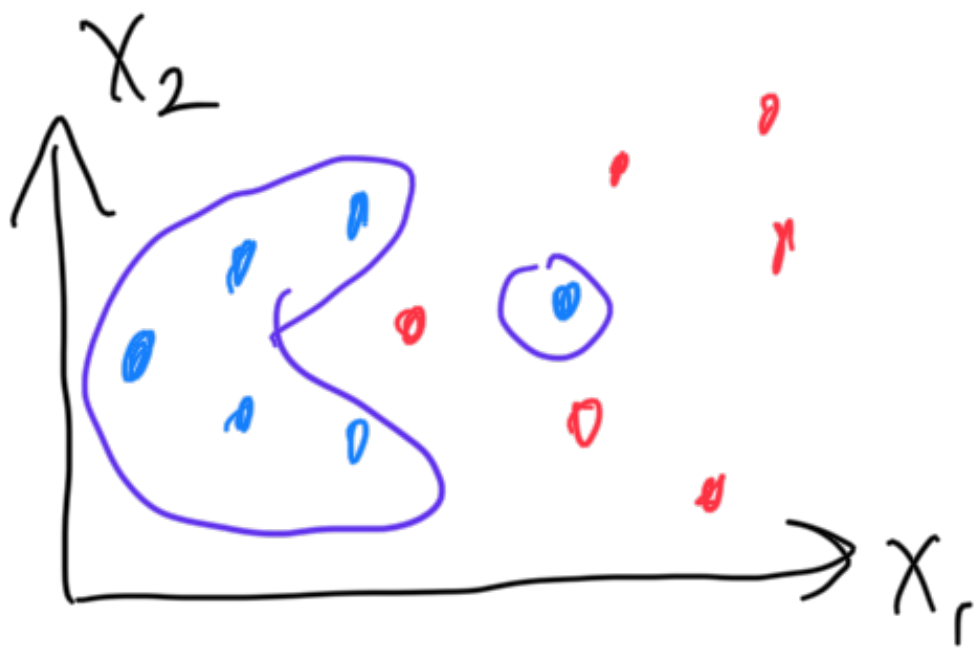
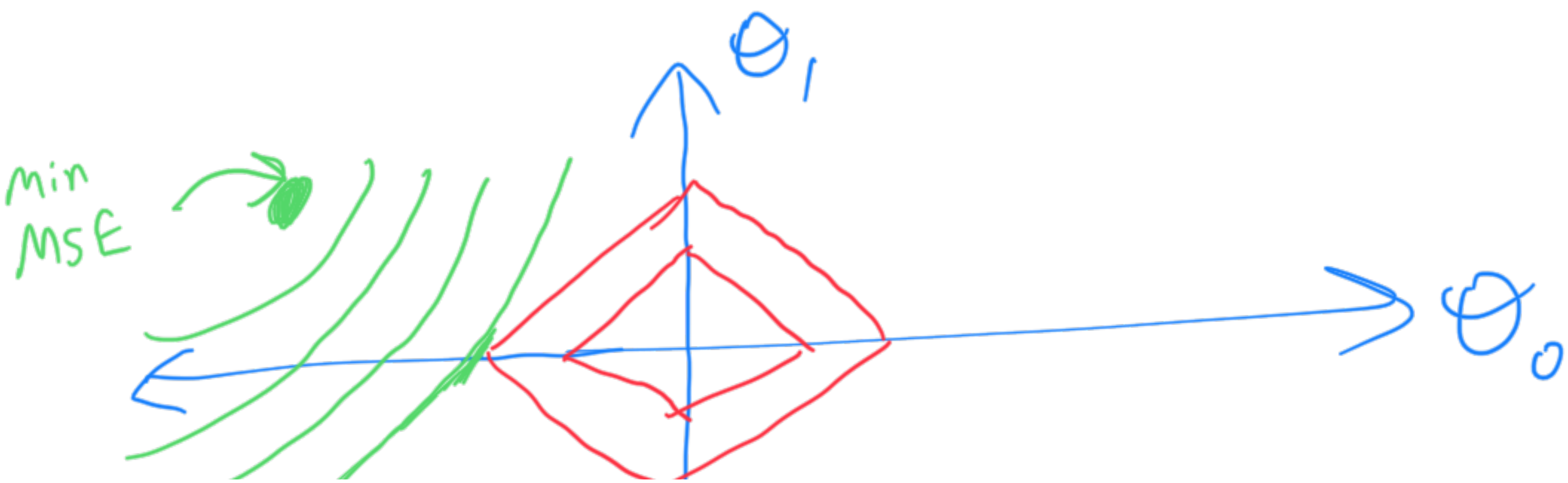
Day 16: Feature Selection and Engineering

