (i, j) of some image I_k ,

the mean μ across all pixels over all images is

$$\mu = \frac{1}{nhw} \sum_{k=1}^n \sum_{i=1}^h \sum_{j=1}^w p_{ij}^{(k)}$$

And the pixel-wise mean μ_{ij} across all images is

$$\mu_{ij} = \frac{1}{n} \sum_{k=1}^n p_{ij}^{(k)}.$$

The general standard deviation σ and pixel-wise standard deviation σ_{ij} is defined similarly.

Which of the following preprocessing steps will not change the performance of a Nearest Neighbor classifier that uses L1 distance? Select all that apply. To clarify, both training and test examples are preprocessed in the same way.

1. Subtracting the mean μ ($\tilde{p}_{ij}^{(k)} = p_{ij}^{(k)} - \mu$.)
2. Subtracting the per pixel mean μ_{ij} ($\tilde{p}_{ij}^{(k)} = p_{ij}^{(k)} - \mu_{ij}$.)
3. Subtracting the mean μ and dividing by the standard deviation σ .
4. Subtracting the pixel-wise mean μ_{ij} and dividing by the pixel-wise standard deviation σ_{ij} .
5. Rotating the coordinate axes of the data, which means rotating all the images by the same angle. Empty regions in the image caused by rotation are padded with a same pixel value and no interpolation is performed.

Your Answer : 1, 2

Your Explanation : Because L1 distance is based on absolute difference, subtracting the mean does not change the distances between data points. Which means there will be no changes in the performance of a Nearest Neighbor classifier

One loop difference was: 0.000000
Good! The distance matrices are the same

(500, 5000)
No loop difference was: 0.000000
Good! The distance matrices are the same

Two loop version took 11.938408 seconds
One loop version took 23.488079 seconds
No loop version took 0.082033 seconds

