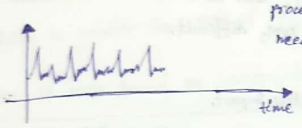
Featurization & Feature Engineering

* **Featurization** → converting any sort of data to numerical vectors.
* **feature engineering** → modify the numerical vector so it works well with a model. (hacks)

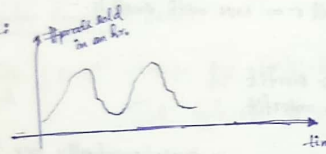
- Ex → BoW, tf-idf, avg W2V etc are various featurizations of text data.

(i) examples of time series data

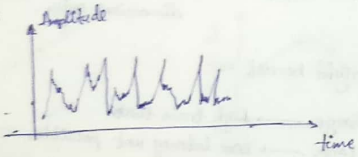
① heart-rate



② e-commerce:



③ speech / Audio



In the end, ML algos can only process numbers. So, everything needs to be converted to that before processing.

Other examples would be stock market, etc.

(ii) **Image data** → face-detection, face recognition, MRI-scans etc.

video → Image data + time series data.

(iii) **Database tables**

- converted to numerical features before supplying to ML.

T_1

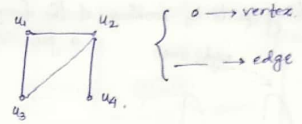
cust ID	DOC

T_2

cust ID	pr ID	Time

P_ID	P_Cu	Discount

(iv) **Graph data** → recommend a friend on FB.



There are tons of types of data.

Researches have spent decades for getting proper featurization.

* 30+ yrs of research for image featurization only.
* featurization for x-ray images mayn't work well with detecting faces.

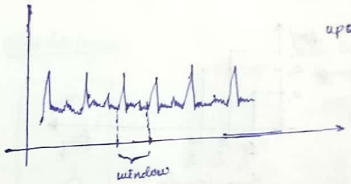
* can't read up all the featurizations to all kinds of data.

So, lookup when required.

Featurization of time series data

① **Sliding window**

window-width → domain & application specific.



We can do various types of calculations upon all the values belonging to the window like

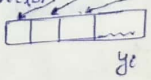
① mean, std-dev
② median, quantiles
③ min, max
④ max − min
⑤ max / min
⑥ # local minima or maxima
⑦ # mean crossing
⑧ # zero crossing.

& so on.

* Which all out of these is suitable is decided by domain expert.

Steps

① decide upon a window width.
② select features to constitute feature vector



Read up research literatures to get what to use.

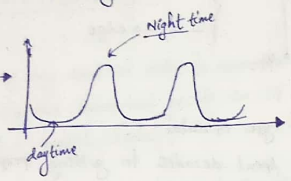
Deep-learning (2012) – automatically learns the best featurization given right amt of data.

② Fourier Decomposition / Transform →

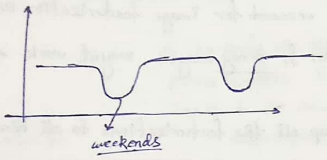
Wherever we have some repeating pattern, looking at the frequency is a good idea.

ex → e-commerce

daily pattern →

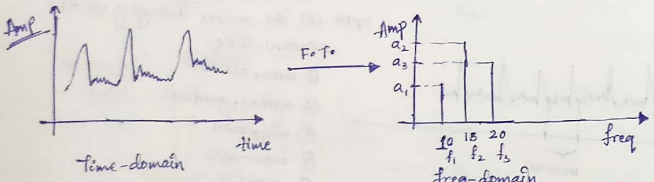


weekly pattern →



Similarly, annual pattern → dip during vacations.

Composite wave (with repeating pattern) —fourier decompose→ sum of multiple sine waves with different frequencies & amplitudes.



feature vector → somewhat like

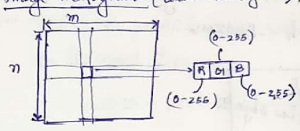
f_1	f_2	f_3	a_1	a_2	a_3

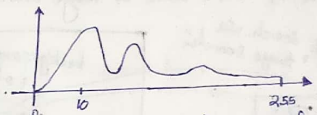
— fourier featurizat^n of time series data.

comes very handy if we have repeating patterns.

Featurization of Image data

② Image histograms (colour histogram). * collect red values fm each of the (m×m) pixel & plot the histogram





This gives the frequency of occurence of a particular intensity of red color in the entire picture.

The histogram can then be converted to a vector.

