KALMEN FILTER

Self-driving-car`s equipment

The rotating thing over here is a laser-range finder ? it take distance scans 10 times a second , about million data points.

It will be really important for the Kalman filter class ? it`s major function is to spot other cars so you don`t run into them.

-there is also a camera on the top. There is a stereo camera system over here.

In the rear there are antennas for a GPS – global positioning system. That it is allow to estimate where the car is in the world.

This is a supplemental system to the localization class I just taught you.

This is data that comes from the laser.

This is car parked in the garage right now.

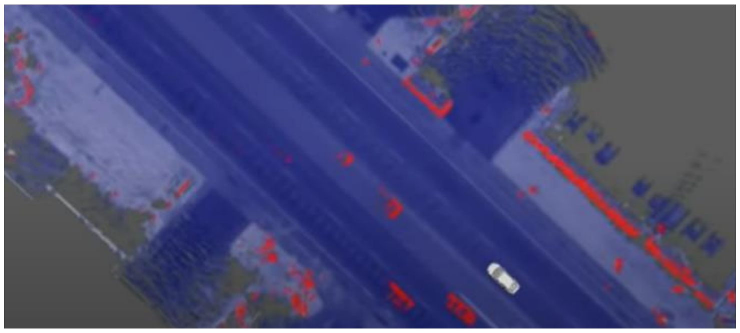
There are all range measurements that tell you how far things are away, and they are essential as the **input Kalman filter** that we are going to learn about

Tracking into:

Self-driving car that uses sensors to sense the environment.

Example

We had a robot that lived in an environment and that could use it sensors to determine where in the environment it is.



Here you can see the Google self-driving car using a road map localization itself but it additions, what is shown here in red are measurement of other vehicles.

KALMEN FILTER

The car uses laser and radars to track other vehicles and today we **are talking about how to find other cars.**

The reason why I would like to find other cars is because we wouldn't want to run into them.

We have to understand how to interpret ***sensor data*** to ***make assessments*, not just where these others cars as in the localization case, but also how fast they are moving.**

So you can drive in a way that **avoids collisions** with them in the future.

That`s important not just for cars, it matters for pedestrians and for bicyclists .

Understand where other cars are an **making predictions** where they are going to move is absolutely essential for **safe driving in the google car project**.

We will learn of tracking the technique i`d like teach you is called a Kalman filter.

This is an insanely popular technique for estimating the safe of a system.

It`s actuality very similar to the probabilistic localizations method.

The primary differences between are that the Kalman filters :**estimate a continuous state.**

Whereas in Monte Carlo localization we are voiced to chop the world into discrete places.

Kalman filter : happens to give us a **unimodal distributions**

Moute Carlo localization : discrete multi_model.

All these technique are applicable to robot localization and tracking other vehicles.

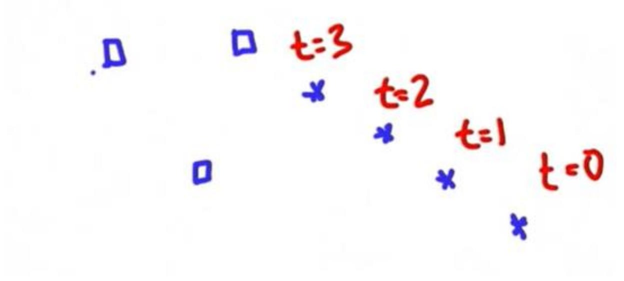
We are going to learn about particle filter.

Which are yet another way to try address the same problem, and indeed they are actually continuous and multimodal.

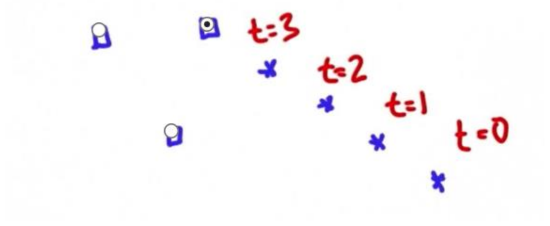
But for the time being let`s look into Kalman filters.

Example/

Consider the car done here. Let`s assume the it see as its measurement an object here, here ,here and t = 0 , t = 1 , t = 2 and t = 3 and when would assume the object would be at t= 4 Check one of those boxes

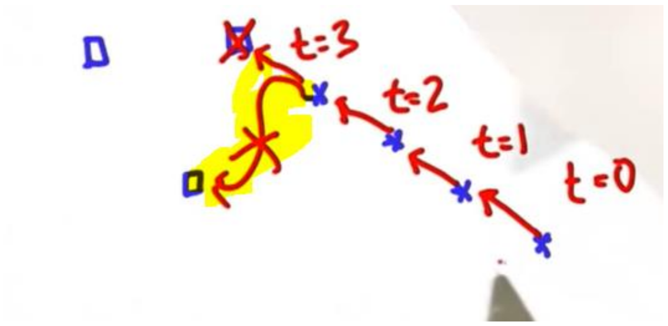


We assume the car will continue to go in a straight line.



From those observation you would say that the velocity points in the direction of this vector.

Assuming no drastic change in velocity,



You expect that the 5th position would be over here.

<u>**The common filter technique observations like these and estimates future locations and velocities based on data like this.**</u>

<u>**You write a software you take a points like those if they are noisy and uncertain and estimate automatically where future locations might be. And at what velocity the object is moving.**</u>

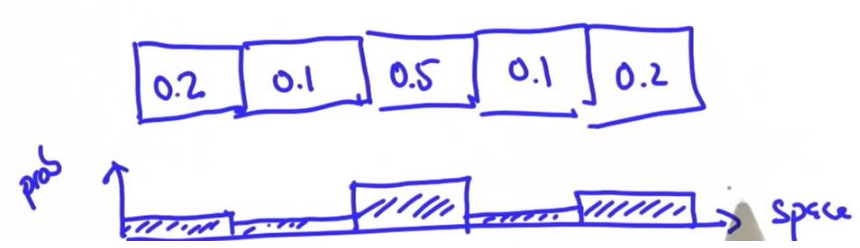
The Google self-driving car uses methods like these to understand where other traffic is based on radar and laser range data.

Gaussian intro.

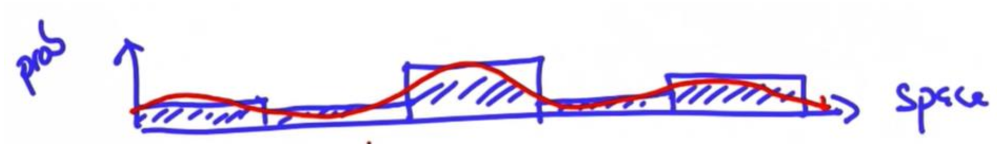
You remember our Markov model where the world was divided into discrete grids. And we assigned to assigned to each grid a certain probability.

Such a representation of probability over spaces is called a histogram.

In that it divides the <u>continuous space into a finite many grid cells</u> and approximates the <u>posterior distribution by a histogram over the original distribution</u>.



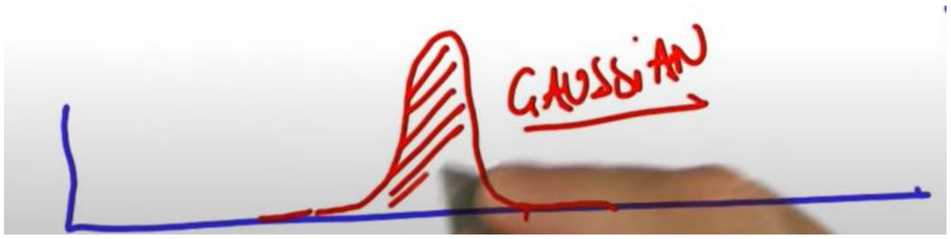
The histogram is a mere approximation for this continuous distribution.



In Kalman filter the distribution is given by what`s called a Gaussian.

A Gaussian is a continuous function over the space of the location.

And the area underneath sums up to 1.



Here`s our Gaussian again.

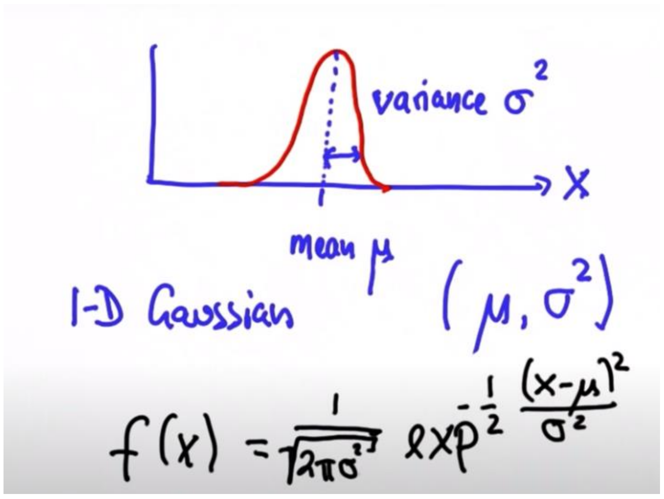
If we call the space x, then the Gaussian is characterized by two parameters

1- The mean often abbreviated with the Geek letter and the width of the Gaussian often called variance and for reasons I don`t want to go into, it`s often written as a quadratic variable sigma square.
2- And Gaussian in 1D, which means the parameter space over here is 1 dimensional, is characterized by mean and variance.
3- Rather than estimating the entire distribution as a histogram, our task in Kalman filters is to maintain mean and variance that is our best estimate of the location of the object we are try to find.
4- The exact formula is an exponential of quadratic function where we take the exponent of the complicated expression over here.
5- The quadratic difference of our query point x relative to the mean divided by variance multiplied by -1/2.

Note

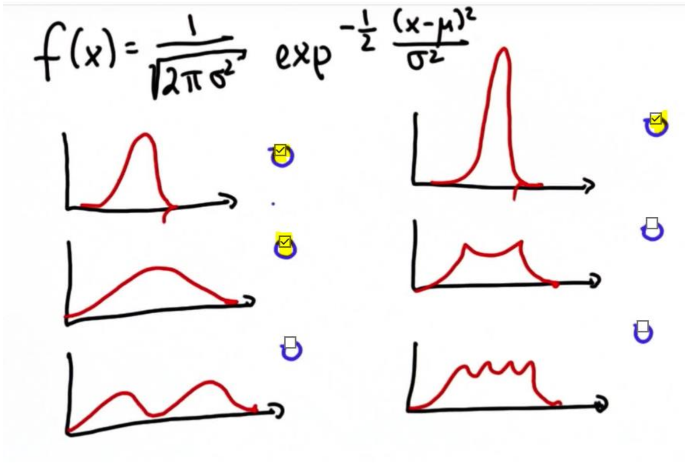
If x equals mean then the numerator becomes 0, and we have x of 0 =, which is 1.

We need to normalize this by a constant I / square root of 2 * mean * variance this constant won`t matters so I ignore it.

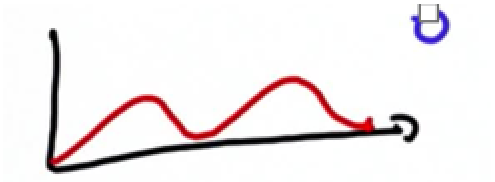

variance σ^2
X
mean μ
1-D Gaussian
(μ, σ^2)
$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$

What matters is we have an exponential of a quadratic function over here which is the Gaussian

`All Gaussians will have a similar shape. Some may have a different mean or variance, but the general shape will remain the same.`


$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$

The shape of Bi-model

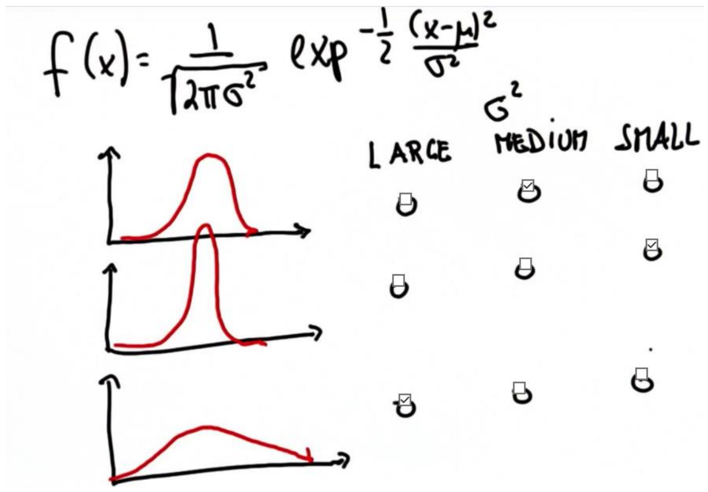


Let`s ask me again about and draw more Gaussian and again excuse my poor drawing skills now i`m asking you about covariance of variance .

