14
8-9-2021

A
Tuesday

Week #3 Lecture #3

Image classification

K-Nearest Neighbors:

Hyperparameters:

1. what is the best value of k to use?
2. what is the best distance to use?

choices about the algorithms themselves

these values vary from dataset to dataset

L_2 (Euclidean distance) $= \sum (I_1^2 - I_2)^2$

Problem Dependent

Setting hyperparameters:-

Idea #1:-

choose h.p that work best on data

[Dataset]

Bad: K=1 always works perfectly on training data

accuracy = 100% overfitting

Overfitting:
seen data ie good results
unseen data ie not good results at all

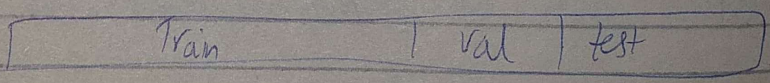
Idea #2.

split data into train & test, choose hif that work best on test data

Take odd numbers of k starting from 1, 3, 5, 7...

still overfitting occurs bcz we are use test data set for our training our k value we won't use test set in future we will only use it once at the end after training. To make it unseen.
No Idea how algo will perform on data

Idea #3:
Single split validation
split data into train, val, & test; choose hif on val & evaluate on test Better!



Take one value of K. lets say 3

Take one Pic from val set,

Take distance from all train set images.

Choose K Nearest neighbours

give most class in these K neighbors to val image

our accuracy will be matching our actual class from the Predicted class.

if accuracy is highest Compared to other values of K, then check for this K on test set.

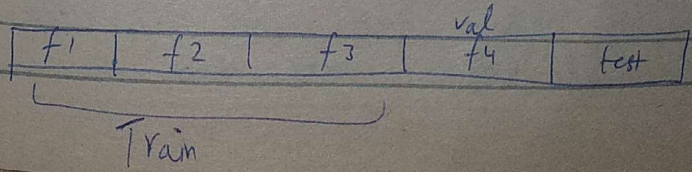
Problem in this:

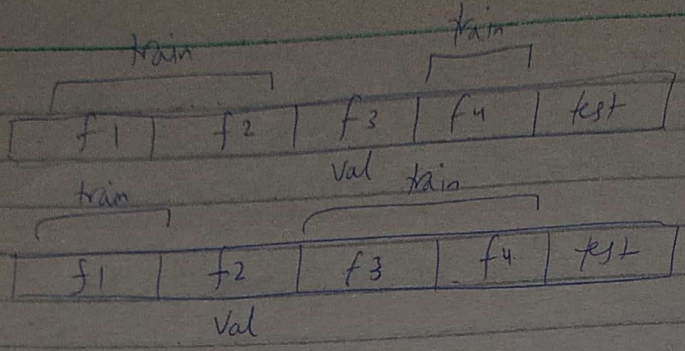
if dataset set is small training won't be accurate so we don't use single split validation.

Idea#4:-

Cross validation

split data in folds; try each fold as validation & average the results





for checking each fold as val set & other as train set we set K as one number & check for each fold & average the results of accuracy

e.g. for K=1

70 %
71 %
72 %
72 %
71 %

avg 71.5 for K=1
accuracy

useful for small datasets but not used too frequently in deep learning.

Computationally expensive

Problem:

KNN with pixel distance never used.

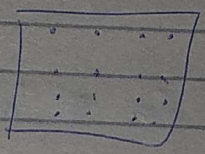
Augmentation = rotation of original Images, increase in dataset variation Increases

A

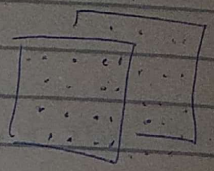
- distance metrics on Pixels are not informative
- very slow at test time.
- curse of Dimensionality

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2

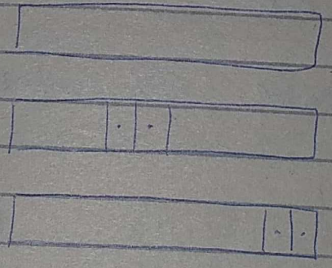


Dimensions = 3
Points = 4^3



$32 \times 32 \times 3 = 3072$
Total Pixels

more dimensions



distance same from original Image but both Images are shifted or tilted & different from each other as well.

Linear classifier.

we don't save whole training dataset, we take one Image

Parametric Approach:

as input & run it through a function

Image → $f(x, W)$ → N numbers giving class scores

W
Parameters
or weights

if 10 classes, it will give 10 Numbers, 1 score for each class

max score will be given to that image

every pixel will be given weights

$f(x, W) = Wx$

Array of $32 \times 32 \times 3$ numbers (3072 numbers total)

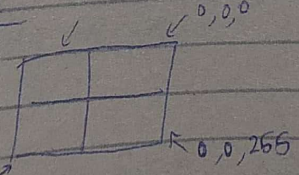
$\underset{x}{3072 \times 1} = 10 \times 1$

$(10 \times 3072)(3072 \times 1) = 10 \times 1$
$W \quad x \quad Wx$

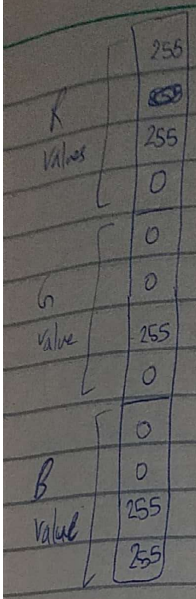
3072×1
$1 \times 3072 \times 10$

Dimensions

for example-



$2 \times 2 \times 3 = 12$



Column vector Dimensions 12x1

we want our answer to be column vector nx1 where n is no. of classes we want to predict.

$$\underset{a \times c}{10 \times 1} = \underset{a \times b}{(10 \times 3072)} \underset{b \times c}{(3072 \times 1)}$$

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

10×1 bias vector

small task:

$n = 5$ classes

$16 \times 16 \times 3 = 256 \times 3 = 768 \times 1$

$$\underset{f}{5 \times 1} = (5 \times 768)(768 \times 1) + (768 \times 1)$$
$$f = Wx + b$$

Now for above example we get Wx as

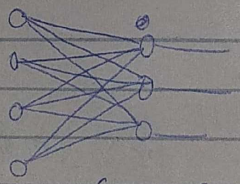
10		1		11
10		2		12
8	+	0.5	=	8.5
9		1		10
7		0.5		7.5
Wx		bias b		

if class have more example in dataset, then it has more ~~bias~~ multiple

we use linear classifier to make NN

~~Example~~ more weight, this pixel is more imp

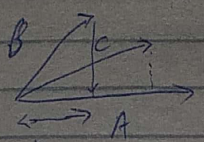
class1	0.2	-0.5	0.1	2.0
class2				0.0 → not imp at all
class3				
class4				



most score after function, then that score gives ans to that image as its class

we are checking cosine similarity.

$A \cdot B = A_x B_x + A_y B_y$ dot Product.



distance A, C

$A \cdot B < A \cdot C$

if we visualize weights, then they would be blurry versions of actual classes

we learn generic image of each class