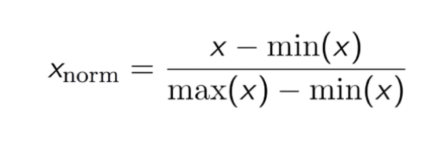
**[Feature Rescaling]**

- there are lots of data having several columns which have different scale respectively.

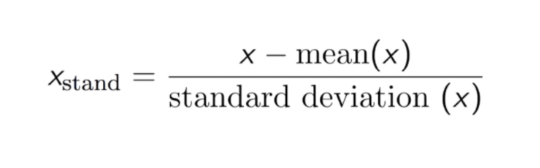
- we often encounter to use Euclidean distance to make a model, but in that case, different scale between column would occur issue, since model would give weight to columns differently.

- so we should do feature rescaling, and there are two method to rescaling ususally used.

<Normalization>



<Standardisation>



- Normalization will genereate smaller standard deviations then standardization. that is, the data are more concentrated around mean. but normalization doesn’t handle outlier well.

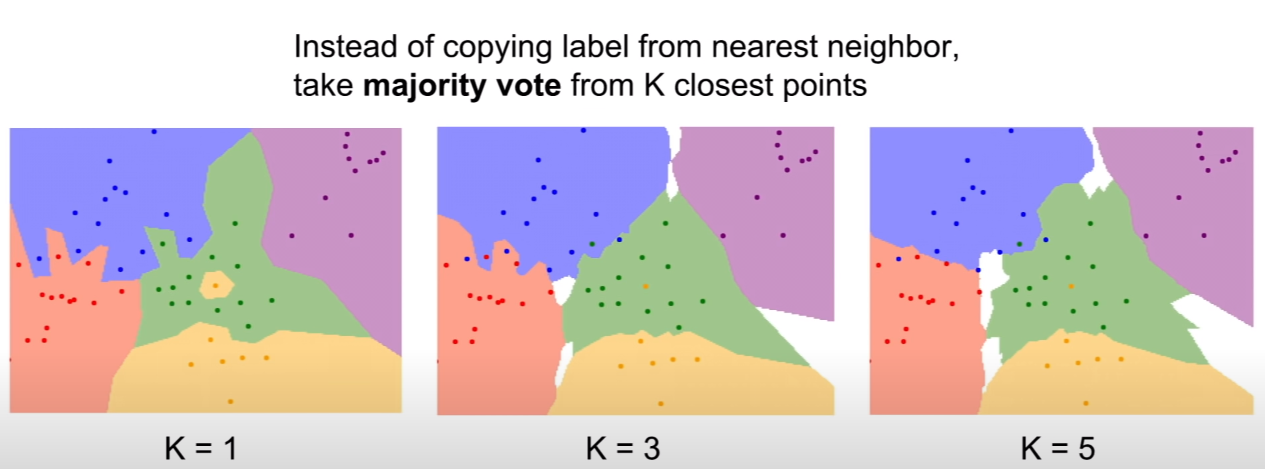
- Standardisation is more robust to outliers. Totally, it is preferable to use Standardisation over nomalization.

**[K nearest neighbor]**

- the value k is how many nearest point we will consider to determine the current value’s cluster.

- as you can see the blow picture, when k = 1, there are some noise like a yellow point in the middle of green sector.