1- **Large**
2- **Medium**

3- **Small**

Wide Gaussians have higher variance than narrow ones.


$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}$
σ^2
LARGE
MEDIUM
SMALL

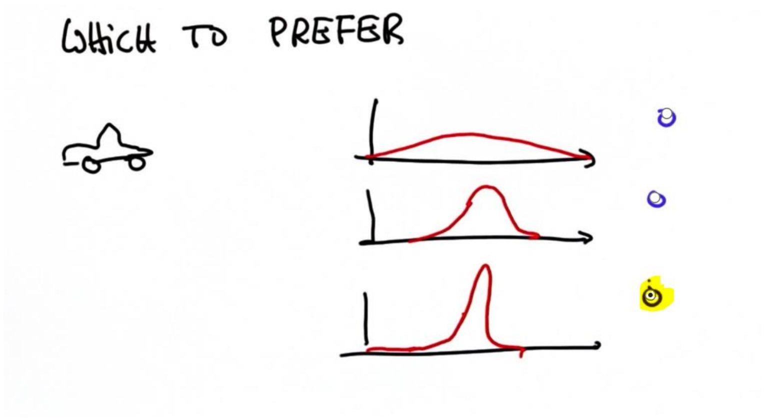
This question asks about the **variance** of each Gaussian (not its covariance).
Correction: In the video under quiz page, it should be "x and u" at 00:15.

1- Put differently, the sigma squared covariance is a measurement of uncertainty
2- The larger sigma—square the more uncertain we are about the actual state.
3- The small variance is a very certain destitution where expected deviation is small.

Which prefer the Gaussian ?

If We track another car with out Google self-driving car, which Gaussian would we prefer

We would definitely prefer a narrow Gaussian, since that means we are confident about our location.

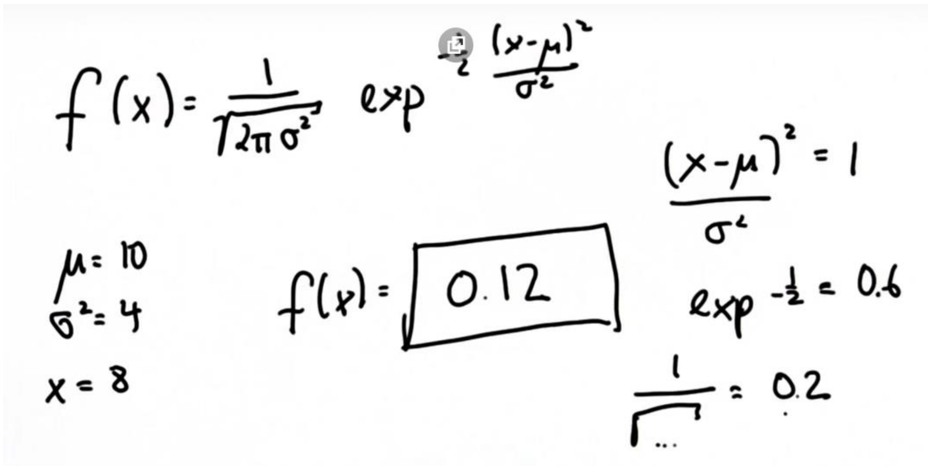


It make a chance of accidentally hitting another car the smallest just by the fact that we know more about the car than in the other distribution.

Gaussian

1- Definition
2- Unimodal   symmetrical
3- Belief in Kalman filter

Let me ask you to calculate the value of f(x)



$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

$$\frac{(x-\mu)^2}{\sigma^2} = 1$$

$\mu = 10$

$\sigma^2 = 4$

$x = 8$

$$f(x) = \boxed{0.12}$$

$$\exp -\tfrac{1}{2} = 0.6$$

$$\frac{1}{\sqrt{\dots}} = 0.2$$

```
#For this problem, you aren't writing any code.

#Instead, please just change the last argument

#in f() to maximize the output.

from math import *

def f(mu, sigma2, x):

    return 1/sqrt(2.*pi*sigma2) * exp(-.5*(x-mu)**2 / sigma2)


print f(10.,4.,8.) #Change the 8. to something else! Change x  = 10 to get maximum number.
```

The answer is assess with the same value as mean.

In which cases this expression over here becomes zero and we get the maximum we get the peak of Gaussian.

Kalman filter represents all destitutions but Gaussians.

Now talk about measurement cycles and motion cycles.

The Kalman filter iterates to different **things**

1. **Measurement updates**
2. **Motion updates.**

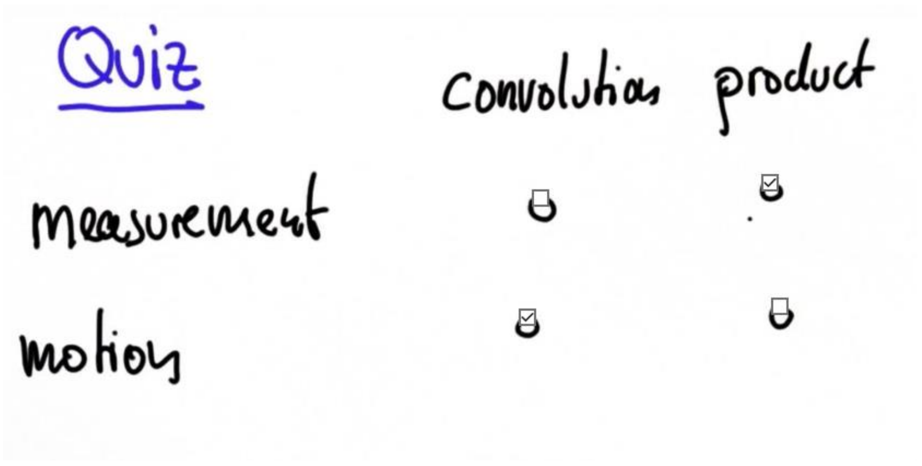
This is identical to the situation before in localization where we got a measurement and then we took am motion.

Here the math changes but the basic principle applies.

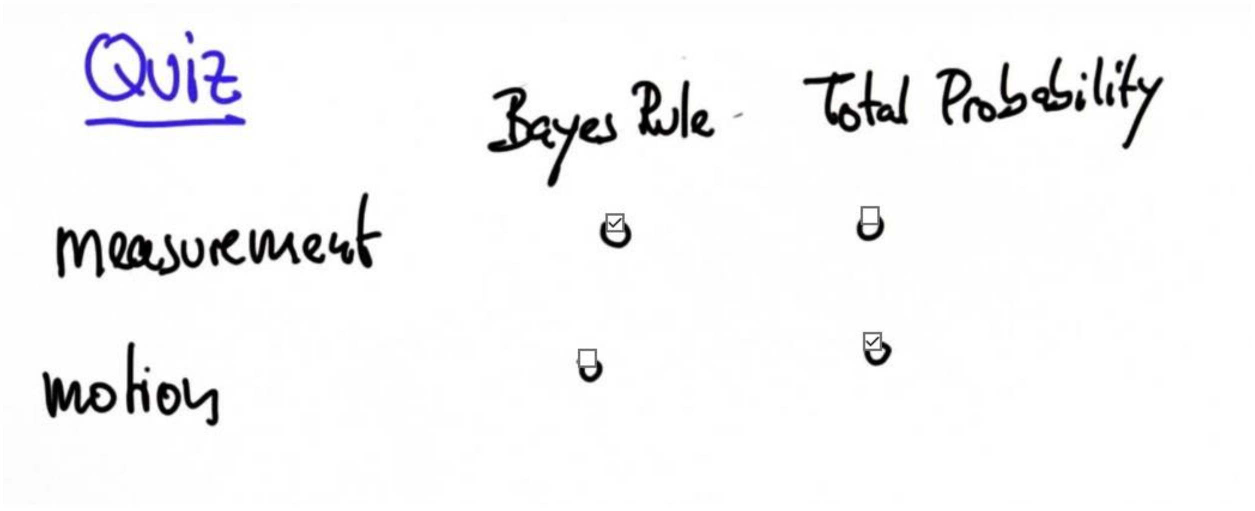
Quiz

You might remember that one of the two steps measurement or motion.

Required a convolution and the other one a product.


Quiz
convolution
product
measurement
motion

From last lesson: measurement meant updating our belief (and renormalizing our distribution). Motion meant keeping track of where all of our probability "went" when we moved (which meant using the law of Total Probability).


Quiz
Bayes Rule
Total Probability
measurement
motion

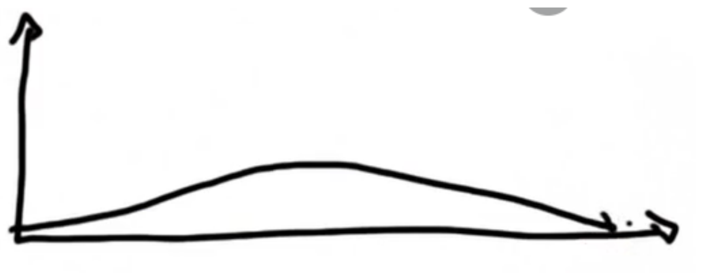
In Kalman filter we iterate measurement and motion.

This is called measurement update and the other called prediction in this we update the Bayes rule in measurement which nothing else the product or a multiplication.

In prediction we use the total probability which is a convolution or simply is addition.

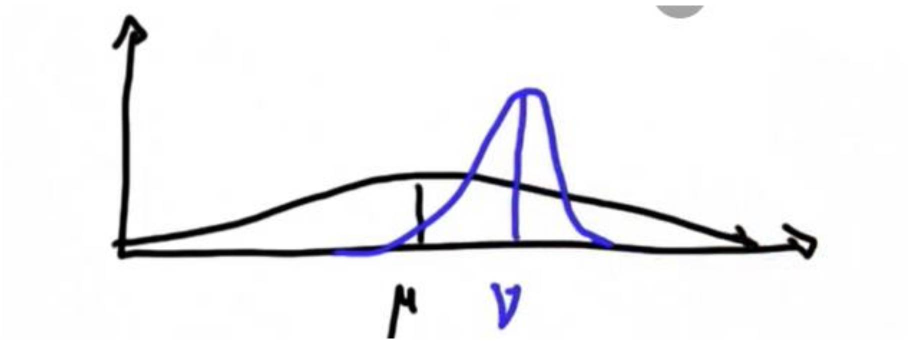
Let`s take first about the measurement the cycle and then the predictions cycles using Gaussian for implementation those steps.

Suppose you are localize another vehicles, and have a prior distribution that looks as follow.



It`s a very wide Gaussian with the mean over here.

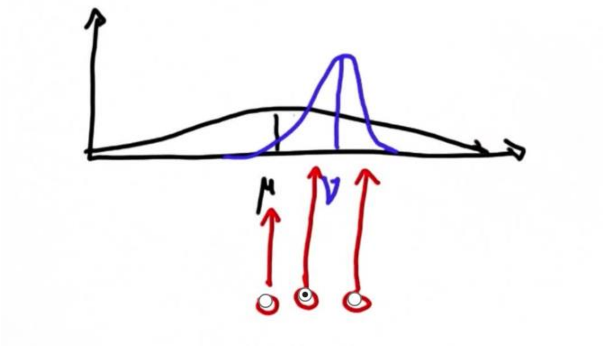
Now , we say get a measurement that tells us somethings about the localizations vehicle



And this example of a much smaller covariance for the measurement.

This is example where in our prior we were fairly uncertain about the location but the measurement told us a bit as to where the vehicle is.

Will the new mean of the subsequent Gaussian be ?



The new belief will be somewhere between the previous belief and the new measurement.

Quiz :

When we graph the new Gaussian I can graph one that very wide and very peaky.

