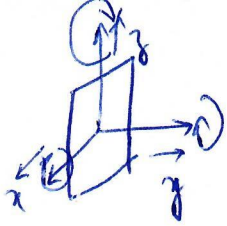
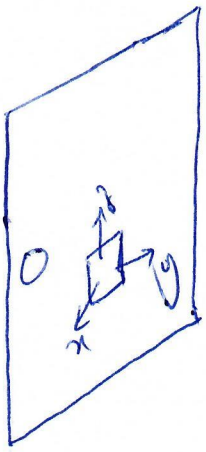
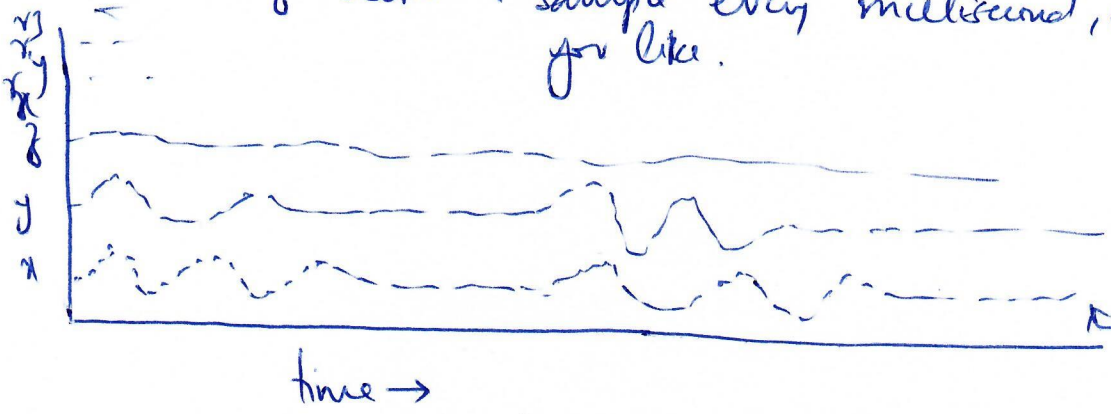




Door open - close
↳ x & y acc values will change
↳ Strictly speaking z won't but due to small inaccuracies in door installation, hinge efficiency etc, it will show some variations

6 - streams of data : Sample every millisecond, or whatever interval you like.



subsequent explanation is only for one stream.

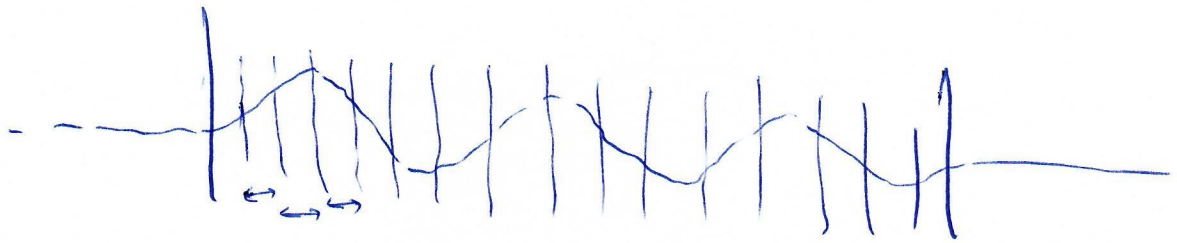
START END DETECTION

Need to detect start & end times

thresholding - Measure your sensor values for each stream for a while

Calculate μ, σ. When sensor value for that stream exceeds $\mu \pm 3\sigma$ for let say 5 ms / 10 ms (you decide the duration) consider the first point when it exceeded for any of the stream as start point. Detect end point in same way.

FEATURE EXTRACTION



divide into x-parts & extract feature for each part
Average, Median, raw value etc. your choice

If you divide into x-parts, you will get x feature values.

Keep x fixed across samples of different lengths.
& don't divide entire ~~door~~ signals using ~~time~~ a specific window width Rather just divide ~~into~~ into equal # of parts

COLLECT TRAINING SAMPLES

for Both door open & door close.

from each sample extract the x values you get a table

1	2	3	..	—	x	Label x+1
						0
						0
						C
						C
						0
						C

↓ ↓ ↓

↳ Normalize each column. & remember the normalization factor this helps in reducing implicit bias towards feature with larger values.

Classifier training

LibSVM: C-SVC RBF kernel.

Cost param : C

kernel param = γ.

need to find appropriate values for this : Grid Search.

Write code that accept y-feature, class labels, γ & C &
perform 10 fold x-validation & returns accuracy.