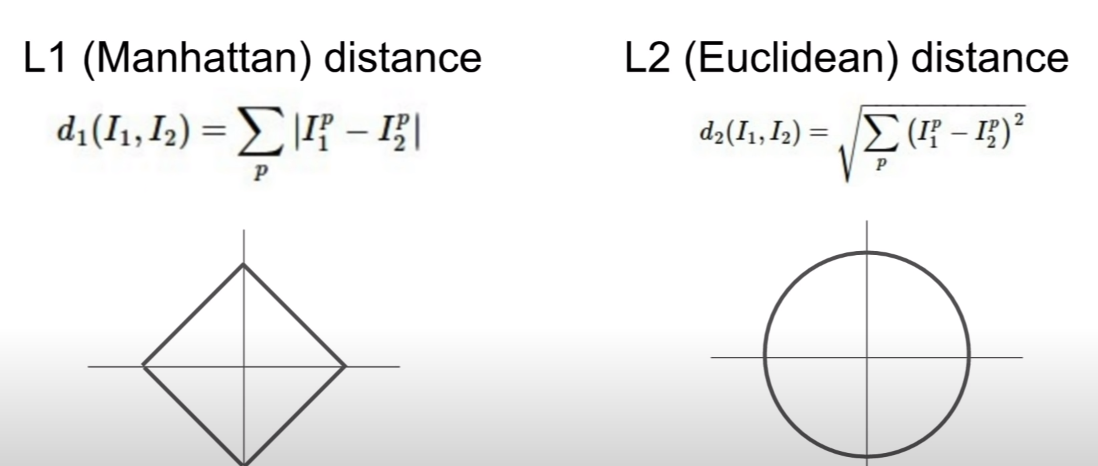
- but by taking larger k value, and taking majority vote, we can avoid those noises.



- In this view point, when we do obeject classification, it is better to take top k’s label than just take only top 1’s label.



[L1 and L2 distance]



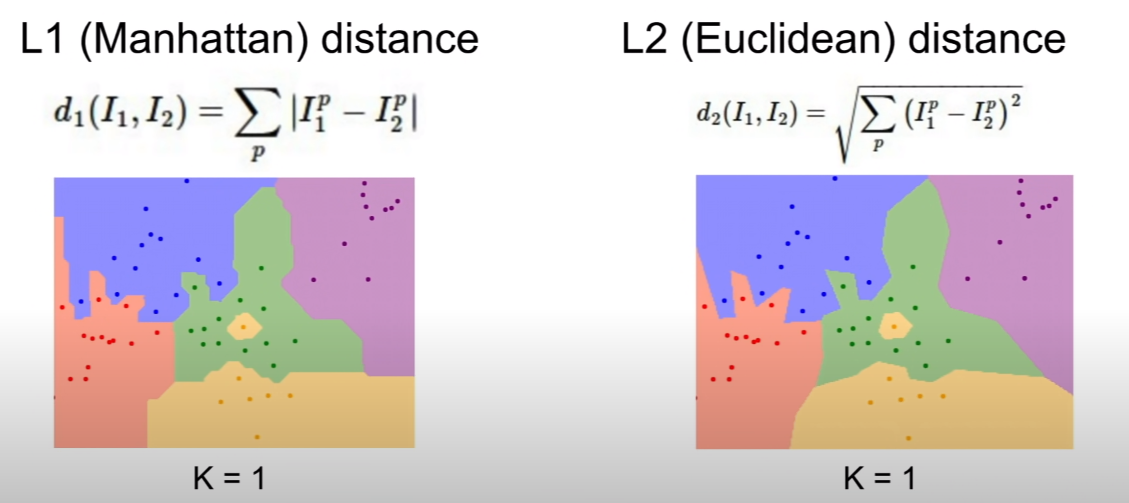
- L1 distance called Manhattan distance is just sum of all the point’s distance in matrix, and compare to label.

- L2 distance called Euclidean distance is sum of all the point’s distance using square and take root.

- so by rotating the coordinate system, we can get different distance from L1 distance.

- as a result, if we know which is an important feature in our record(entries) we can use L1 distance to give different weight. Otherwise, I mean, if we don’t know which one is important or we think they have similar importance, then we can use L2 distance.

- in addition, the PHD (lecturer) said, if we know a feature’s meaning, I mean, a column’s meaning in given data set, we should use L1 distance, but if they don’t have specific meaning, just taking L2.



- the above ficture is when we apply both distance algorithm to K nearest neighbor.

- as we can see, L1 distance make more corrodinate friendly, I mean, the boundaries tend to follow corrodinate axes, but L2 distance doesn’t care the axes.