- If I were to measure where the peak of the new Gaussian is , this would be very narrow and skinny Gaussian
- This is one that`s even wider than the two original Gaussian

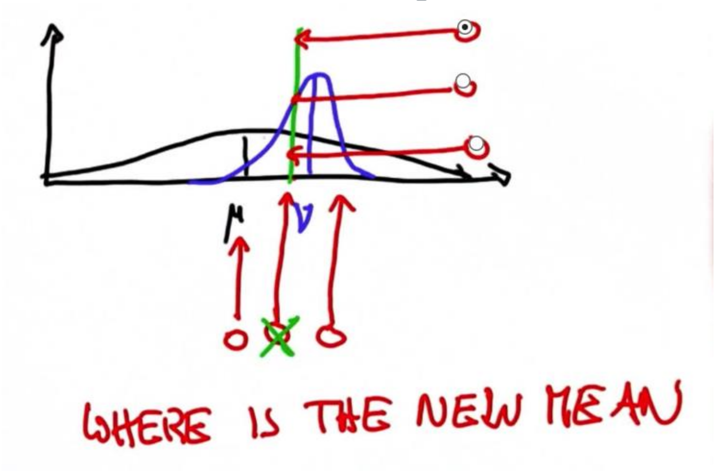
Quiz

Which one do you belief is the correct posterior after multiplying these two Gaussian

If the new peak were that low, it would mean that the additional measurement has made us LESS certain of our position. **But measurements always increase our certainty in our position.**

The new belief will be more certain than either the previous belief OR the measurement.

The takeaway lesson here: more measurements means greater certainty.



That is, covariance is smaller than either of the two covariances in isolation.

Inutility speaking, this is the case because we actually gain information.

- Two Gaussian together have a higher information content than either Gaussian in isolation.

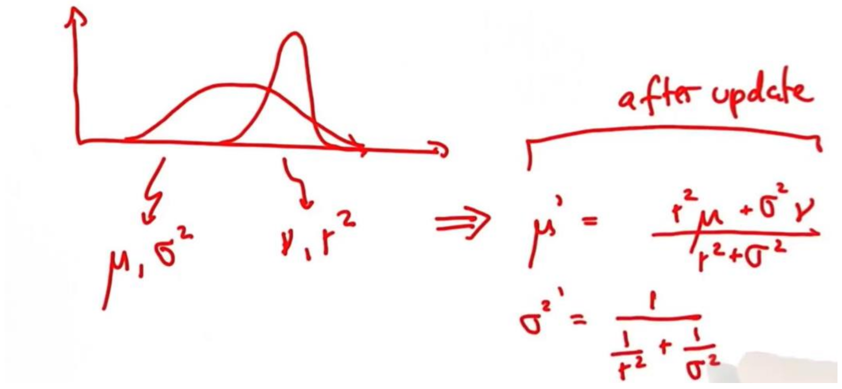
Prove the two Gaussians have a higher peek value

Suppose we **multiply two Gaussians** as in Bayes rule a ***prior*** and a measurement probability.

1- The prior has a mean and variance.
2- And the measurement has a mean and variance

The new mean , it is called prime mean is weighted sum by r-squared and mu is weighted by sigma-squared normalized by the sum of the weight factors.

The new variance term i`m going to write sigma squared prime here for the new one after update the update is given by this equation over here.

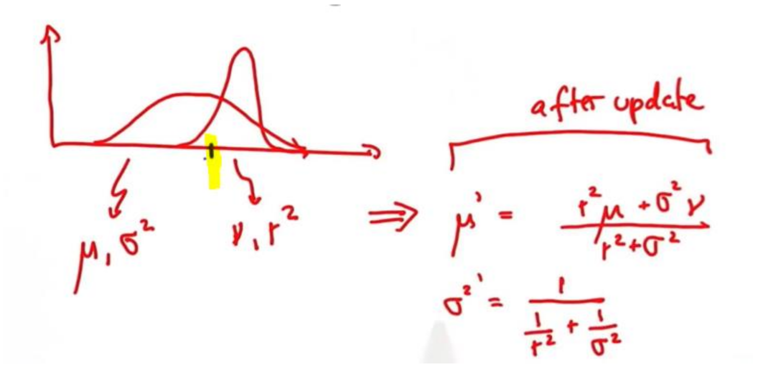

after update
μ, σ^2
ν, r^2
$\mu' = \frac{r^2\mu + \sigma^2\nu}{r^2+\sigma^2}$
$\sigma^{2'} = \frac{1}{\frac{1}{r^2}+\frac{1}{\sigma^2}}$

Let`s put into the action.

We have a weighted mean over here.

Clearly, the prior Gaussian has a much higher uncertainty therefore the variance is higher.

That mean that mu is weighted much, much, larger than the mu so the mean will be closer to the mu than the mu, which means that it will be somewhere over here.

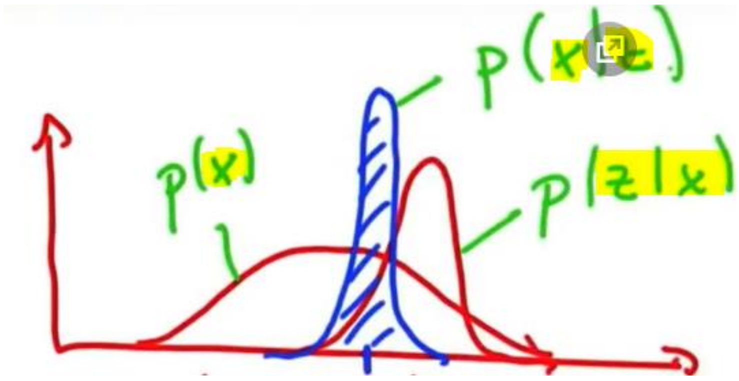

after update
μ, σ^2
ν, r^2
$\mu' = \frac{r^2\mu + \sigma^2\nu}{r^2+\sigma^2}$
$\sigma^{2'} = \frac{1}{\frac{1}{r^2}+\frac{1}{\sigma^2}}$

Interestingly enough, the variance term is unaffected by the actual means.

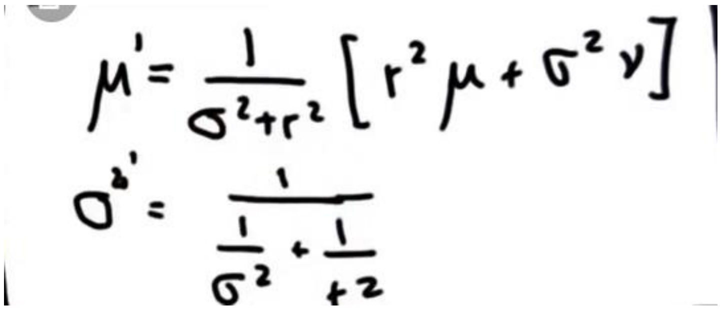
It just uses the previous variance and comes up with a new one that`s even peakier. The result might look like this.

This is the **:Kalman situation** for the measurement update step.

Where this is prior , this is the measurement probability and this is the posterior.



Quiz :



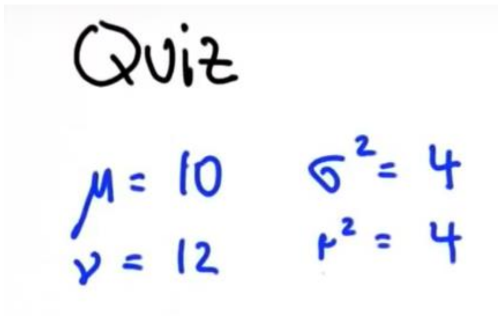
$$\mu' = \frac{1}{\sigma^2 + r^2}\left[r^2\mu + \sigma^2\nu\right]$$

$$\sigma^{2\prime} = \frac{1}{\frac{1}{\sigma^2} + \frac{1}{r^2}}$$

Suppose I use the following Gaussians:

These are Gaussians with equal variance but different means that might look as follow as

Compute for me the



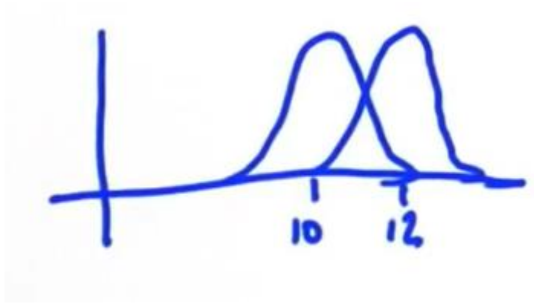
Quiz

$\mu = 10$ $\sigma^2 = 4$

$\nu = 12$ $r^2 = 4$

Prime mean = 11

Prime Variance = 2



Notice that the new mean is between the previous two means and the new variance is LESS than either of the previous variances.

Quiz:

Suppose our mean is 10 and 13 and the variances are imbalanced 8 and 2.

There`s a relatively shallow distribution centered on 10 and a much more peaked distribution centered on 13.

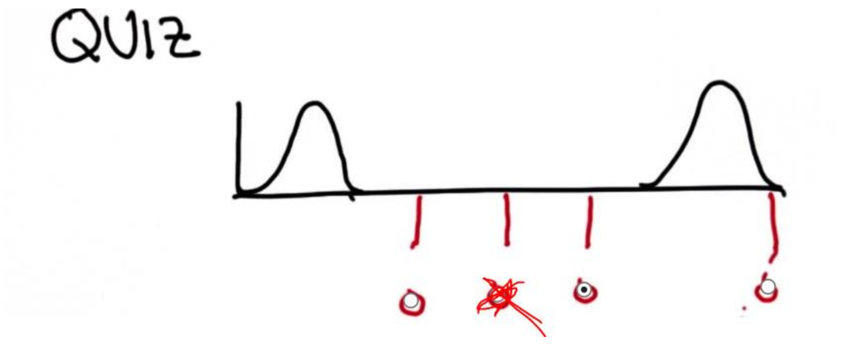
Compute the what the resulting mu prime and sigma =-squared prime are.

New mean = 12.4

And the new variance is 13

12.4 is much closer to 13 than 10 and that`s because the Gaussian centered on 13 has a much narrower variance than the one on 10.

We find the resulting variance is smaller than each of the two variances over here.



QUIZ:

Suppose we have a prior that sits over here and measurement probability that sits over here really far away and same have same covariance.

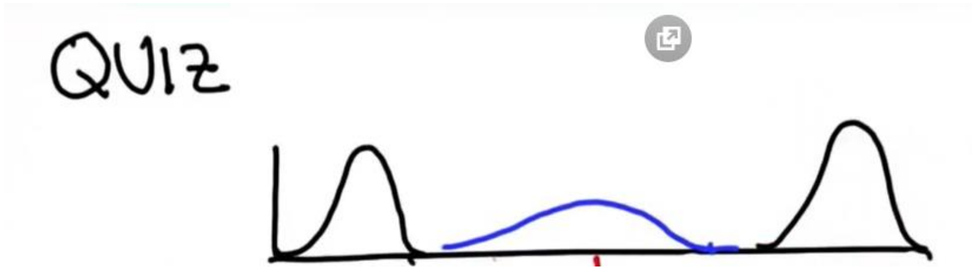
Let me first quiz you where the new mean would be.

Solution

It`s in the straight middle because , these two variances are the same.

So we just average the means

Quiz:

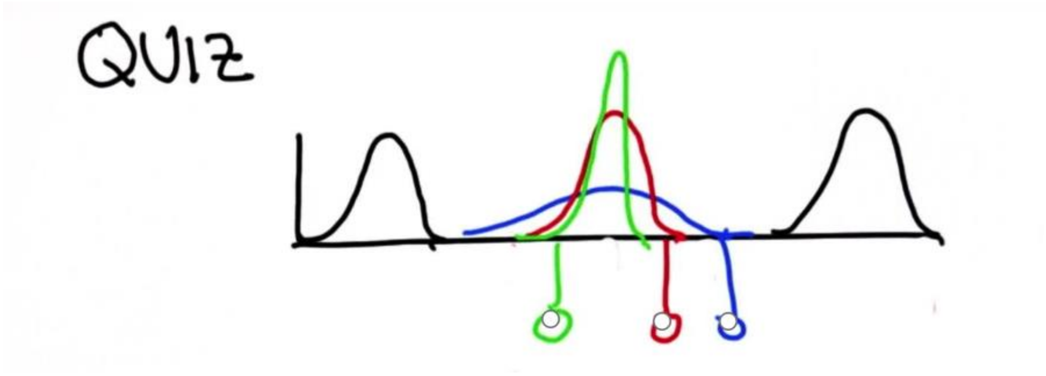


Let me ask the hard question now. Will it be a Gaussian like this where the variance is larger, a Gaussian with the exact the same variance.