Feature Selection: Remove features which are not helpful for prediction



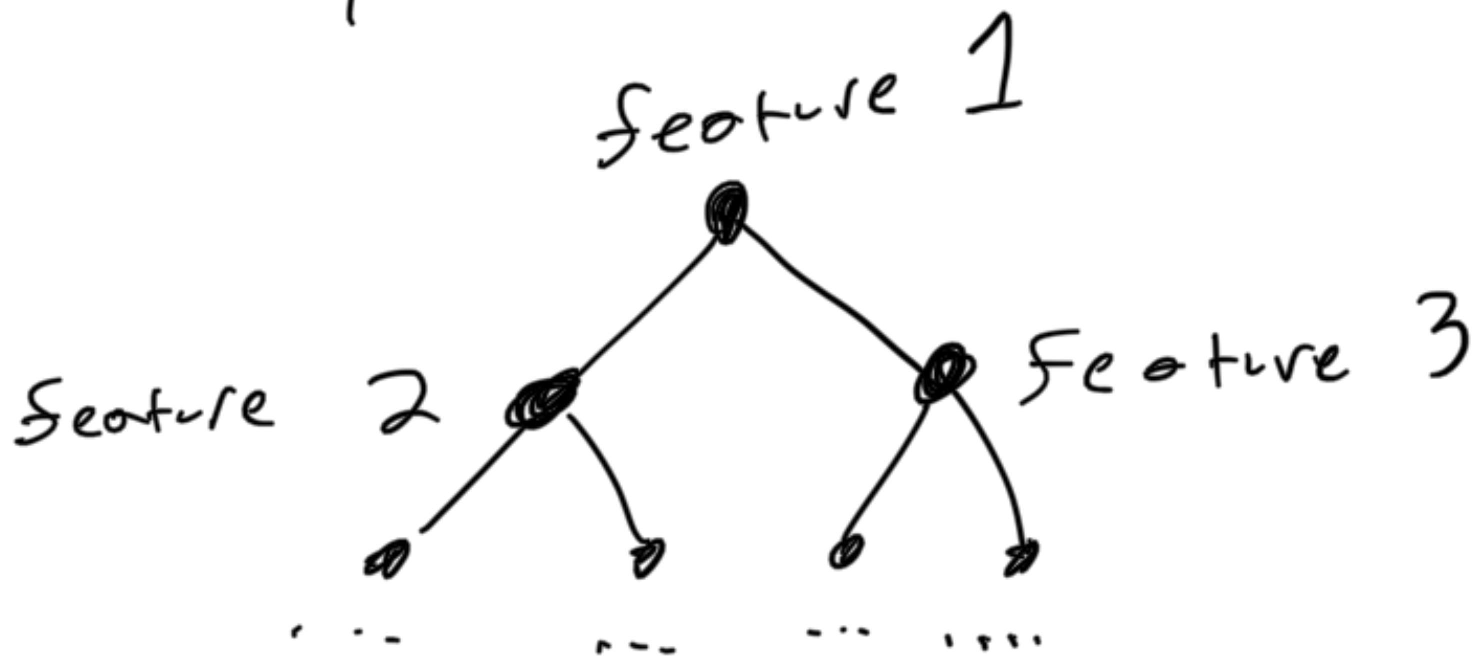
Method 1: L1 Regularization

Penalizing weights by adding $\lambda \sum_{i=1}^{n} |w_i|$ to the loss function



Method 2: Decision Trees

Decision Trees perform feature selection for us:



Can use the top-N nodes of the tree as the top-N features

Method 3: Linear/Logistic Regression

$$\hat{y} = \sigma\left(12x_1 - 2x_2 - 15x_3 + b\right)$$

$$y = \text{sigmoid}(12x_1 - 2x_2 - 15x_3 + b)$$

"Cancer"