- we can test and see how k nearest neighbor works depending on k using below URL.

> http://vision.stanford.edu/teaching/cs231n-demos/knn/

[hyperparameter]

- hyperparameter is paremeters we can choose ahead of training model such as k in k nearest neighbor.

- hyperparemeter is very problem-dependant, and ususally we could get a valud of a hyperparemeter heuristically.

- we shouldn’t take a value of hyperparameter to fit train data, since it could cause overfitting.

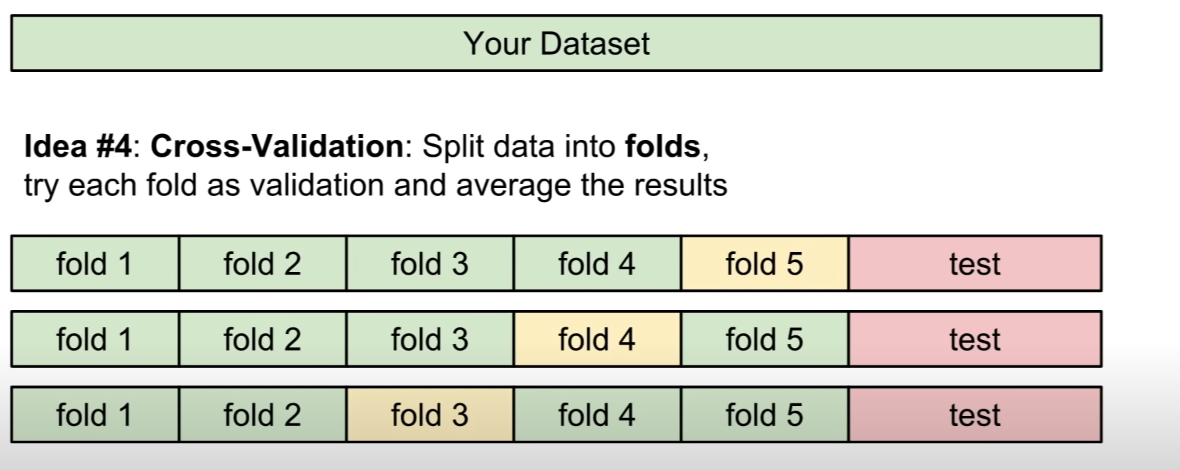
- next idea is to split our traing set to train and valid set, and choose the best value which works best on valid data. but we also shouldn’ take this method. since in this method, hyperparemeter will care our valid data, but not new test data that is unseen data.

- so the best method is to split the data into three different set such as training, validation and test.

> after spliting, trying many cases, I mean, several hyperparameter on training set, and evaluate them on validation set.

> now, pick the best performed hyperparemter and apply it to test set which is unseen data.

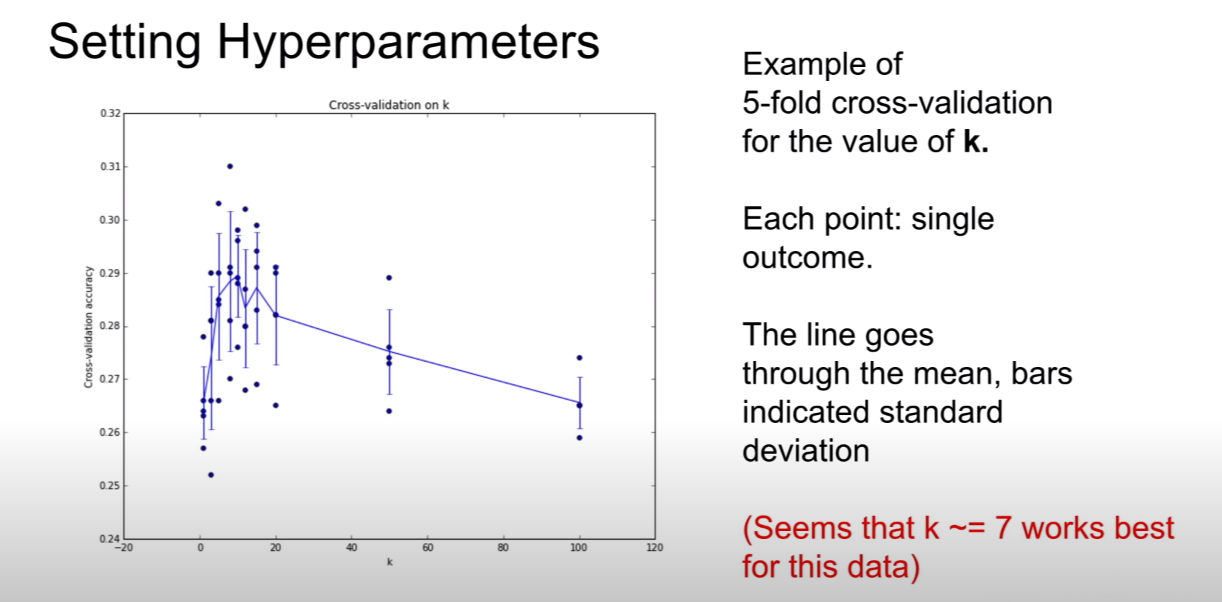
- another common method is cross validation.