Or an even more peaked Gaussian that`s more certain.

Than the two original factors in this calculations

Please check exactly one of those of the three boxes over here.



This can be hard to wrap your head around, but multiple measurements ALWAYS gives us a more certain (and therefore taller and narrower) belief.

Solution is green

You would think if this was your initial measurement probability you really don`t know where you are, are you should pick a very wide Gaussian.

But the truth is our new sigma-squared is obtained independent of the means.

It`s formula over here.

Now because both means are the same, this resolves to 1 over 1/sigma square 2

That`s the same as sigma over 2.

Which means a new variance squared is half of the old one.

That makes it a narrower Gaussian, so the green here that`s the most peaked is indeed the correct answer.

This is very counter-intuitive but now we understand why.

I Hope you feel comfortable with the fact that we have actually gotten more information about the location which is manifest by a more focused estimate.

Let`s write program.

In which we calculate the new mean and the new variance term.

Implements those equations so that we can test them.

$$\mu' = \frac{1}{\sigma^2 + r^2}\left[r^2\mu + \sigma^2\nu\right]$$

$$\sigma^{2\prime} = \frac{1}{\frac{1}{\sigma^2} + \frac{1}{r^2}}$$

Create the function update, that take as input a mean and a variance for the first distribution and mean and variance for the second distribution and outputs the new mean and the new variance of the product of those.

Here, I am testing it with a mean of 10 and variance of 8 and the mean of 13 and a variance of 2

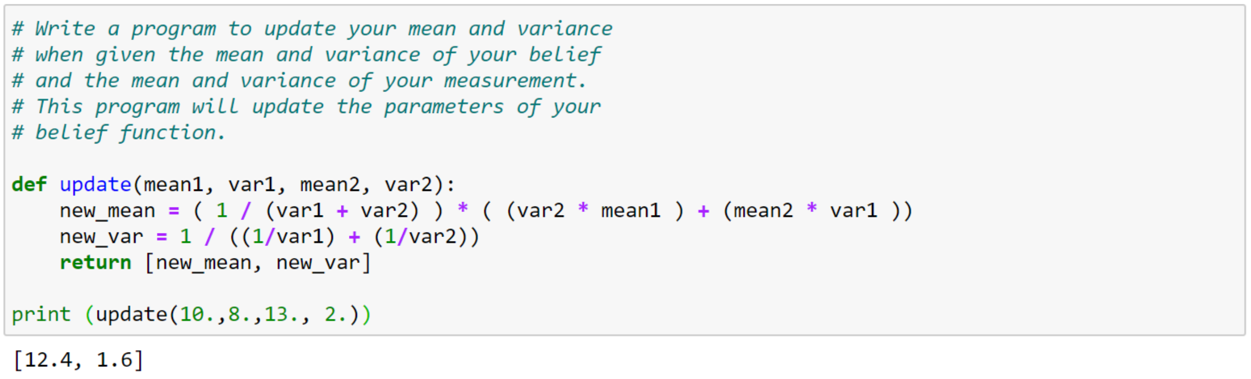
```
# Write a program to update your mean and variance

# when given the mean and variance of your belief

# and the mean and variance of your measurement.

# This program will update the parameters of your

# belief function.


def update(mean1, var1, mean2, var2):
```

new_mean = (1 / (var1 + var2)) * ((var2 * mean1) + (mean2 * var1))

new_var = 1 / ((1/var1) + (1/var2))

return [new_mean, new_var]

print update(10.,8.,13., 2.)



```
# Write a program to update your mean and variance
# when given the mean and variance of your belief
# and the mean and variance of your measurement.
# This program will update the parameters of your
# belief function.

def update(mean1, var1, mean2, var2):
    new_mean = ( 1 / (var1 + var2) ) * ( (var2 * mean1 ) + (mean2 * var1 ))
    new_var = 1 / ((1/var1) + (1/var2))
    return [new_mean, new_var]

print (update(10.,8.,13., 2.))
```

```
[12.4, 1.6]
```

You programmed essential update step in the Kalman filter the measurement update step.

The other one the predication step and motion step is much easier to program.

Motion update

We knew there was a **measurement update** and a **motion update**, which is called prediction.

And we know that the measurement update is implemented by multiplication, which is the same Bayes rule.

And the motion update is done by total probability or an addition .

So we tackled the more complicated case.

This is part the hard part mathematically when we solve Bayes rule we even derived it mathematically,

And you were able to write a computer program that implements this step of the Kalman filter.

I don`t want to go into too much depth here.

But the total conditional it is really easy step Let me write it down for you.

- **Suppose you live in a world like this.**
- **This is current best estimate of where you are, and this is your uncertainly**
- **Now say you move to the right side a certain distance and the motion itself has its own set of uncertainty.**

- **<u>Then you arrive at the prediction that adds the motion of command to the mean, and it has an increased uncertainty over the initial uncertainty.</u>**

Intuitively it make sense

If you move to the right by the distance, in expectation you are exactly where you wish to be.

But you have lost information because your motion tends to lose information as manifest by this uncertainty over here.

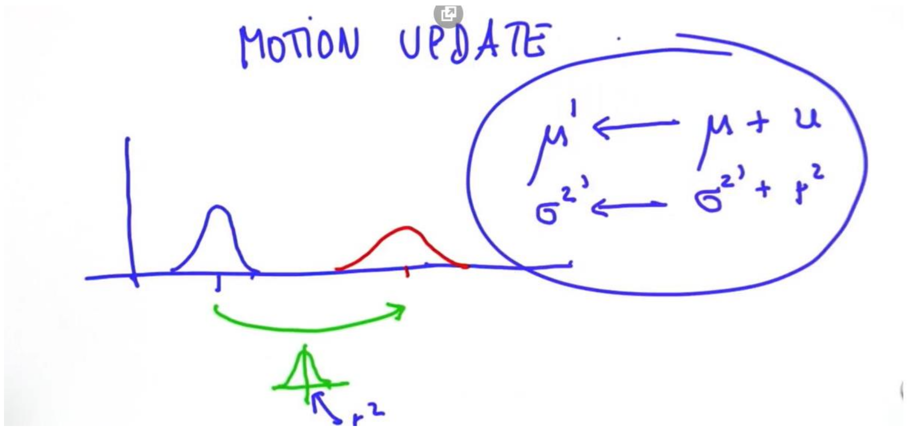
It is really easy math.

You new mean is your old mean + the motion , often called U.

So, if you move over 10 meters, this will be 10 meters.

And your sigma is your old sigma square + the variance of the motion Gaussian.

Just addition math.


MOTION UPDATE
$\mu' \leftarrow \mu + u$
$\sigma^{2'} \leftarrow \sigma^{2'} + r^2$
r^2

In summary

1- we have a Gaussian
2- We have a Gaussian for the motion with and r-square as its own motion uncertainty, and the resulting Gaussian in the prediction step just adds these 2 things gaussian initial + Gaussian motion equal mean + U and sigma square + R square

Quiz:

We have a Gaussian before the prediction step. Which mean is 8

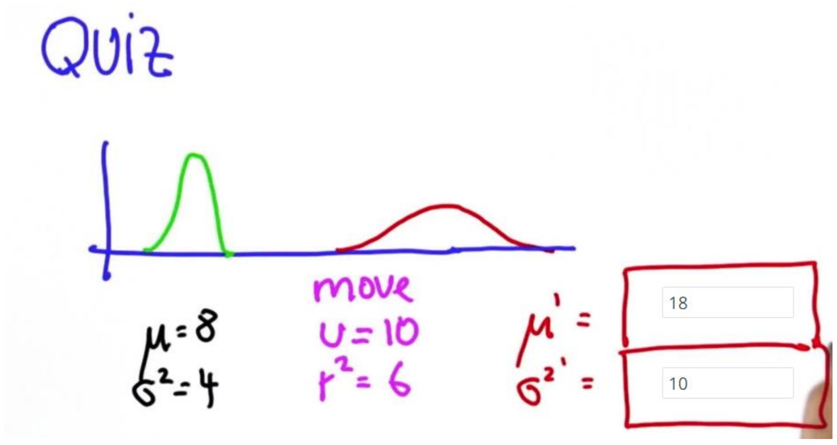
And sigma square is 4

And move the right to 10 and with the motion uncertainty of 6.

Now Describe to me the predictive Gaussian

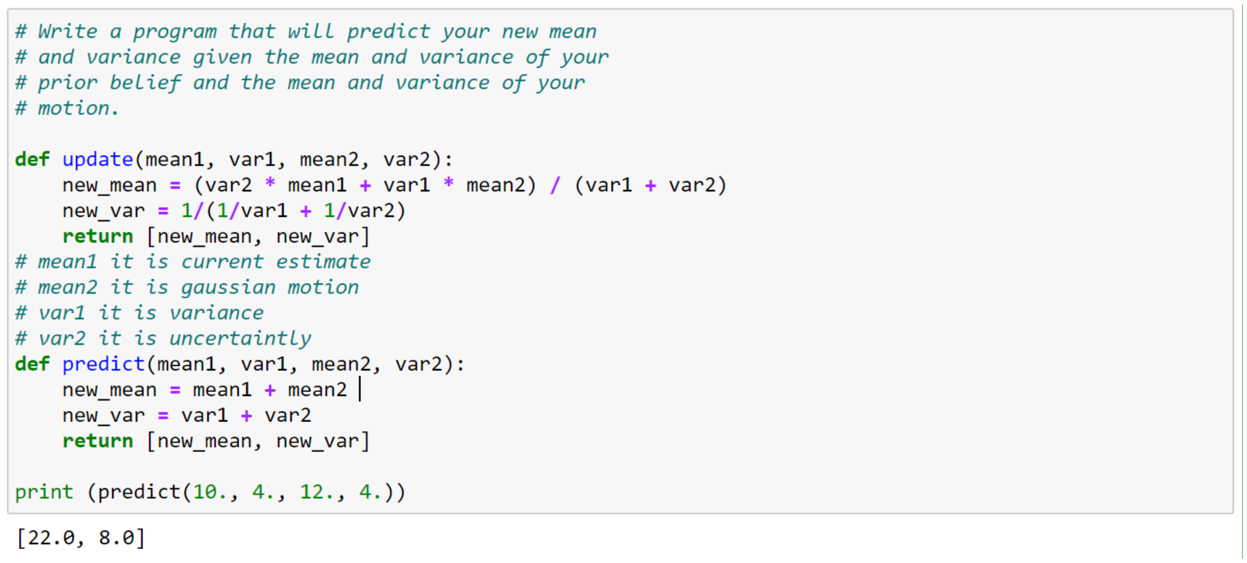
New mean

New sigma-



Quiz:

Create the predict function which take current estimate and its variance and the motion and its uncertainty and compute to knew updated prediction, mean and variance.



```
# Write a program that will predict your new mean
# and variance given the mean and variance of your
# prior belief and the mean and variance of your
# motion.

def update(mean1, var1, mean2, var2):
    new_mean = (var2 * mean1 + var1 * mean2) / (var1 + var2)
    new_var = 1/(1/var1 + 1/var2)
    return [new_mean, new_var]
# mean1 it is current estimate
# mean2 it is gaussian motion
# var1 it is variance
# var2 it is uncertaintly
def predict(mean1, var1, mean2, var2):
    new_mean = mean1 + mean2 |
    new_var = var1 + var2
    return [new_mean, new_var]

print (predict(10., 4., 12., 4.))
```

```
[22.0, 8.0]
```

Kalman Filter code:

Create a program take a two function

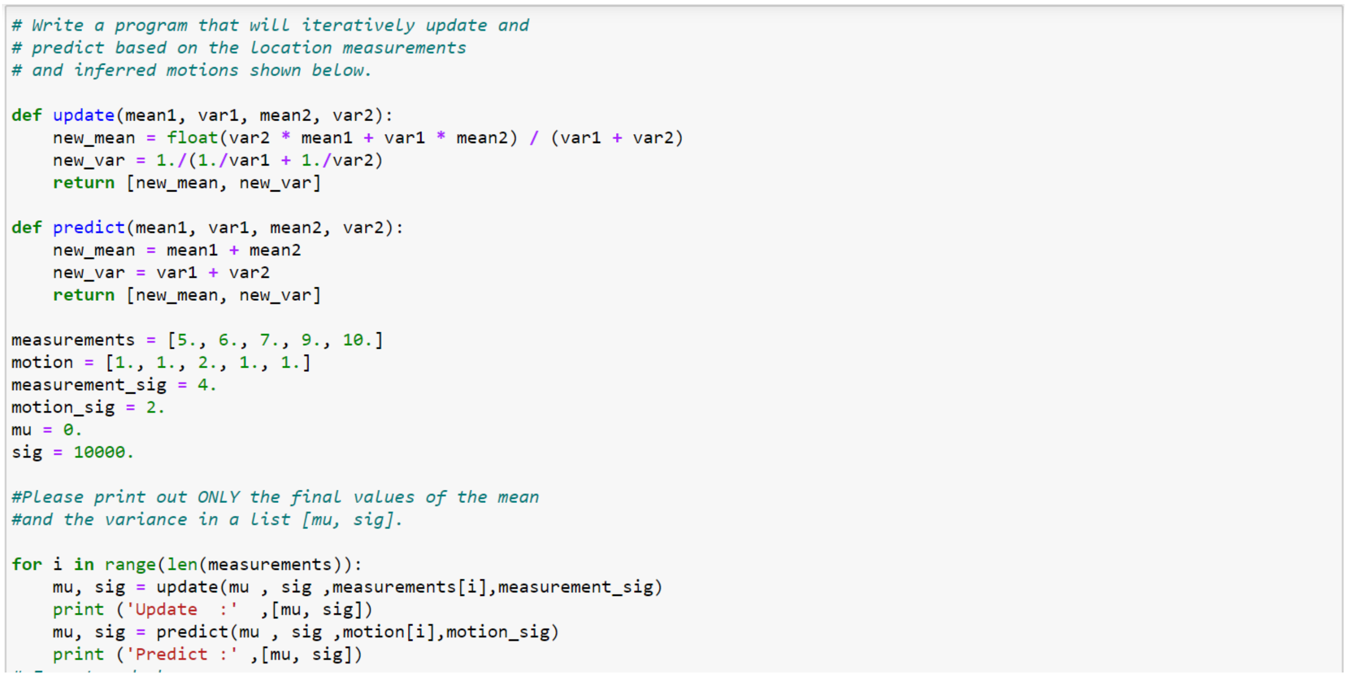
1- Update
2- Predict

And feeds into a sequences of measurements and motions.

This all would out really well if the initial estimate was 5.

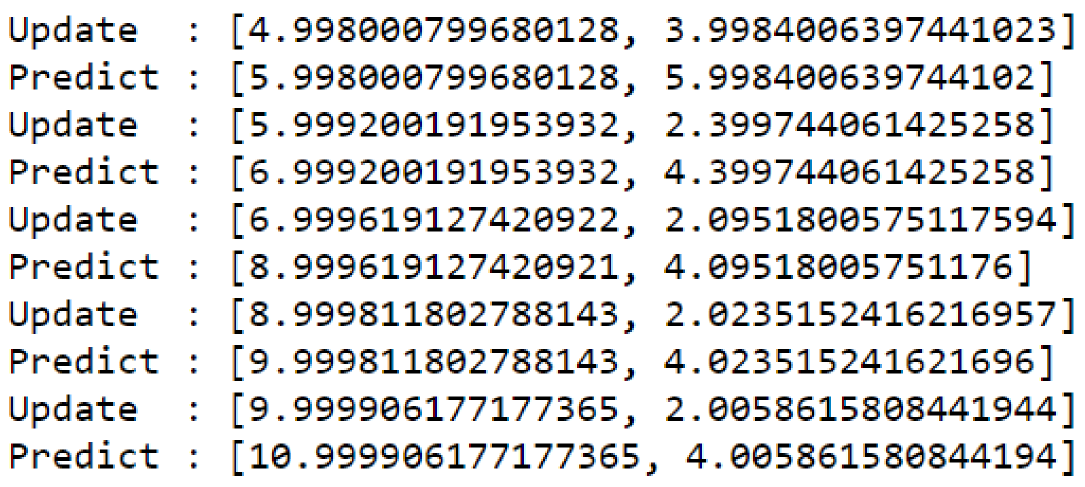
Motion sigma is the motion uncertainty.

1- **First estimate for position should basically become 5 , 499 and the is your initial uncertainty is so large. The estimated is dominated by first measurement.**
   You uncertainty shrinks to 3.99, which is slightly better than the measurement uncertainty,
2- **You then predict that you add 1, but uncertainly increases to 5.99 . which the motion of uncertainty of 2**
3- **Measurements = [5 , 6 , 7 , 9 , 10]**
4- **You update again based on the measurement 6 , you get your estimate of 5.99**
5- **Finally, output result   your measurement 10 and move 1.**



```
# Write a program that will iteratively update and
# predict based on the location measurements
# and inferred motions shown below.

def update(mean1, var1, mean2, var2):
    new_mean = float(var2 * mean1 + var1 * mean2) / (var1 + var2)
    new_var = 1./(1./var1 + 1./var2)
    return [new_mean, new_var]

def predict(mean1, var1, mean2, var2):
    new_mean = mean1 + mean2
    new_var = var1 + var2
    return [new_mean, new_var]

measurements = [5., 6., 7., 9., 10.]
motion = [1., 1., 2., 1., 1.]
measurement_sig = 4.
motion_sig = 2.
mu = 0.
sig = 10000.

#Please print out ONLY the final values of the mean
#and the variance in a list [mu, sig].

for i in range(len(measurements)):
    mu, sig = update(mu , sig ,measurements[i],measurement_sig)
    print ('Update  :'  ,[mu, sig])
    mu, sig = predict(mu , sig ,motion[i],motion_sig)
    print ('Predict :' ,[mu, sig])
```

Soution



```
Update  : [4.998000799680128, 3.9984006397441023]
Predict : [5.998000799680128, 5.99840063974402]
Update  : [5.999200191953932, 2.399744061425258]
Predict : [6.999200191953932, 4.399744061425258]
Update  : [6.999619127420922, 2.0951800575117594]
Predict : [8.999619127420921, 4.09518005751176]
Update  : [8.999811802788143, 2.0235152416216957]
Predict : [9.999811802788143, 4.023515241621696]
Update  : [9.999906177177365, 2.0058615808441944]
Predict : [10.999906177177365, 4.005861580844194]
```

Explain:

Measurement sigma = **measurement uncertainty**

The first output : 4.99 it make me sense because we had a huge a initial uncertainly, and of 5 with a relativity measurement uncertainly.

And affecting the resulting sigma term is 3.99 which is better than 4 and 1000 slightly better than 4.

We are slightly more certain than the measurement itself.

We now apply the motion of 1 we get to 5.9 our uncertainty increase by exactly 2 from 3.9 to 5.98

And the next update comes in at 6 and it give us a measurement of 5.99and now reduce uncertainty of 2.39

This code for implements for full Kalman filter for 1D.

And then you apply it to a measurement sequence and a motion sequences with certain uncertain associates .

Suppose you are really certain about the initial position.

Sig = 0.0000001 it`s wrong but it should be 5 first estimate.

And new we assume a really small uncertainly.

Guess what`s going to happen to the final prediction? Run code

I find this has an affect on the final estimate.

And the way this take place is initially after our first measurement update , we believe in the position of 0 but really small uncertainly even smaller than this one over here

Now we understand a lot of about the 1D Kalman Filter.

you understand how to incorporate measurements and incorporate motion.

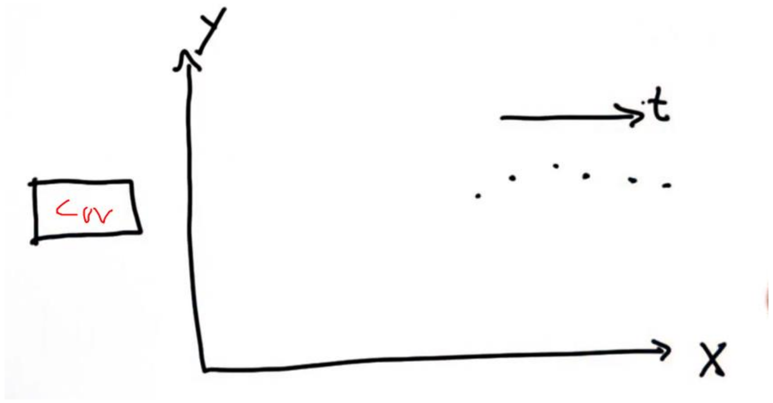
In reality we often have many Ds and then things become more involved.

i`m going to just tell you how things work with an example

I`m great to estimate in higher dimensional state spaces.

Suppose you have 2-Dimansional state space of x and y like camera image.

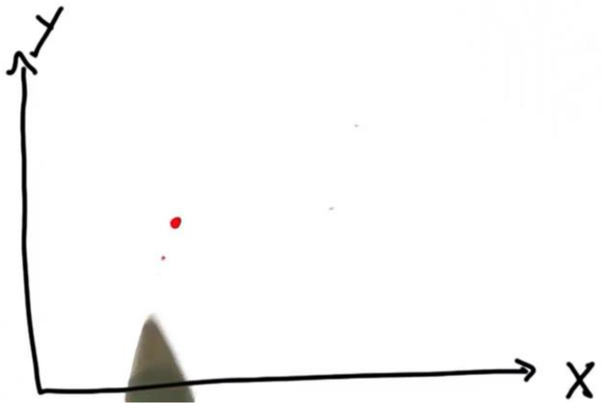
In our case we might have a car that uses a radar to detect the location of a vehicle over time.



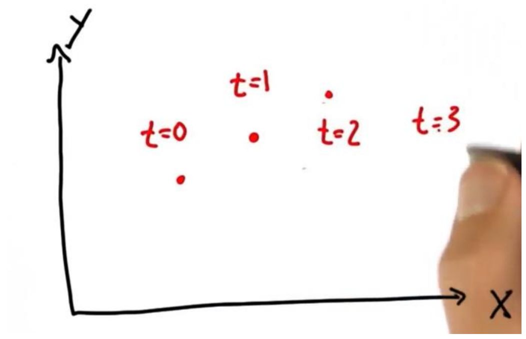
Then what the 2d Kalman filter affords you is somethings really amazing.

Suppose the time t = 0 , you observe the object of interest to be at this coordinate.

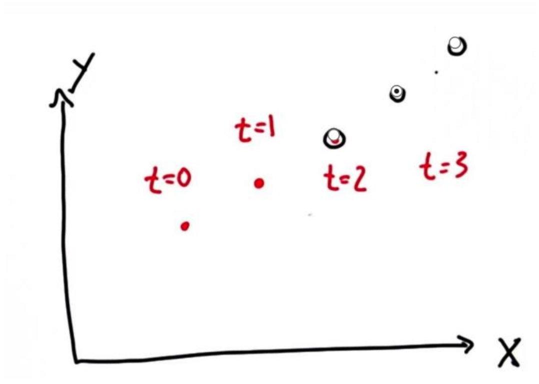
This might be another car in traffic for the google self-driving car.



One time step later , you see it over here and another see later you see it right over here.



Let me give you 3 different places



What Kalman filter does for you , if you do estimation and high dimensional spaces, is to not go into x and y spaces, but allows you to implicitly figure out what the velocity if the object and then uses the velocity estimate to make a really good prediction about the future.

- ***Now notice the sensor itself only see position [It never see the actual velocity]***
- ***Velocity is inferred from seeing multiple positions.***

So one of most amazing things about Kalman filters in tracking applications is it`s able to figure out , even though it never directly measure it.

The velocity of the object and from there is able to <u>make predictions about future locations that incorporate velocity.</u>

<u>That`s one of most the reasons that Kalman filter are such a popular algorithm in artificial intelligence and in control theory in large.</u>

Kalman Filter Land

To explain how this work , I have to talk about high dimensional gaussian.

These are often called multivariate Gaussians.

The mean is now a vector with 1 element for each of the variance.

Mean = [mu0, mu1 , mu2 , mu3 , mu4]

The variance here is replaced by what`s called a co-variance, and it`s matrix with D rows and D columns.

If the dimensionality of the estimate is D.

The formula is somethings you have to get used to.

Let me explain it to more intuitively.

Here`s is a 2 dimensional space.