X_1 contributes strongly towards "Cancer"

X_2 technically contributes to "no cancer", but the coefficient is small, so remove this feature

X_3 contributes strongly towards "no cancer"

Method 4: Mutual Information

Rank features by information gain:

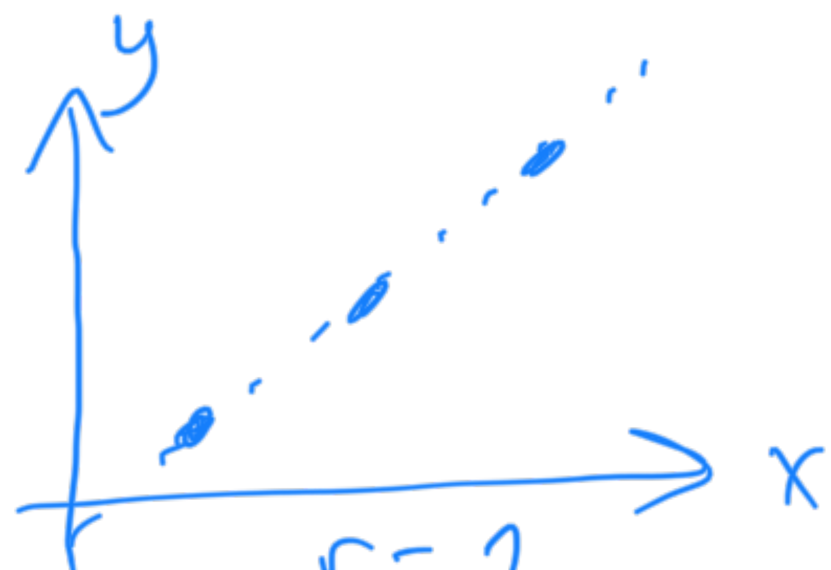
$$IG(x, y) = \underbrace{H(y)}_{\text{uncertainty of } y} - \underbrace{H(y \mid x)}_{\text{uncertainty of } y \text{ if you know } x}$$

How does knowing x affect y?

Method 5: Statistical correlation

Sort features by absolute value of correlation coefficient between feature and outcome variable:

$r = -1$



$r = 1$

$r = 0$

$0 < r < 1$

$-1 < r < 0$

Method 6: Recursive Feature Elimination (RFE)

Key insight: features often have complex interactions / interdependencies

Examples:

- $18 < BMI < 25 \rightarrow$ associated with being healthy
- $24 < BMI < 32 \rightarrow$ healthy if % body fat < 2%

score on midterm, score on final

Steps to RFE:

While # features > desired # features:

- Train model using current set

of features

- Rank features using any feature ranking method (e.g, th ones above)
- Remove the least important feature

Common Feature Representations for NLP

① Bag of Words (BOW)

- Create a vector with length = # of words to consider
- use one vector position per word
- The value in each position is the # occurences of the word

"Great food, great music, and great vibes"

$$\longrightarrow \begin{array}{ccccccc} [\,3 & 0 & 0 & 1 & 1 & 0 & \dots\,] \\ \text{great} & \text{Hawaii} & \text{orange} & \text{food} & \text{vibes} & \text{art} & \dots \end{array}$$

- Vector is of length N, where N is a hyperparameter and only the top-N frequenty occuring words are included in the vector

② Term Frequency - Inverse Document Frequency (tf-idf)

Associate each word in the document with # representing how relevent that word is

$$\text{tf-idf} = \underbrace{tf}_{\text{term frequency}} \cdot \underbrace{idf}_{\text{inverse document frequency}}$$

Term Frequency $tf(t, d)$ = relative frequency of term t in document d = $\dfrac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$

every other term

"element of"