- In above picture, there is 5 folds. and by taking each fold as a validation set, we can check values of hyperparemeter.

- but cross validation is usually used on small data set but not deep learning. but it’s computationally expensive.

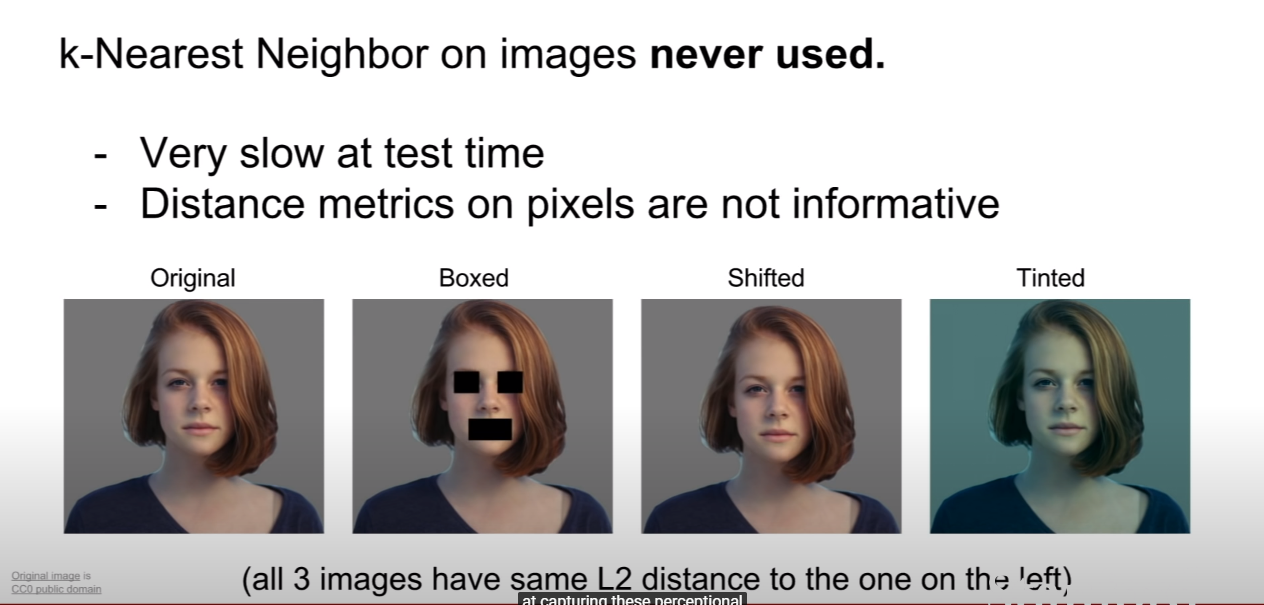
- a student’s Question : is test data always fitted on training data?