A 2- dimensional gaussian is defined over that space, and it`s possible to draw the counter lines of the gaussian it might look like this.

1- ***The mean of gaussian is this X0 , Y0 pair, and the co-variance now defines the spread of the gaussian as indicates by the counter lines.***
2- ***A gaussian with a small amount of uncertainly might look like this.***

It might be possible to have a fairly small uncertainty in 1 dimension, but a huge uncertainty in the other.

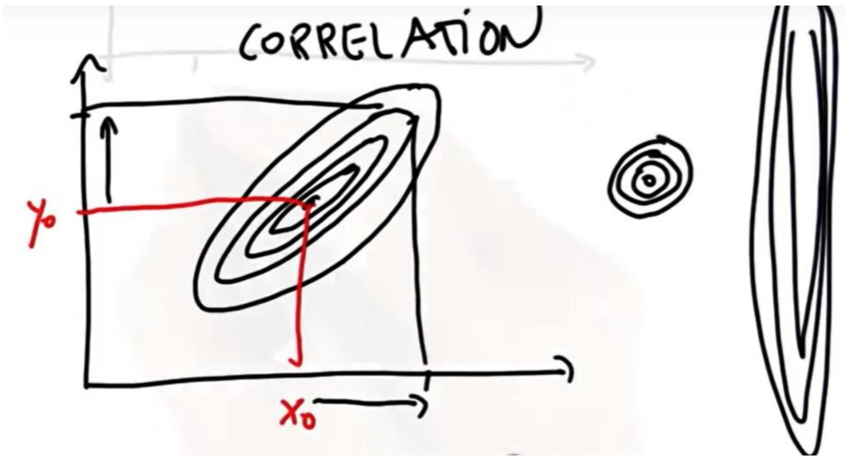
Huge uncertainty in the x-dimension is small and the y -dimension is large. And when the gaussian is tilted as showed over here, then the uncertainty of x and y is correlated which means if I get information about x.

It actually sits over here-that would make me believe that y probably sits.

Somewhere over here That's called correlation.

I can explain to you the entire effect of estimating velocity and using it in filtering.

Using gaussian like these.



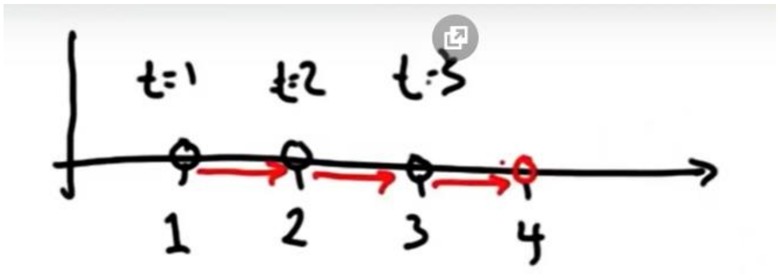
I can explain to you the entire effect of estimating velocity and using it in filtering and it become really simple.

The problem `m going to choose is a 1-dimesnional motion example let assume t = 1 and t = 2 over here.

And t = 3

Then you would assume that at t = 4 the object sits over here, and the reason why would assume this is even though it`s just seen these different discrete locations, you can infer from it there is actually velocity that drives the object. to the right side to the point over here.

Now how does the Kalman filter address this ? This is the true beauty of the Kalman filter.

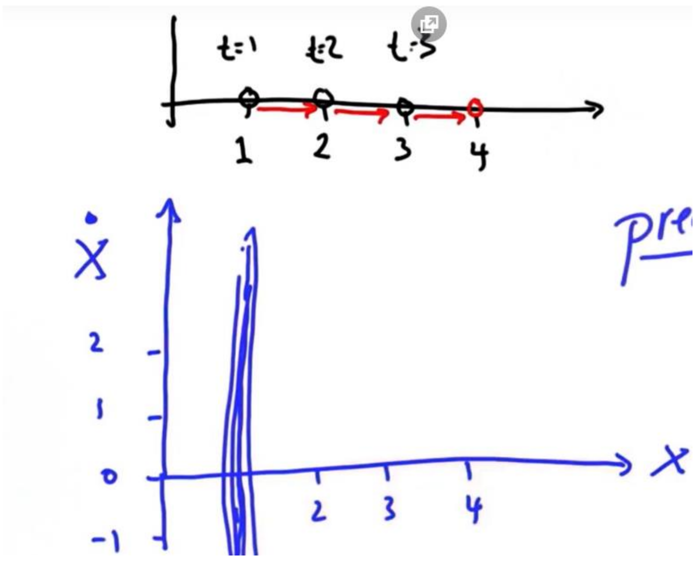


In Kalman filter Land we are going to build 2-dimensional estimate.

1- *One for the location*
2- *One for the velocity x dot*

Note: the velocity it can be zero , it can be negative or it can be positive.

1- *If initially I know my location , but not my velocity, then I represented it with a Gaussian that`s elevated around the correct location, but really broad in the space of velocities*



In the prediction step , I don`t <u>know my velocity</u> , so I can`t possibly predict for the location

I`m going to assume.

But miraculously they will be some thing interesting correlation.

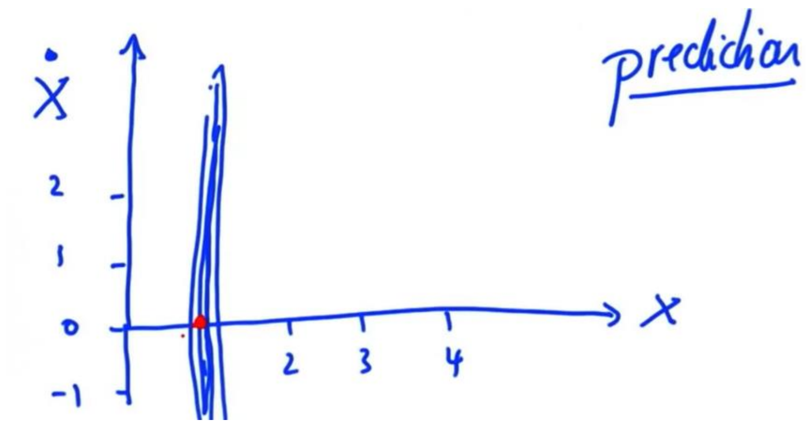
So let`s for a second just peak a point on this distribution over here.

1- ***Let me assume my velocity is 0 Note (of course , in practice I don`t know the velocity)***

But let me assume for a moment the velocity is 0

Where would my posterior be after the prediction?

Well we know we started in location 1 , the velocity of zero , so my location would likely be here.



Now let`s change my belief in velocity and pick a different one.

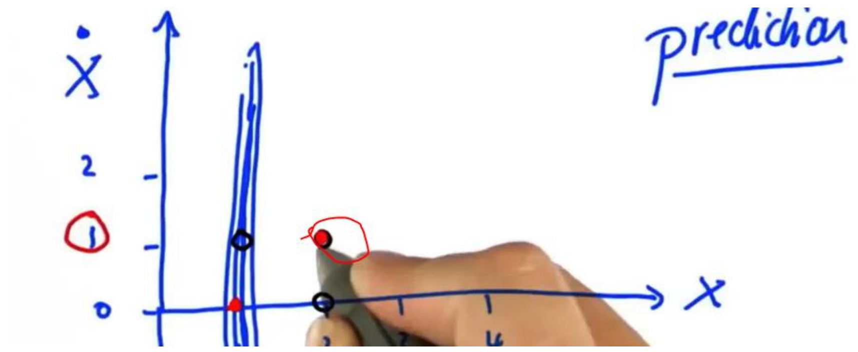
Let`s say the velocity is 1. Where would my prediction be 1 time step later staring at location and velocity 1

Our prediction is that we would move forward in the x direction by one and that our velocity is still one.

The answer is why ?

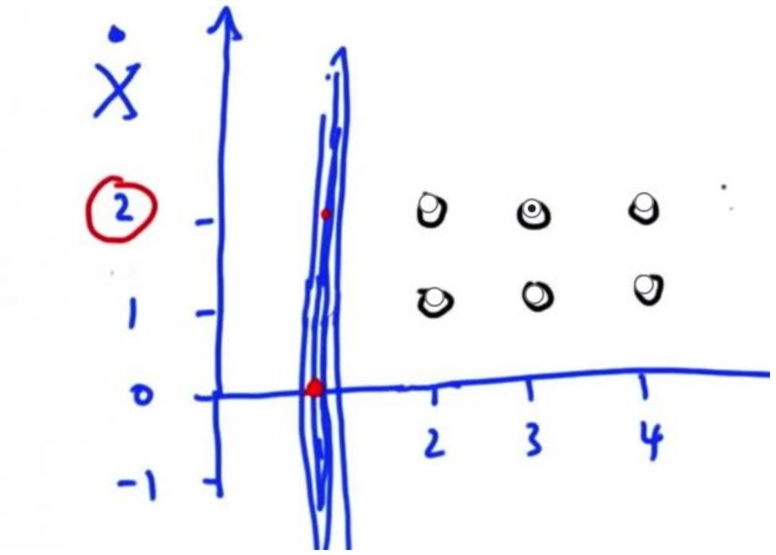
If a car starting point is the point over here, for which we know the location is 1 and the velocity is 1 and if we predict 1 time step in the future, then for the prediction we know the location will be 2

And the velocity might be a little uncertain but it says about the same.



Let me consider a velocity of 2 , which mean this is starting let me ask you where you would expect among those choices.

We'd expect our velocity to remain unchanged, but we should move forward in the x direction by two (since the velocity was two).



Again, this model assumes that in the absence of more knowledge, the velocity shouldn`t really change.

More Kalman Filter.

When you put all this together you find that all these possibilities on the Gaussian over here.

Link to a Gaussian that looks like this.

This is really interesting 2-dimensional Gaussian which you should really think about.

Clearly, if I were to project this Gaussian uncertainty into the space of possible locations, I can`t predict a thing. It`s impossible to predict where the object is the reason is I don`t know the velocity.

Also, Clearly if I project this Gaussian into the space of x dot, it is impossible to say what the velocity is.

- ***A Single observation or single prediction is insufficient to make that observation.***

However what we know is our location is correlated to the velocity.

The faster I move, the further on the right is the location. This Gaussian expresses this.

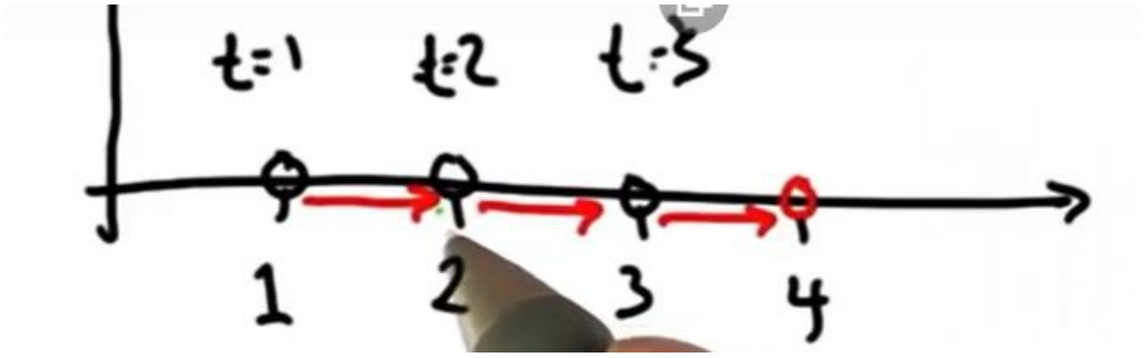
If I , for example figure out that my velocity was 2 then I was able , under this Gaussian , to really nil that my location is 3.

That is really markable.

You still haven`t figured out where you are , and you haven`t figured out how fast you are moving.