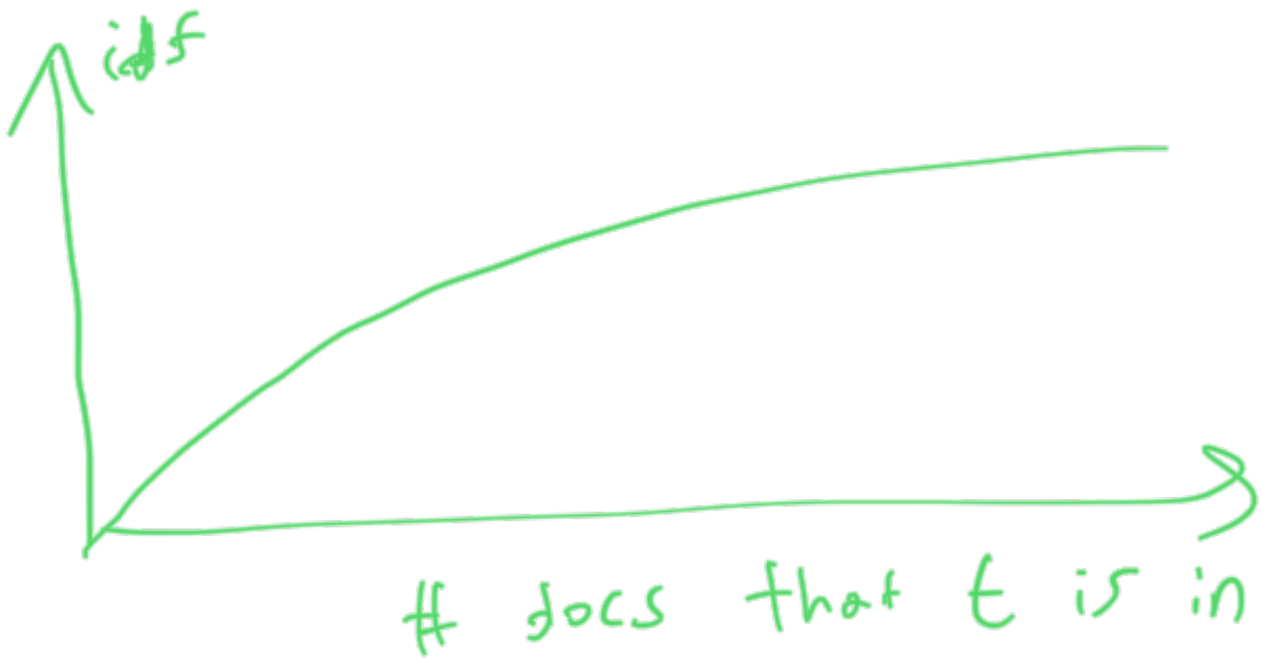
Inverse Document Frequency $idf(t, D)$ = measure of how much info the word provides within all documents D = $\log\left(\dfrac{\text{number of documents}}{\text{number of docs where word } t \text{ appears}}\right)$

Notice that:

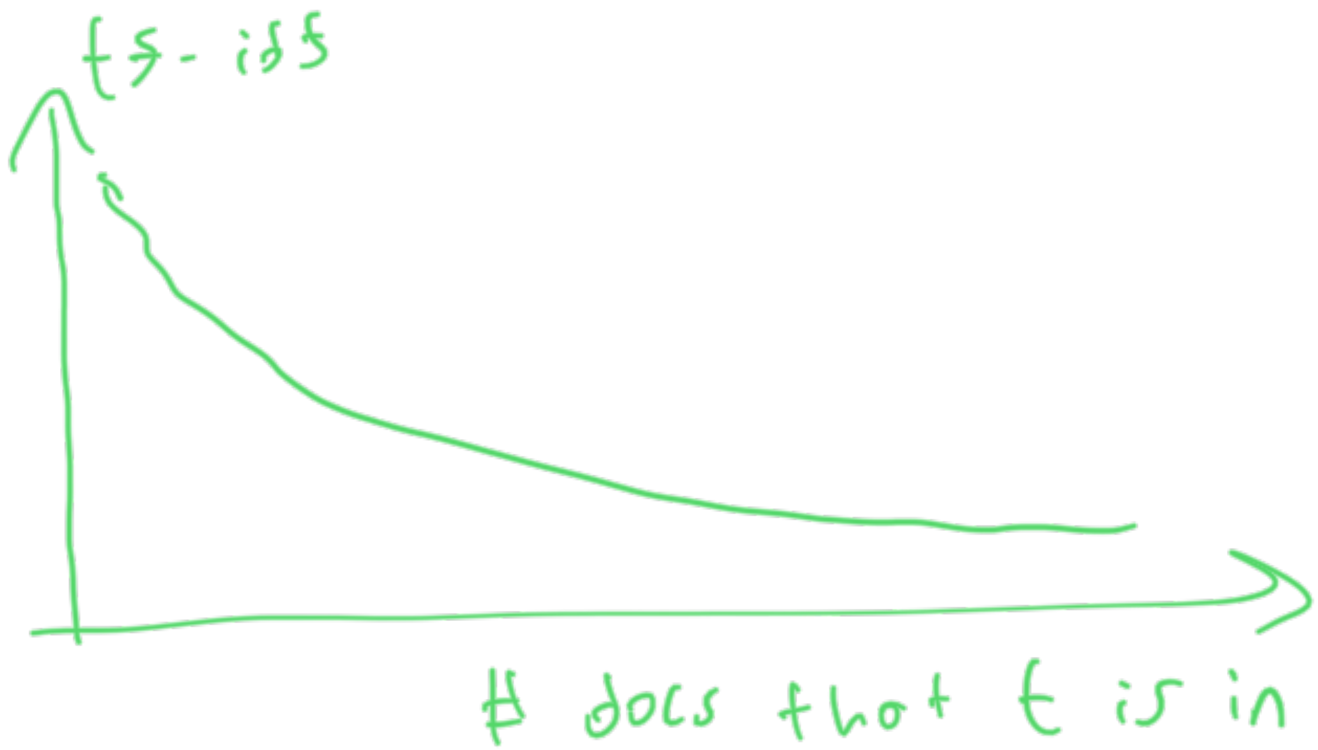
* idf is 0 when word appears in every document
* idf increases logarithmically as # docs where

increases

word t appears in



Therefore:



Then, can create a BOW-style vector using the tf-idf representation of each word:

Example:

Document	Text	# words
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8

Word	tf (for A)	tf (for B)	idf	tf·idf
Jupiter	1/5	0	$\ln(2/1) = 0.69$	0.138
is	1/5	1/8	0	0
the	1/5	2/8	0	0
largest	1/5	0	0.69	0.138
planet	1/5	1/8	0	0.138
Mars	0	1/8	0.69	...
fourth	0	1/8	0.69	
from	0	1/8	0.69	
sun	0	1/8	0.69	

Doc A representation:

$$\begin{array}{ccccccccc} [\,0.138 & 0 & 0 & 0.138 & 0.138 & 0 & 0 & 0 & 0\,] \\ \text{Jupiter} & \text{is} & \text{the} & \text{largest} & \text{planet} & \text{Mars} & \text{fourth} & \text{from} & \text{sun} \end{array}$$

Documents with similar, relevant words will have similar feature vectors

(BoW is the "tf" part of tf-idf)

Common Feature Representations for Computer Vision

(1) Just use the original image

$$\left[\underbrace{\begin{bmatrix} 255 & 255 & 255 \\ 0 & 255 & 0 \\ 0 & 255 & 0 \end{bmatrix}}_{\text{Red Channel}}, \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Green Channel}}, \underbrace{\begin{bmatrix} 255 & 255 & 255 \\ 0 & 255 & 0 \\ 0 & 255 & 0 \end{bmatrix}}_{\text{Blue Channel}} \right]$$