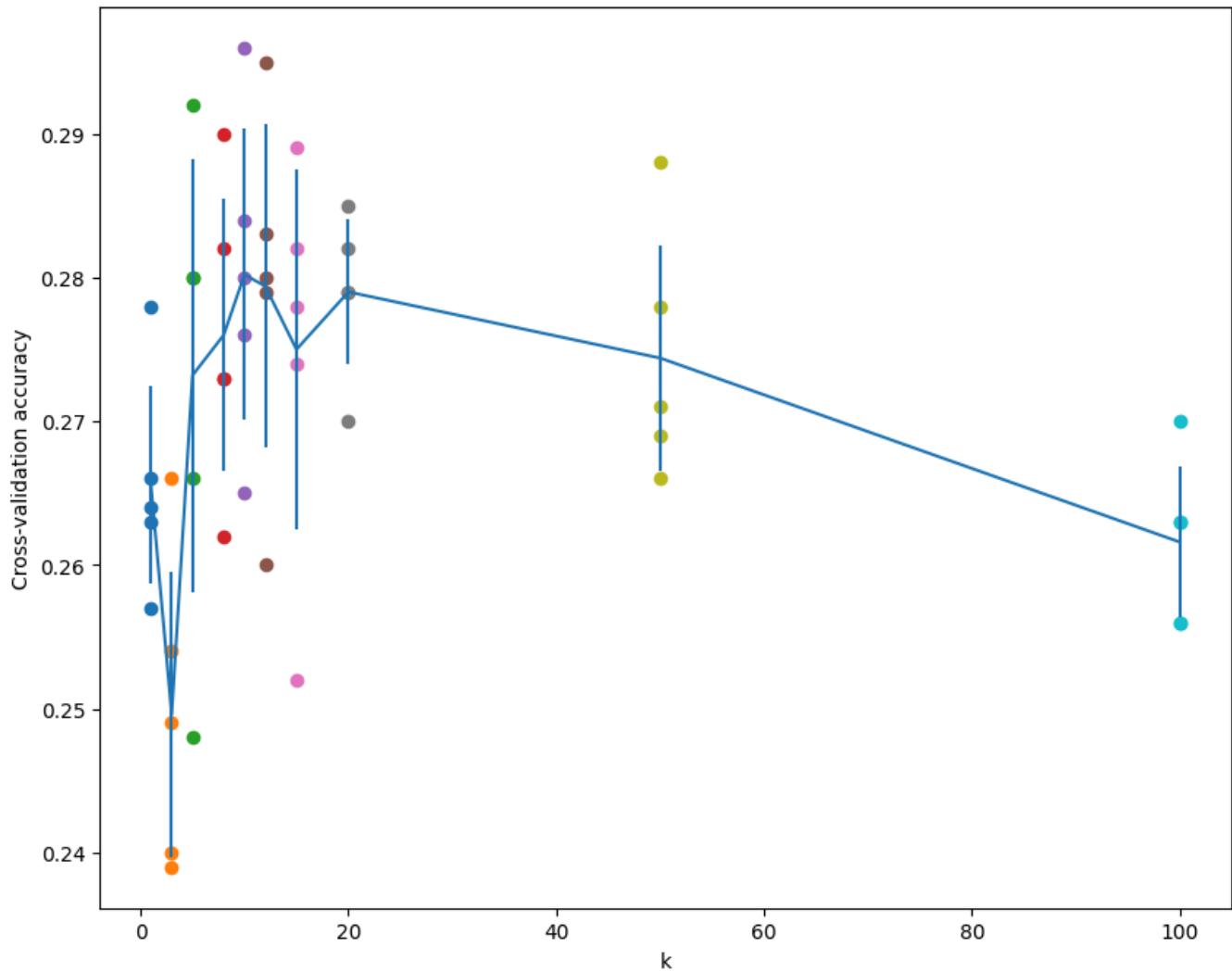
Cross-validation

We have implemented the k-Nearest Neighbor classifier but we set the value $k = 5$ arbitrarily. We will now determine the best value of this hyperparameter with cross-validation.

```
{1: [0.263, 0.257, 0.264, 0.278, 0.266], 3: [0.239, 0.249, 0.24, 0.266, 0.254], 5:  
[0.248, 0.266, 0.28, 0.292, 0.28], 8: [0.262, 0.282, 0.273, 0.29, 0.273], 10: [0.265,  
0.296, 0.276, 0.284, 0.28], 12: [0.26, 0.295, 0.279, 0.283, 0.28], 15: [0.252, 0.289,  
0.278, 0.282, 0.274], 20: [0.27, 0.279, 0.279, 0.282, 0.285], 50: [0.271, 0.288, 0.27  
8, 0.269, 0.266], 100: [0.256, 0.27, 0.263, 0.256, 0.263]}  
k = 1, accuracy = 0.263000  
k = 1, accuracy = 0.257000  
k = 1, accuracy = 0.264000  
k = 1, accuracy = 0.278000  
k = 1, accuracy = 0.266000  
k = 3, accuracy = 0.239000  
k = 3, accuracy = 0.249000  
k = 3, accuracy = 0.240000  
k = 3, accuracy = 0.266000  
k = 3, accuracy = 0.254000  
k = 5, accuracy = 0.248000  
k = 5, accuracy = 0.266000  
k = 5, accuracy = 0.280000  
k = 5, accuracy = 0.292000  
k = 5, accuracy = 0.280000  
k = 8, accuracy = 0.262000  
k = 8, accuracy = 0.282000  
k = 8, accuracy = 0.273000  
k = 8, accuracy = 0.290000  
k = 8, accuracy = 0.273000  
k = 10, accuracy = 0.265000  
k = 10, accuracy = 0.296000  
k = 10, accuracy = 0.276000  
k = 10, accuracy = 0.284000  
k = 10, accuracy = 0.280000  
k = 12, accuracy = 0.260000  
k = 12, accuracy = 0.295000  
k = 12, accuracy = 0.279000  
k = 12, accuracy = 0.283000  
k = 12, accuracy = 0.280000  
k = 15, accuracy = 0.252000  
k = 15, accuracy = 0.289000  
k = 15, accuracy = 0.278000  
k = 15, accuracy = 0.282000  
k = 15, accuracy = 0.274000  
k = 20, accuracy = 0.270000  
k = 20, accuracy = 0.279000  
k = 20, accuracy = 0.279000  
k = 20, accuracy = 0.282000  
k = 20, accuracy = 0.285000  
k = 50, accuracy = 0.271000  
k = 50, accuracy = 0.288000  
k = 50, accuracy = 0.278000  
k = 50, accuracy = 0.269000  
k = 50, accuracy = 0.266000  
k = 100, accuracy = 0.256000  
k = 100, accuracy = 0.270000  
k = 100, accuracy = 0.263000  
k = 100, accuracy = 0.256000  
k = 100, accuracy = 0.263000
```

[0.2656 0.2496 0.2732 0.276 0.2802 0.2794 0.275 0.279 0.2744 0.2616]

Cross-validation on k



Got 141 / 500 correct => accuracy: 0.282000

Inline Question 3

Which of the following statements about k -Nearest Neighbor (k -NN) are true in a classification setting, and for all k ? Select all that apply.

1. The decision boundary of the k -NN classifier is linear.
2. The training error of a 1-NN will always be lower than or equal to that of 5-NN.
3. The test error of a 1-NN will always be lower than that of a 5-NN.
4. The time needed to classify a test example with the k -NN classifier grows with the size of the training set.
5. None of the above.

Your Answer : 2, 4

Your Explanation :

1. **False:** The decision boundary of the k-NN classifier is determined by the data distribution, which is not linear.
2. **True:** If the k-NN classifier is trained with $k=1$, it will be overfitted to the training data. Therefore, the training error of 1-NN will always be lower than or equal to that of 5-NN.
3. **False:** A 1-NN classifier tends to overfit to the training data and does not generalize well to unseen test data. Therefore, the test error of 1-NN is higher than that of 5-NN.
4. **True:** Since the distance matrix must be computed between the training and test data, the computation time increases proportionally with the size of the training set.