> answer : when we split a given data randomly, we can ususally get train and test data having same probability distribution. but if we took earlier period of data as a train, and later period of data as a test. the test data couldn’t represent whole data set, and train data as well.  


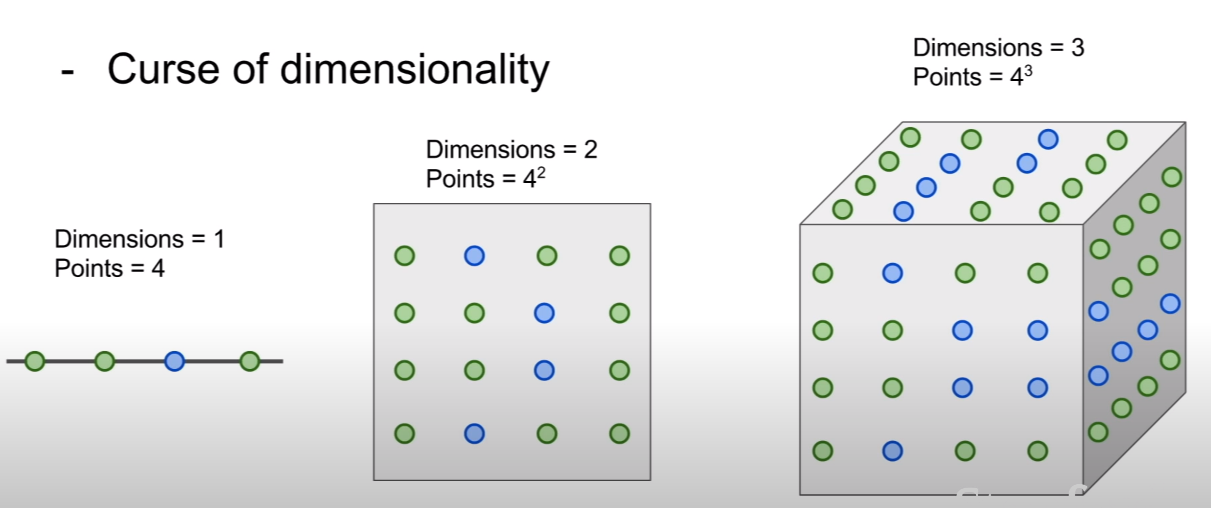
- but in practice, k nearest neighbor is never used.

> First, it’s terribly slow at test time.

> Second, L1 and L2 distance is not a very good way to measure distance between images. since this vectorial distance functions do not correspond very well to perceptual similarity between images. that is, it’s different with the way human’s eyes work.