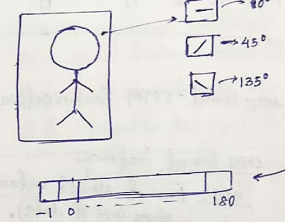
similarly we'll have vectors for Green & Blue colors.
We can concatenate them to get the resultant feature vector.

* Such featurizat^n can be used for tasks like:- check whether there is sky or not in given image.

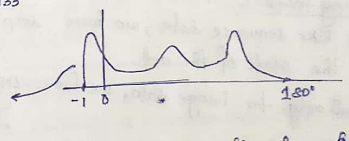
→ If there is sky, there'll be high percentage of high intensity blue color.

* But colour histograms can't recognize shapes.

edge histogram



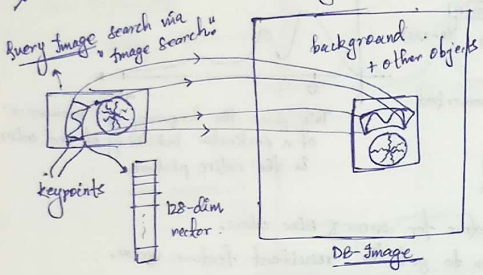
So, image is broken down into pixels as in above and edge angle for each pixel assigned.
No edge = -1.



These two histogram featurizations are very rudimentary.
For something like face detection, we have more complicated featurization - "haar" features.

(b) Scale Invariant feature transform (SIFT)

* extremely useful for detecting objects in an image.
* variant of SHIFT used in image search - Amazon.



→ It detects keypoints. — mostly ~~edges~~ corners.
for every keypoint, creates a 128-dim. vector.
These keypoints help in detecting an object in the image.

SIFT → scale invariant → Even if query image is larger than DB-Image or vice-versa, it'll ~~a~~ still detect.

Similarly, Rotation invariant ✓

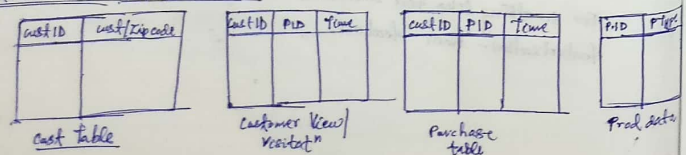
Opencv → can be used to get SIFT features of a given image.

Deep Learning features

Just like sequence data, we have deep learnt LSTM featurization as the state of the art.

Similarly, for image data, we have CNN learnt features.
└ best (learnt almost automatically given lots of data).

Relational data & featurization



Can featurize DB-data by extracting using pandas, python, SQL. & domain knowledge.

Ex → Task is to predict if a customer would purchase a product in the next 7-days.

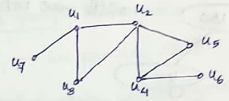
cust ID + prod. ID ⟶ 1/0 — doesn't buy.
buys.

↳ We have to use domain knowledge to convert the given (cust ID + prod ID) to a feature vector for the given task.

Some relevant features —

① f_1 → #times the cust ID visited the page of prod ID in last 24hrs.
② f_2 → #times the cust ID visited any prod of same type as prod ID in last 24hrs.
③ f_3 → income level.
④ f_4 → zipcode. - signifies urban/rural area.

Graph-data & featurization



social graph.
u_i :- vertex (users).
(u_i, u_j) :- edge (friendship).

Task :- recommend new friends to a user u_i, say u_4.
Some useful features would be
① f_1 — #mutual friends b/w u_4 & u_i $\forall i \neq 4$.
② f_2 — #path b/w u_4 & u_i $\forall i \neq 4$.

Feature Engineering : Indicator variable.

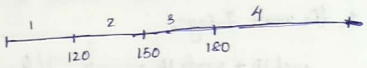
Ex → height can be used → directly as a real valued feature.
→ converted to binary indicator variable feature before using.
ex → $h > 150 \longrightarrow 1$
$h \leq 150 \longrightarrow 0$
why we may need to do so & decide threshold are problem specific.
it may refer both (1) & female or anything.

Ex2 → categorical feature → country
if (country = India) or (country = USA)
return 1
else
return 0.

Feature Binning (Bucketing)

→ Logical extension to ~~feature~~ indicator variable.

Ex → if $h < 120$ cm
return 1
else if $h < 150$
return 2
else if $h < 180$
return 3
else
return 4



Interview Ques:→ predict gender given – weight(w), height(h), hair length(hL), eye color (ec).