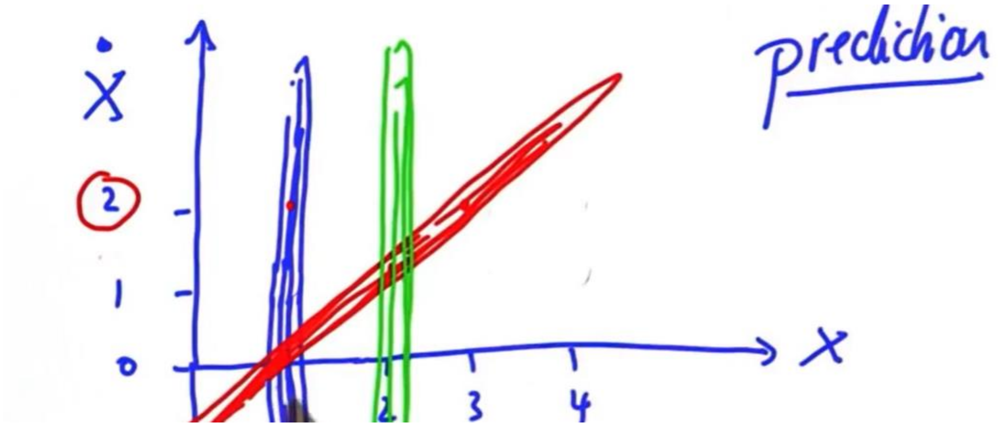
But we have the learned so much about the relation of these 2 things with the Gaussian.

To understand how powerful this is let`s now flod in the second observation at time t = 2.



This is observation tell us nothing about the velocity and only something about the location.

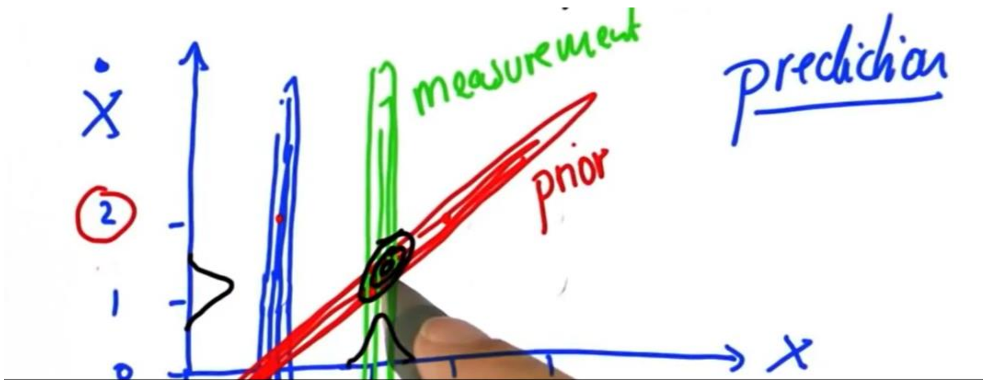
So if I were to draw this as a Gaussian it`s a Gaussian just like this.



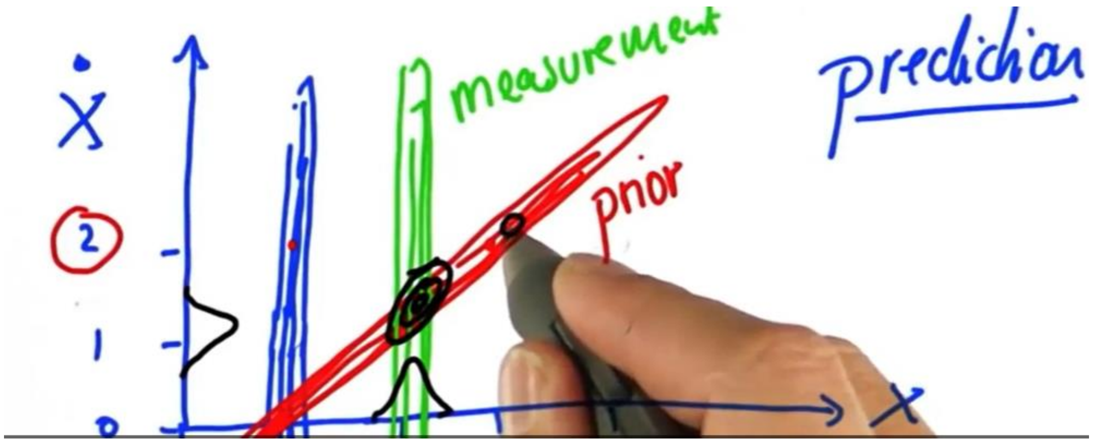
Which says something about the location but not about the velocity.

But if we multiply by prior from the prediction step with the measurement probability, then miraculously I get a Gaussian that's sits right over here.

This Gaussian now has a really good estimate what my velocity is and a really good estimate where I am.



if I take this Gaussian and predict 1 step forward then I find myself right over here.



That`s exactly the effect we have over here.

After that I get a gaussian like this , I predict right over here.

Think about this.

This is really deep insight about how Kalman filters work.

In particular, we have only been able to observe 1 variable

We have been able to measure observation to their location to infer this other variable.

And the way we have been able to infer is that there`s a set of physical equations.

Which say that my location after time steps is my old location + my velocity.

This set of equations has been able to propagate constraints from subsequent measurements back to this unobservable variable , velocity so we are able to estimate the velocity as well.

This is really key to understanding Kalman filter.

It is a key to understanding a Google self -driving car , estimate locations of the other cars and is able to make predictions.

Even if it`s unable to measure velocity directly.

The variables of a Kalman filter they are often called states because they reflect states.

Of the physicals role like where is the other car and fastest moving.

They separate into 2 subsets

1- ***The observables like the momentary location***
2- ***And the hidden which in our example is the velocity which I car never directly observe***

But because those 2 thing interact subsequent observations of the observable variables give us information about these hidden states so , we can also estimate What these Hidden variable are.

So from multiple observation of places of the object the location we can estimate how fast it`s moving

That is actually true for all of the different filters but because Kalman filter happen to be very efficient to calculate when we have a problem like this.

You tend to often use just a Kalman filter.

In addition, the formula `x' = x + ẋ` shown a 2:30 should include a multiplication by time units: `x' = x + Δtẋ'`

```
T = 2 it is observation I draw the Gaussian which says something
about the location but not about the velocity.
```

```
We have been to measure observation to infer this other variable
and the way we have been able to infer s that there`s a set of
physical equations which say that the location after times step,
is my old location + any velocity
```

```
Observation: is my location in time step
```

```
Hidden is the velocity which I can never directly observe.
```

Design Kalman Filter

When we design a Kalman Filter, you need effectively 2 things.

1- *For State you need, a [state translation function] and that`s usually a matrix so we are now in the world of Linear algebra.*
2- *For a measurement, you need a measurement function.*

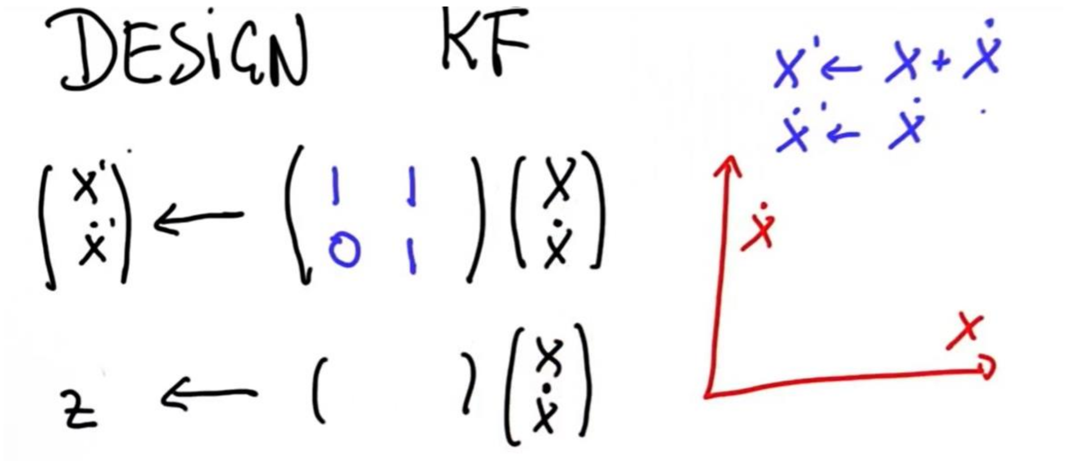
Let me give you those for our example of the one demotion of an object .تخفيض رتبة كائن واحد

Be known that the new location is the old location + velocity.

Turns into the matrix.

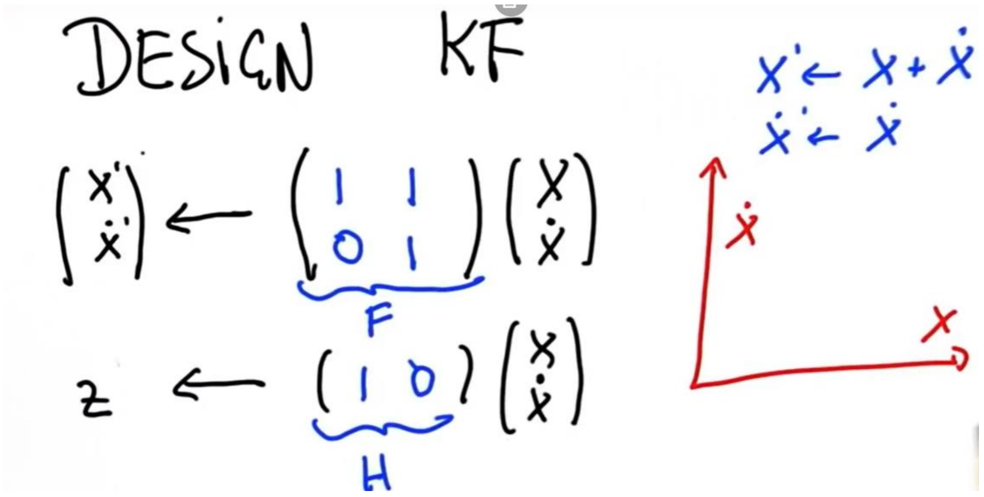
The new velocity should just be the old velocity which gives us 0 over here and 1 over here.

If you multiply this matrix by this vector, this is exactly what you are getting.


DESIGN KF
$x' \leftarrow x + \dot{x}$
$\dot{x}' \leftarrow \dot{x}$
$\begin{pmatrix} x' \\ \dot{x}' \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$z \leftarrow (\quad) \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$\dot{x}$
x

For the measurement, we only observe the first component of place not velocity, and that uses a matrix like this.

This matrix would be called F and this called H.


DESIGN KF
$x' \leftarrow x + \dot{x}$
$\dot{x}' \leftarrow \dot{x}$
$\begin{pmatrix} x' \\ \dot{x}' \end{pmatrix} \leftarrow \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{F} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$z \leftarrow \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{H} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
$\dot{x}$
x

The Actual update equations for a Kalman filter are involved.

There`s a prediction step where I take my <u>best estimate</u> x, multiply it with a <u>state translation matrix</u> F and I add whatever motion I know U = [motion vector].

That give me my new X , x = F x + u

I also have a covariance that characters my uncertainty

P = uncertainty covariance

P new = f.p.f transpose

There`s also a measurement update step where we use the measurement z.

We compare the measurement with our prediction where H is the measurement function that maps the state to measurement.

Error = z - H.x

The error is mapped into a matrix <u>S</u> which is obtained by projecting the system uncertainty

S = H.P.H transpose into the measurement space using the measurement function projection + the matrix R the character of measurement noise.

R = measurement noise

This is then mapped into a variable called K, which is often called Kalman gain

K = P . H transpose . S^{-1}***, Where we invert the matrix s, and then finally we actually update our estimate and our uncertainty***

X bar = x + (K.y(error)) and then finally , we actually update our estimate and our uncertainty

P bar =(I – K.H) .P using what ought to be the most cryptic equation that you `ve seen in a long time.

I = identity matrix

Now I wrote this down so that you have a complete definition, but this is something you should not memorize this math you to higher dimensional spaces,

There`s a set of linear algebra equation that implement the Kalman filter

The assignment

```
x_prime = x + x_dot
```

assumes that the time interval delta_t is equal to 1. A more general formula is

```
x_prime = x + x_dot*delta_t
```

after assignment

So this completes my unit on Kalman filters. You learned actually quite a bit.

You learned about Gaussians, how to do measurement updates using multiplication,

How to do prediction or state transitions using convolution, and you even implemented your first Kalman filter , which is really super cool.