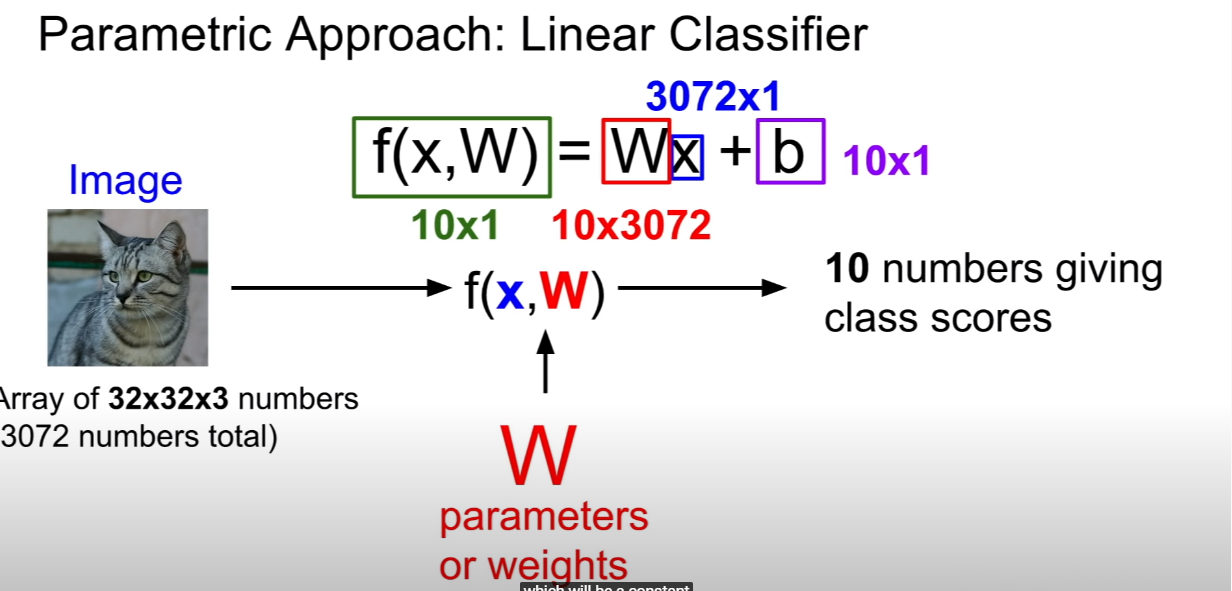


- All the picture above has same distance, so distance matrice on pixels doesn’t have information.



- and like above, depending on dimensionality, needed points to train grows exponentially.

**[Linear Classification]**



- parametric’s meaning is if we change a value of a functions, then it make change to another value.

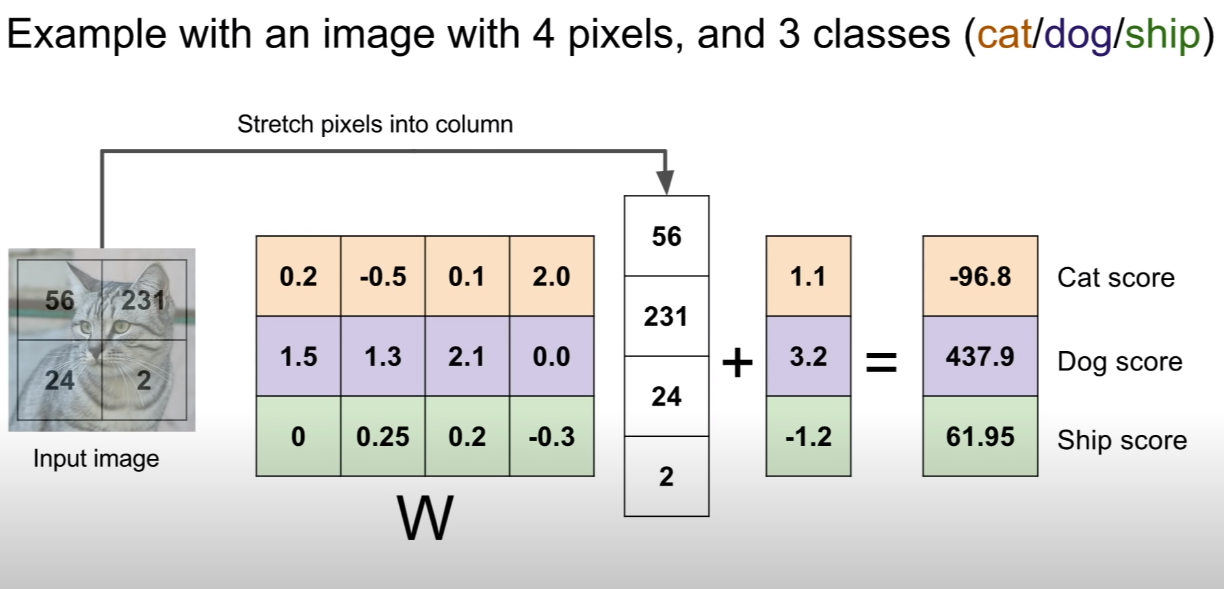
- in Parametric approach, if we get W by training a model using training set, we don’t need training set anymore to predict test set.

- in above example, we have 10 label, so we should get 10 predictions, so function f is Wx +b.