D_{train}

h	w	hL	ec	g

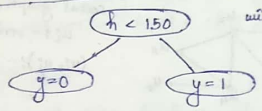
challenge:→ perform data driven binning on feature 'height' using D_{train}

Soln:→ take h & gender(g) & make a new table.

h	g

Now train a decision tree.
simple
DT

DT finds the threshold with max inf. gains.



* Many kaggle competitions, this trick comes handy.

Interaction Variables

Ex → h, w, hL, ec → gender
x_i y_i

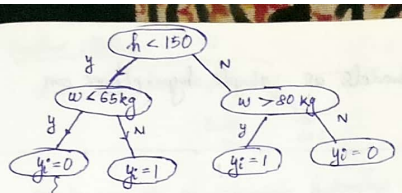
say fm domain knowledge we know if $(h < 150)$ & $(w < 60)$ → female- high probability.

Then, we can create a new 5th feature by 2-way ~~logical~~ interaction.

$f_5 =$ $(h < 150)$ AND $(w < 60)$

similarly, we can also have $f_6 = h * w$ — math 2-way interaction feature

Again, DT can be used to create nice interaction features.



$(h < 150\text{cm})$ AND $(w < 65\text{kg})$ Similarly for each leaf node we can have an interaction variable.

* find those interaction features are guaranteed to be helpful because DT was constructed to predict gender using information Gain.

Mathematical transforms

what is the best transform?
→ problem specific.

x – single feature.
$\log x$, e^x
$\text{sqrt}(x)$, $\sqrt[3]{x}$
x^2, x^3, x^4 ... (polynomial)
$\sin(x), \cos(x), \tan(x)$

Ex → if 'x' has power law distributions then taking log(x) transform as it'll have a gaussian distributions.

Model Specific Featurization

Ex → f_1 → has powerlaw distⁿ
We want to apply Logistic Regression → same as Gaussian Naive Bayes (assumes features are gaussian distributed).
Then its sensible to use $\log(f_1)$ transformation.

Ex2 → f_1, f_2, f_3, $y \in \mathbb{R}$.
fm domain knowledge we have idea that
$y \approx f_1 - f_2 + 2f_3$ → linear combination of f's.
then linear models are powerful if used as compared to other.
linear regression with
$w_1 = 1$, $w_2 = -1$, $w_3 = 2$ → fits decently.

Ex3 fm domain knowledge, say we know
y depends on → interaction of f_1 & f_2.
Then better use DT/RF/GBDT

Esp BOW → high dimensions.
Then better to use linear models as simple hyperplanes can separate data.

Feature Orthogonality

* The more different/orthogonal the features are, the better the model would be.
* So, wherever adding a new feature, keep in mind that the new feature should be correlated to the o/p 'y' & very less correlated to other features.

Ex → f_1, f_2, f_3 — Model → $\hat{y_i}$ & we also have y_i — actual labels.

How to design a new feature f_4 s.t.
f_4 — correlat" → y_i & f_4 — less corel" → f_1, f_2 & f_3

Sol" → $e_i = y_i - \hat{y_i}$ Try to design f_4 correlated to e_i
So, f_4 related to $\hat{y_i}$ but less correlated to other features.
Similar to boosting ←
* Remember to avoid overfitting.

Domain Specific featurizat"

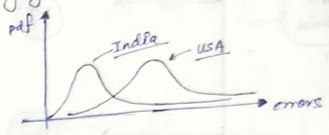
Ex → Predicting heart attack using ECG data.
* Not reading these at all is a big blunder. ← {Always important to research and study existing featurizations by doctors/specialists.
Can create new feature out of these existing using ML hacks from experience.

blog.kaggle.com/winner-solution — details of the feature engineering they have done.

Feature slicing

Ex → $h, w, hL, kc, country$ (x_i) → gender, y_i
country: India 80%, USA 20%

Due to majority of datapoints belonging to India, model performs
① better on Indian test data.
② worse on USA test data.



strategy → D → $D_{India} \rightarrow M_1$, $D_{USA} \rightarrow M_2$

This idea of slicing the dataset and training separate models helps if
ⓐ both category have different behaviour.
ⓑ sufficient data points for each category.

Real world example → Mobile & Desktop users in Amazon have different types of traffics.
So, feature slicing done.

Ensemble Models

Multiple models used together.
$\{M_1, M_2, M_3, M_4 \dots M_k\}$ — combine → More powerful Model.

4 types of ensembles

① Bagging (Bootstrapped Aggregation).
② Boosting
③ stacking
④ cascading

Mostly used in Kaggle as well as Real world scenarios.

Key Aspect → More different the models are, the better & powerful it becomes.