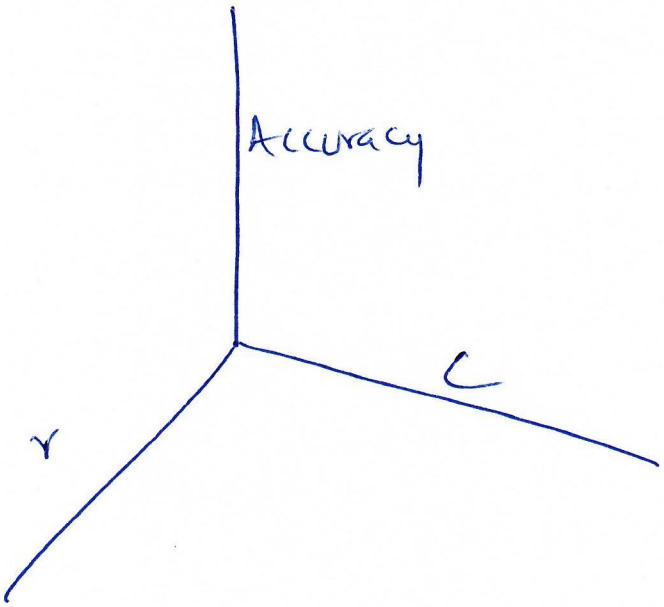
model ← svmtrain (input attr values & class labels) (output)

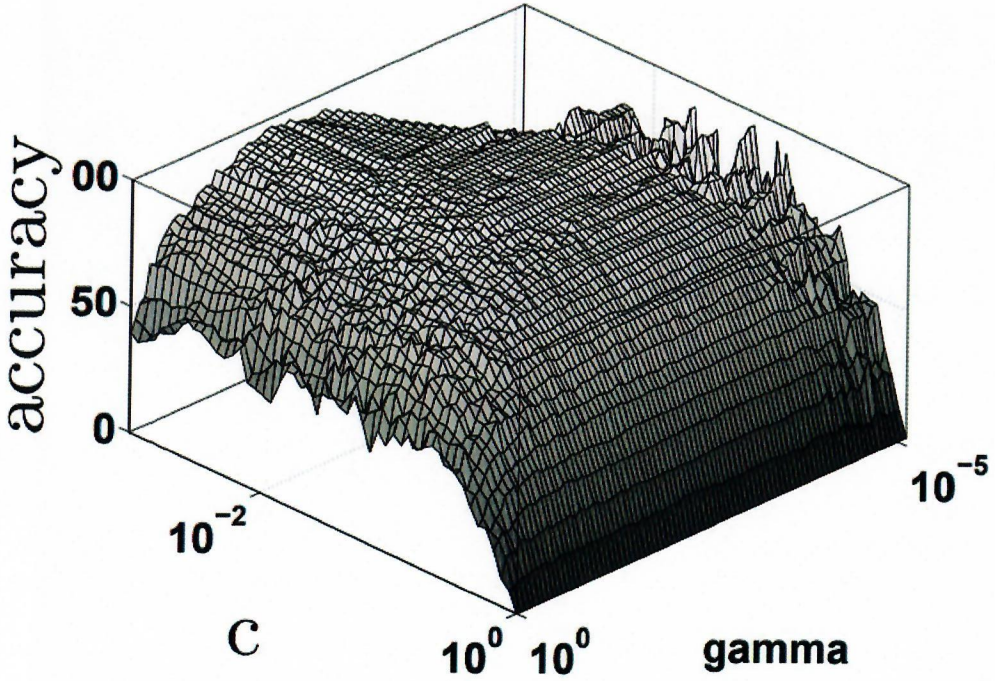
decision ← svmtest (input) model + test sample attrs

