Weekly Report(Sep.17,2018-Sep.22,2018)

Zhang Yuandi
15826138027@163.com

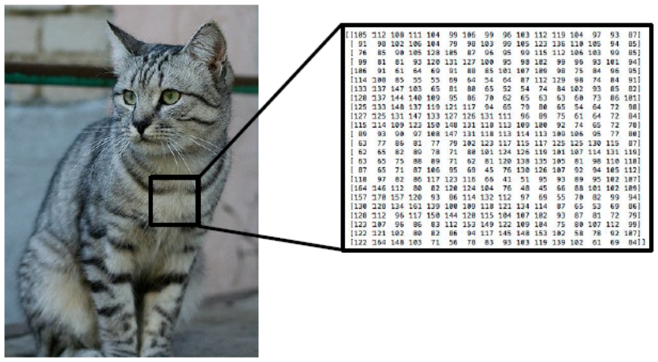
Abstract

Sorry for a long time suspension of learning. In the last week, I learned lecture1 to lecture5 of **CS231n** and week1 of **Neural Networks for Machine Learning**. Because week1 is the introduction part which can be seen also in CS229, so I didn't write them in this paper.

1 CS231n

1.1 Image Classification

When we see a picture, our brain can tell us what is it. For example, when we see this image below, we say it is a cute cat. But our computer doesn't work like this. What it sees are pixels. So, if we want computer to do image classification, we need to help computer to learn classification through different pixels of different objects.