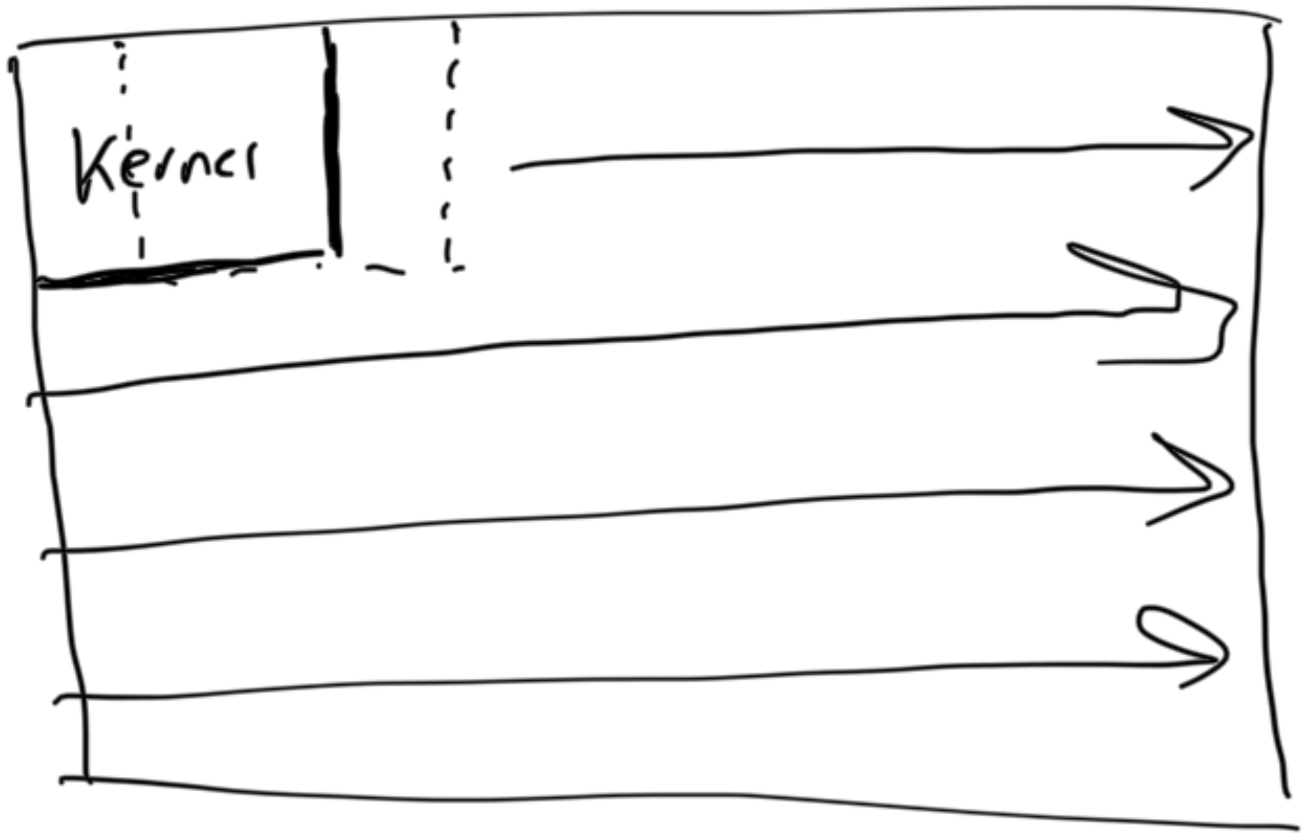
=

(2) Apply an <u>image kernel</u> throughout the image ("<u>convolution</u>")

A <u>Kernel</u> (in the context of CV) is:

- a small matrix
- applied via a <u>sliding window</u>

- take dot product between the original image portion and Kernel to get the new pixel value for that position



original image

Different Kernels filter images in different ways:

- Blur filter
- Sharpen filter
- Edge filter

setosa.io/ev/image-kernels

Example

Original image:

255	255	255
0	255	0
0	255	0

Kernel:

$$\begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$$

First:

$$\begin{bmatrix} 255 & 255 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \end{bmatrix} = 255 \cdot 2 + 255 \cdot 0$$

$$\begin{bmatrix} \cdots \\ 0 & 255 \end{bmatrix} \cdot \begin{bmatrix} \cdots \\ 0 & -1 \end{bmatrix} = \cdots + 0\cdot 0 + \cdots$$

$$= 265$$

So, the new image so far is:

265	

Do same thing for other positions:

$$\begin{bmatrix} 255 & 255 \\ 255 & 0 \end{bmatrix} \cdot \text{Kernel} = 510$$

$$\begin{bmatrix} 0 & 245 \\ 0 & 255 \end{bmatrix} \cdot \text{Kernel} = -255$$

$$\begin{bmatrix} 245 & 0 \\ 255 & 0 \end{bmatrix} \cdot \text{Kernel} = 490$$

... of the convolution is:

So, the result of the

265	510
-255	490