Lecture 2 : Image Classification

강승준

Basic code :

<training> -> usage

def train(images, labels): def predict(model, test\_images):

return model return test\_lables

How to train from training part :

One possible way : **Nearest Neighbor**

Nearest Neighbor : would find the distance between two images. \* optimize with diff

도표이(가) 표시된 사진

자동 생성된 설명 Two possible ways to calculate distance.

**Abstract of working procedure** :

Each test image is given to the trained model

By model it calculates from the template with diff function above(L1 ,L2)

By comparing distances, it choose the closest distance and shows the label of it

Fall back : since the template(benchmark) is trained in o(1) (single template)

We have to compare for N classes -> o(n)

: **fast at learning But slow at prediction : BAD**

**K-NN vs 1-NN**

In 1 -NN we used o(1) to find “single closest” label

차트, 도표이(가) 표시된 사진

자동 생성된 설명 However in K-NN => vote for around distance

:: if we use L1 : it would calculate all the vertical+horizontal distance same object.

:: if we use L2 : it would find the “circle” position in the base point.

**Since comparing just “pixel values” => weak for deformation**

**About KFOLD – cross validation.**

테이블이(가) 표시된 사진

자동 생성된 설명 We FOLD the dataset to avoid overfitting :: saves the data for test ( untrained data set)

**Cross validation :**

In this picture) we train 3 models initialized with start status.

**After training different number of folds**

Model 1 : fold1, fold2 ,fold3, fold4

Model2 : fod1, fold2, fold3, fold5

We use model.predict(test) to get the scores of each model.

And by using mean of the score we get **the average score => can be used for prediction.**

Linear Classifier

Best picture to understand linear classifier

텍스트이(가) 표시된 사진

자동 생성된 설명S

W : is the Weight matrix, each indicates the weight of pixel.

X : is the input

도표이(가) 표시된 사진

자동 생성된 설명 B : is the bias for model.

Characteristics of linear classification

Predictions are linear

텍스트, 고양이, 보기, 실내이(가) 표시된 사진

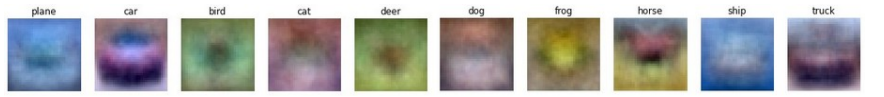
자동 생성된 설명F(x,w) == F(cx,w) => it will choose the same category.

Same Cateogry : “CAT”

**What does “W” means in “spatial information”**

W is a matrix of weights to find the specific category.

* It emphasizes the “edges”(or same lines of the training images)
* To maximize the scores when same edges are in the input.

**Works as “TEMPLATE” of category**

Geometric view point

차트이(가) 표시된 사진

자동 생성된 설명Characteristics

1. The spatial separation is divided with **plane**

**Because it is LINEAR CLASSIFICATION**

1. In 3 category classification 3 planes overlaps each other.

차트이(가) 표시된 사진

자동 생성된 설명By KNOWING this we could understand why these examples are hard to separate.

There are “two” classes to divide. => we have two planes to use

It is impossible to divide with two planes.

How to choose W

We use loss function to choose w

loss for single pixel

텍스트, 손목시계이(가) 표시된 사진

자동 생성된 설명

loss for dataset

Multiclass SVM LOSS

: the score of the correct class should be higher than all the other scores

도표이(가) 표시된 사진

자동 생성된 설명so if the score of correct class is higher than other category’s highest class + margin we set the loss as 0

this shape is called

Hinge

텍스트이(가) 표시된 사진

자동 생성된 설명

Possible outcomes

1. If(sj > sy) we get the loss
2. If(sj < sy+1) we get o for the cost

* Characteristics

Little change of score doesn’t affect category selection

Min / Max : 0~INF

Random score ) what would be the loss : ~~ 1 \* (n -1)

What happens if we add all classes (including I =yi) : inflated by 1

W == 2W results are the same 🡺 How do we choose???

Regularization텍스트, 편지이(가) 표시된 사진

자동 생성된 설명

Why do we use regularization?

* 텍스트, 편지이(가) 표시된 사진

  자동 생성된 설명Expressing Preference

In the case above the result of the model will be same.

But by adding L2 regularization the out put with w2 would get the small loss

* Using L2 we can hint the model to use all the inputs.
* 텍스트이(가) 표시된 사진

  자동 생성된 설명Using L1 we can hint the model to choose the single point important for the input.
* Prefer simpler model

By using regularization,( L1 , L2 ) in this case, the overfitted line will get big Regularization values -> leading to high losses

Another Loss function

텍스트이(가) 표시된 사진

자동 생성된 설명 Cross Entropy Loss

What it does

1. Get the scores of the model
2. Use exp to make all the values positive
3. Use normalization range (0~1) => we get the probabilities

도표이(가) 표시된 사진

자동 생성된 설명

Lecture 4 Optimization

Basic question : how could we optimize the Loss of the function?

Numeric Gradient & Analytical Gradient( will be discussed in lecture 6)

Numeric Gradient :

Get the gradient of single pixels weight each.

By doing this we could identify the slope of the multi dimension

In usage : we use both to compare the gradient we are using is right

블러이(가) 표시된 사진

자동 생성된 설명 Ex) gradgradcheck();

Hyper parameters

Weight initialization :: where to start from the slop

Number of steps :: how many times the position moves

Learning rate :: how far it will go on each step

텍스트이(가) 표시된 사진

자동 생성된 설명

So how do we optimize?

**Using SGD**

텍스트이(가) 표시된 사진

자동 생성된 설명

We use gradients descent by using minibatch( sampled data ) not the whole data.

텍스트, 편지이(가) 표시된 사진

자동 생성된 설명cant understand this part.

Problems of SGD

1. If the gradient of the weight is different significantly -> it vibrates.

* Very slow progress

1. Saddle point

* SGD stops optimizing

텍스트이(가) 표시된 사진

자동 생성된 설명To solve this problem, we use Momentum

V is velocity for moving

Rho : used to get the before velocity

By using this two(gradient, velocity)

We can pass the saddle point & local minimum

Momentum Vs nestrov Momentum

차트, 도표, 다각형이(가) 표시된 사진

자동 생성된 설명In netrov momentum,

We starts at the same point and move to historic velocity first. And at that position, we calculate the gradient of the position and add to the velocity vector.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명AdaGrad

Grad part is + with dw^2

This leads to accumulating the grad\_squared.

In flat region the grad\_squared.sqrt() is small -> leads to acceleration.

In steep region the grad\_squared.sqrt() is larger -> leads to slow down.

As the AdaGrad moves, the grad\_squared is accumulated

Leading to smaller steps / decaying learning rate

Adam

테이블이(가) 표시된 사진

자동 생성된 설명

In Adam we use RmsProp + momentum

RMSProp : solves the problem of Adagrad : the accumulation of the momentum.

By making grad\_squared to reflect the decay rate -> makes even speed.