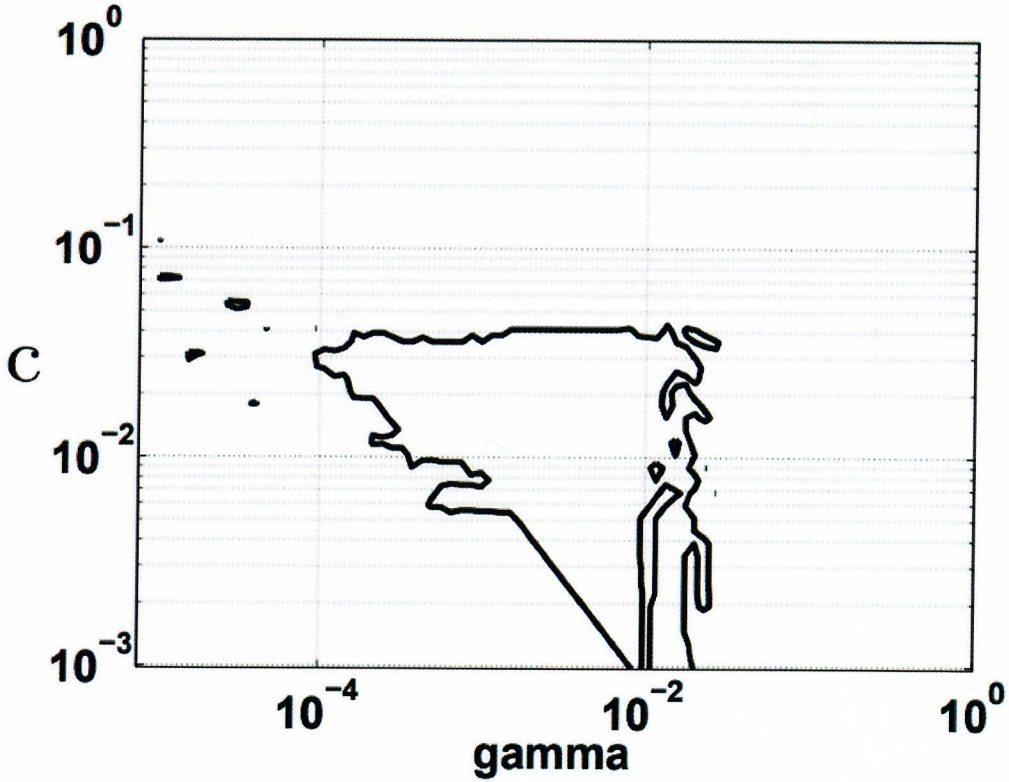
This code is going to be your black box. to select values for C & γ.

Read section of libsvm to see what range of γ & C you should search on.

Get a surface plot - use values of C & γ with highest Acc




accuracy
00
50
0
10^{-2}
C
10^{0}
10^{0}
gamma
10^{-5}


10^{0}
10^{-1}
10^{-2}
10^{-3}
C
10^{-4}
10^{-2}
10^{0}
gamma

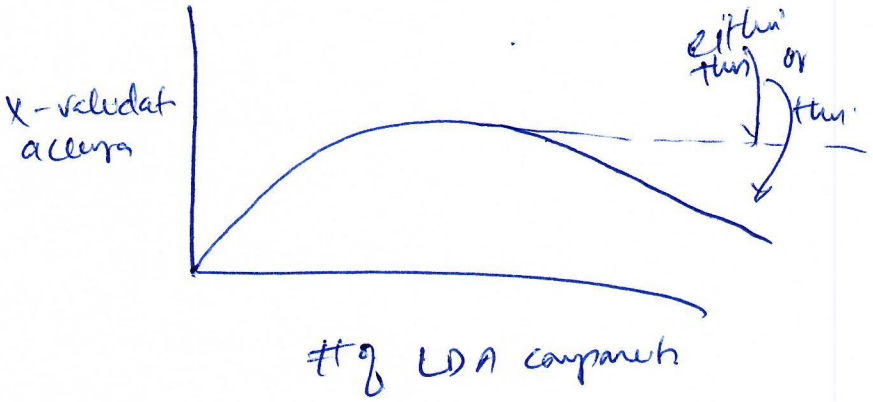
~~Next~~ Handle curse of dim

If you don't get good accuracy, may be that's because you have n intervals & f features per interval i.e. $n * f * 10$ samples needed. for $n=20, f=3$, that makes it 600 samples for door close & 600 for door open.

Apply LDA: This will give you nf new features!

You already have your black box code! use it to determine the # of components with which you get highest accuracy.

It will look something like this



~~You can~~

~~Size of~~ # of intervals x .

```
for x = 1 : xMax
    Extract 39 features from each interval
    Apply PCA/LDA & get x.39 new features
    for w = 1 : x.39
        Select first w components as features
        for gamma = γmin : γinterval : γmax
            for C = Cmin : Cinterval : Cmax
                Perform 10-fold x-validation & record results
                [x, w, γ, C, x-validation accuracy]
            end
        end
    end
end
```

At the end, you will know which combination of x, w, γ & C works best (if there are multiple with very high values then all in parallel & do majority voting)

does not necessarily have to be highest & equal. eg

If you see that 10 combinations give acc > 99.2%, pick all of them.

PARALLELIZE THIS CODE
EQUAL # OF OPEN & CLOSE CLASSES

REMEMBER YOU HAVE 6 STREAMS, DON'T FORGET ABOUT REMAINING 5

- Train all classifiers using all combinations you selected.

- At run time, detect start & end
- for any given combination
- Pick stream & divide into k intervals, extract features,
- Project using same principle components that you used to project the train
- Pick top w values & feed to classifier.
- If you did normalization, normalize the test samples with same normalization factor as used in when normalizing the train data.
- Use the model for this combination of k, w, r & c to come at a decision.
- Majority voting across all decisions i.e. from diff combinations