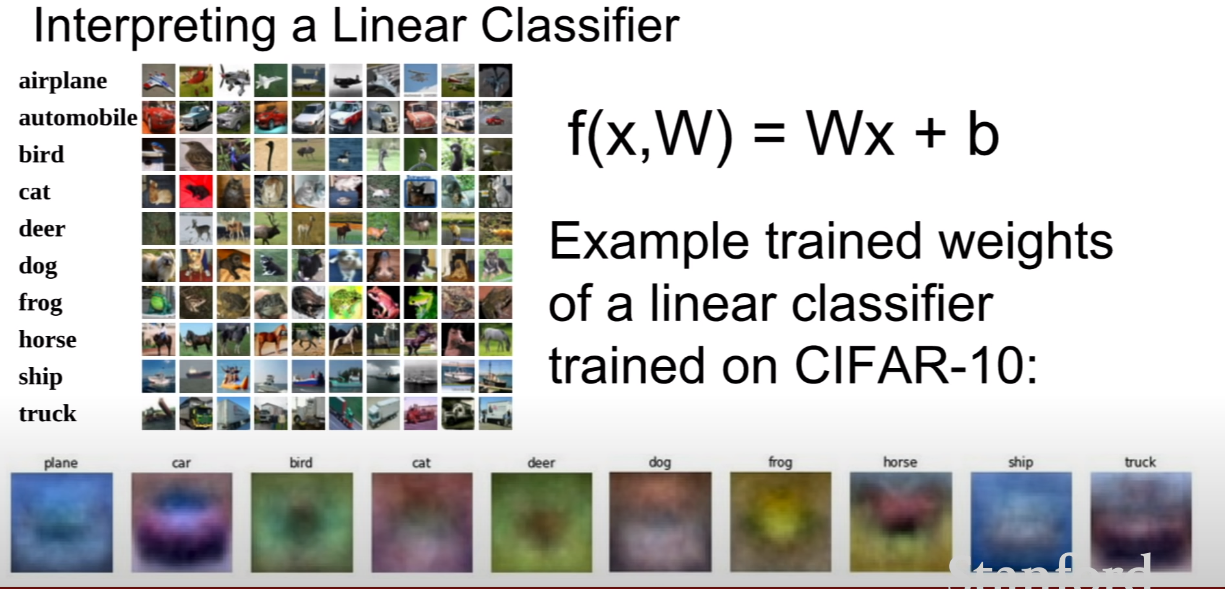
> W is weight, x is input, b is bias. 3072 meaning is that one one pixel has rgb information size of 255 bit respectively so it is 3 byte. and there is 32 x 32 pixels in whole image. so it will be 3072.

> and 3072 x 1 matrix has to be 10 x1 vector since there are 10 label. so we take 10 x 3072 as W.

> bias just gives us a preference of for some classes onver another. meaning is, if we have unbalanced data such as there is lots of cats, or dog, bias works to make unbalanced result.