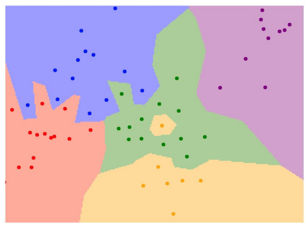
```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

Unlike sorting a list of numbers, there are no obvious ways to hard-code the algorithm for recognizing a cat, or other classes. Thus, we seek from Data-Driven Approach:

1. Collect a dataset of images and labels.
2. Use Machine Learning to train a classifier.
3. Evaluate the classifier on new images.

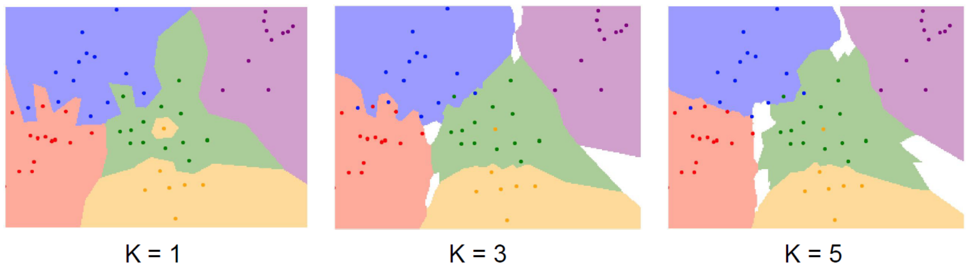
1.1.1 Nearest Neighbor

We can try this simple way to do classification, which defines the nearest neighbors are of the same kind. By memorizing all data and labels, it can predict the label of the most similar training image. However, this method is not so good like what this image below shows. We can see there is a yellow part in the center of green part, and those boundaries seem like fingers, which all indicate incorrectness.



1.1.2 K-Nearest Neighbors

Instead of copying label from nearest neighbor, take majority vote from K closest points. When K gets larger, the outcome looks better.



1.1.3 Distance Metric

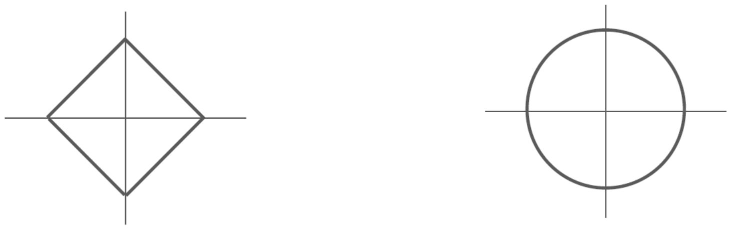
We can measure distance by two ways. One is L1 (Manhattan) distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|,$$

the other is L2 (Euclidean) distance:

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}.$$

The picture below shows what these distance mean, the left is L1 distance and the right is L2 distance.



1.1.4 Setting Hyperparameters

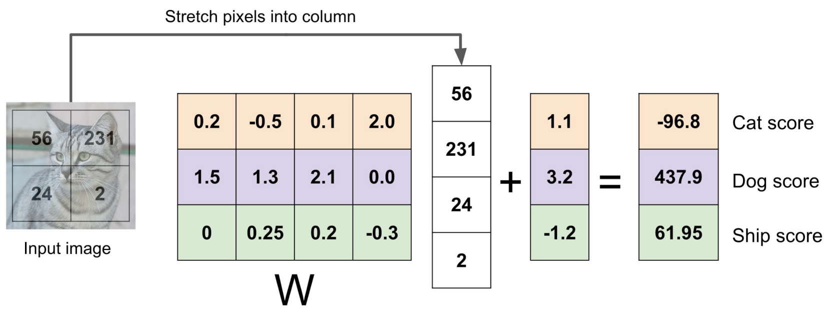
When we set our program, we always need to know what is the best value of k to use and what is the best distance to use. These are hyperparameters problems, that is, choices about the algorithm that we set rather than learn. Our idea is Cross-Validation: split data into folds, try each fold as validation and average the results.

But actually, k-Nearest Neighbor on images is never used. Because it is very slow at test time and distance metrics on pixels are not informative. And curse of dimensionality will result in huge amount of calculation. So, let's try parametric approach: Linear Classifier.

1.2 Linear Classifier

What we'll do can be illustrated like:

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



Let's define a loss function that quantifies our unhappiness with the scores across the training data and come up with a way of efficiently finding the parameters that minimize the loss function.

1.2.1 Multiclass SVM Loss

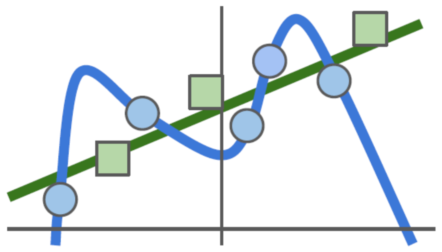
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_j; W)_{y_i} + 1)$$

Suppose that we found a W such that $L = 0$. Is this W unique? The answer is not. $2W$ is also has $L = 0$. But what if another W_i that doesn't have linear correlation with W? Is there any possibility?

1.2.2 Regularization

But in real time we sometimes get this outcome:



We get the blue output in our training data, however, our test data may distribute like those green points. So we apply regularization to our loss function.

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

1.3 Softmax Classifier

The loss function of softmax classifier is

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

1.4 Optimization

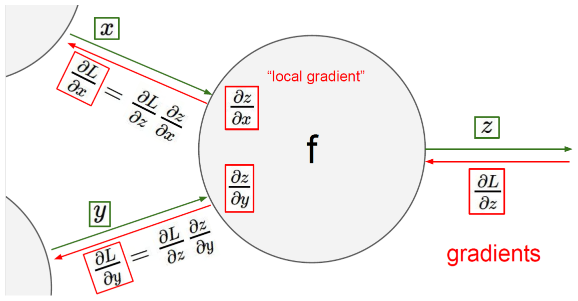
When we do optimization, the suggested way is always using analytic gradient and checking implementation with numerical gradient. This is called a gradient check.

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

But as there are so many data, it is impossible for us to write down every gradient. Thus, backpropagation is essential.

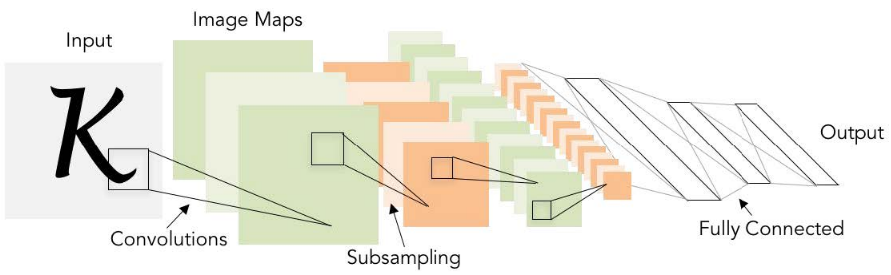
1.5 Backpropagation

Backpropagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. The image below shows its process generally.


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

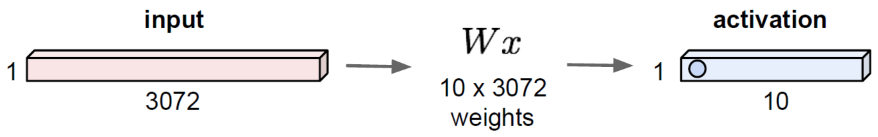
1.6 Convolutional Neural Networks

There is a long history of CNN, and scientists pay more attention to it nowadays.



1.6.1 Fully Connected Layer

For an example, if we get a $32 \times 32 \times 3$ image, we can stretch it to 3072×1. And 1 number is the result of taking a dot product between a row of W and the input.



But to preserve spatial structure, we use convolution layer.

1.6.2 Convolution Layer

Instead of stretching the image, we keep the structure. And we use a 3D filter to convolve, slide over the image spatially and compute dot products. And the 1 number is the result of taking a dot product between the filter and a small chunk of the image. And we know the output size is equal to

(N - F) / stride + 1. To maintain the size of output, we do zero padding. But it is confusing. Doesn't it change the information of the image?

2 Plans for Next Week

Well, I really need to think about how to adjust my time for my major assignments are so many that I'm afraid that I'll fail them if not spent time appropriately.

1. Learn lecture6 to lecture10 of **CS231n**.
2. Learn week2 and week3 of **Neural Networks for Machine Learning**.