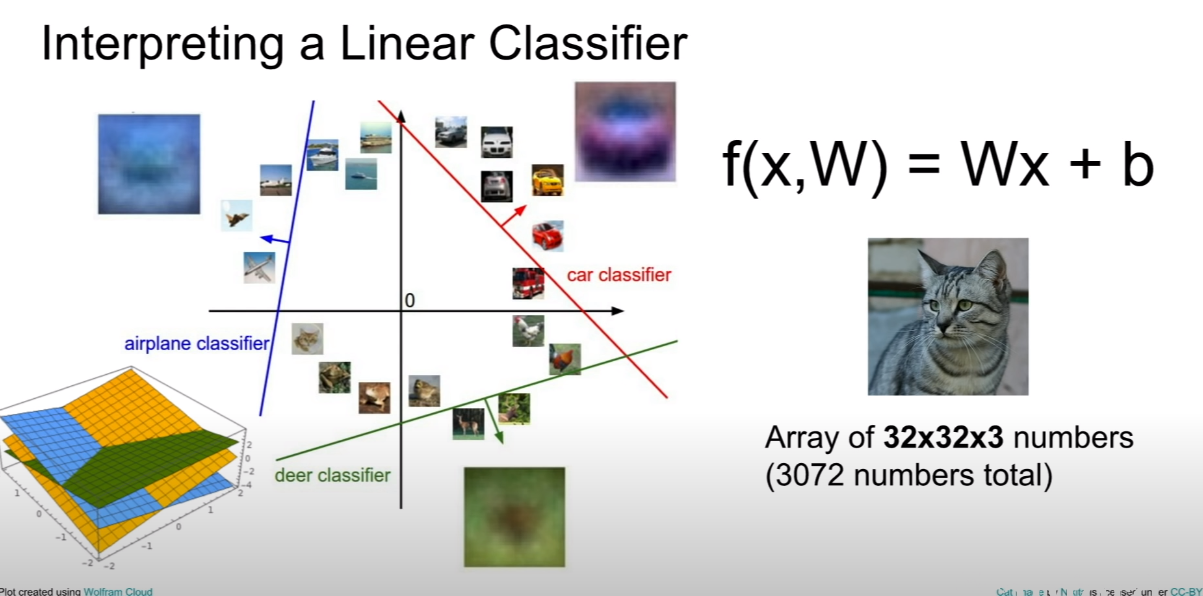


- above picture respresent how linear classification works.

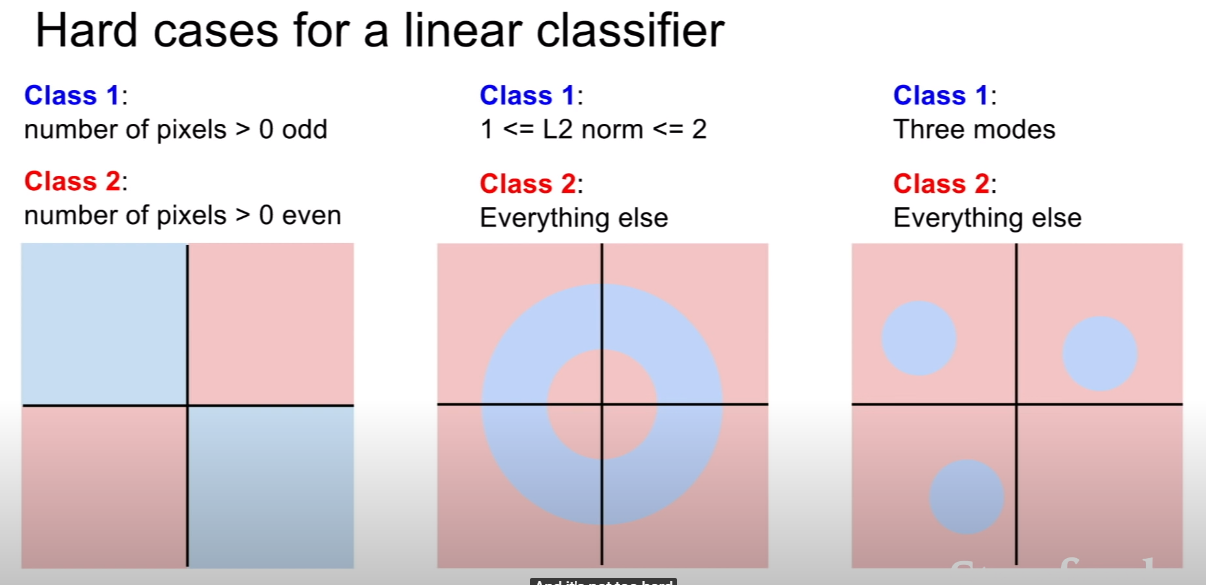


- the bottom pictures show that filter, I mean W trained by CIFAR-10.

- but linear classifier is allowed just one template (the bottom pictures X) per category.



- as you can see, linear classifier is to draw a line to classify a given label to rest labels.



- linear classifier’s problem is like above, there is no way to split red and blue by drawing only one line.