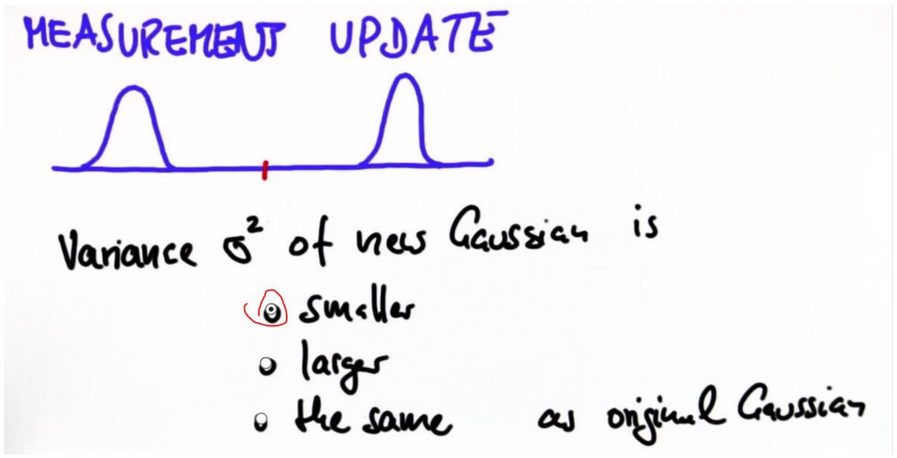
You have implemented it in the context of vehicle tracking and use this to <u>estimate a non-observable velocity for measurement data.</u>

Assignment:

Measurement Update : Kalman Filter

Let`s start with 2 Gaussian and say they have identical variance we know that the resulting Gaussian`s mean will be just at the center between those 2 Gaussians.

I wonder if the variance summation to end 2 is now smaller , larger or the same as either of the individual Gaussian.

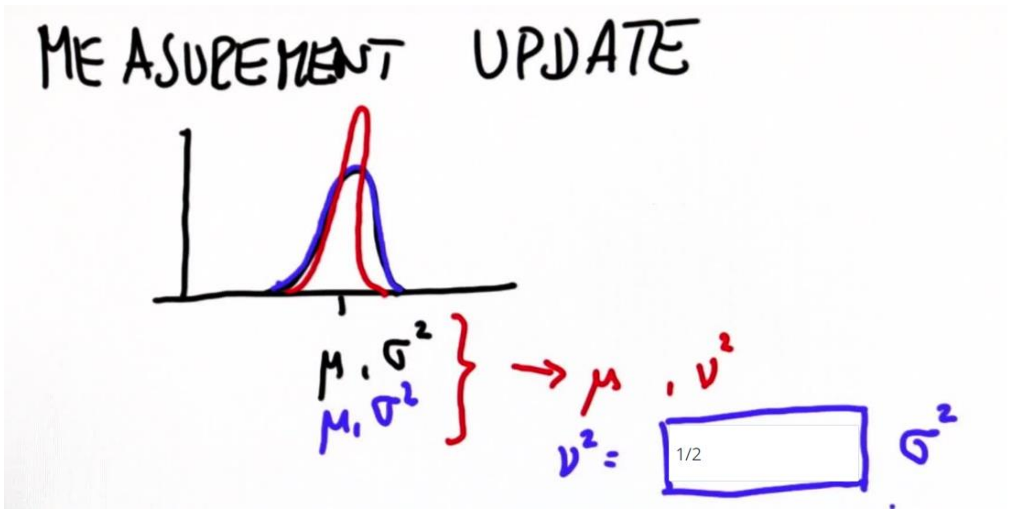


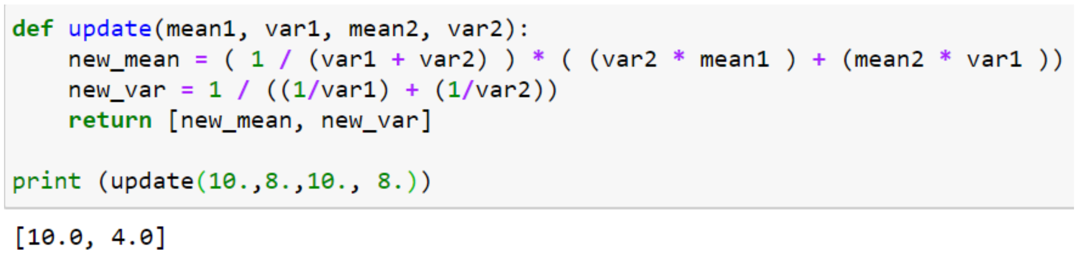
Quiz: Measurement update

Say we have a prior of Gaussian with mean , mu and covariance sigma square and our measurement has exactly the same mean and same covariance.

Suppose we multiply both and get a new mean which is the same as the old mean and we get a new covariance , sigma square which as you know , corresponds to a more a peaked Gaussians.

I want you to express nu squared as a multiple of sigma squared.





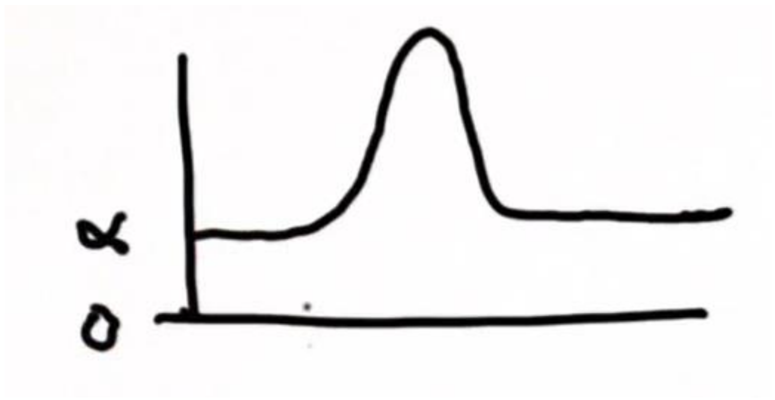
```
def update(mean1, var1, mean2, var2):
    new_mean = ( 1 / (var1 + var2) ) * ( (var2 * mean1 ) + (mean2 * var1 ))
    new_var = 1 / ((1/var1) + (1/var2))
    return [new_mean, new_var]

print (update(10.,8.,10., 8.))
```

```
[10.0, 4.0]
```

Quiz: Heavy tail Gaussian

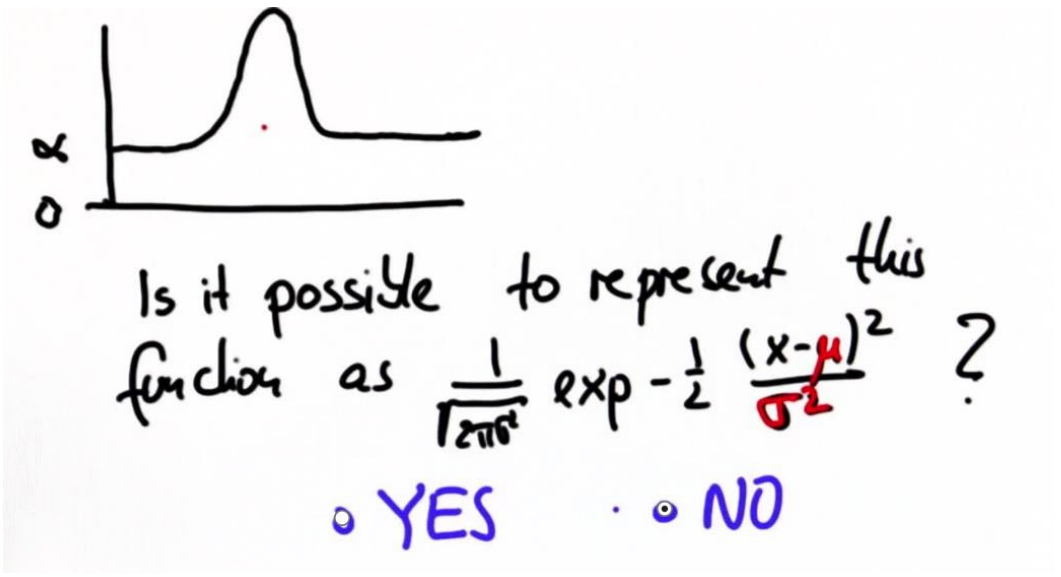
I have another question Gaussian question this is called : Heavy tail Gaussian +/- infinity

The Gaussian levels off at some finite value alpha, as opposed to zero



Is it possible to represent this function as a Gaussian ? this is formula for the Gaussian again

Particularly , can you find a mu and sigma for which this exact function over here is obtained ?


α
0
Is it possible to represent this function as $\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}$?
YES
NO

Suppose we let ‘x’ got to infinity, then x-mu square for any fixed mu would go to infinity So if x with a – infinity would go to zero.

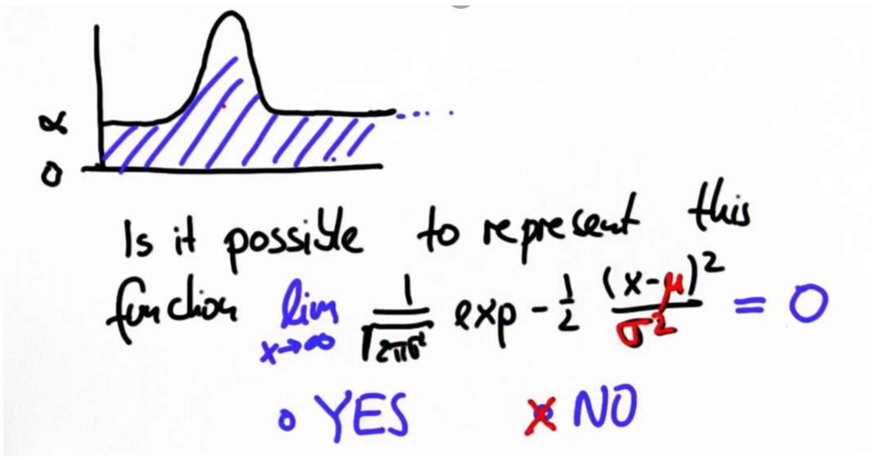
Because 1 / sqrt (2 * mu * sigma) is constant .

Therefore , we know that in the limit of x goes to infinity this expression must be zero

Limit x to infinity

However, in this graph it says at alpha and doesn`t go to zero So, therefore , there can`t be a valid Mu at sigma square.

If you deep in probability you know that the area in the Gaussians has to integrate into one and this area average it is actually infinite in size so it`s not even a valid execution.


α
0
Is it possible to represent this function $\lim_{x\to\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2} = 0$
YES
✗ NO

Quiz: how many dimension

Tracking problem that we talk about the class

In class we addressed a 1 – dimensional tracking problem where we estimated the location of the system and its velocity.

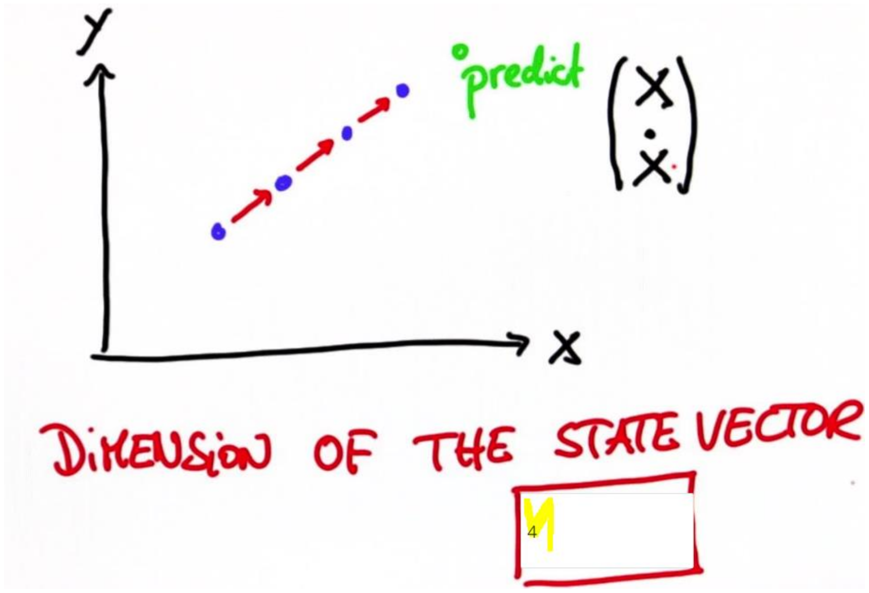
I`d like now to generalize this a 2-dim problem.

We will given coordinates x and y and the object we are tracking moves in 2 -dim space.

And we wish use a Kalman filter to understand where the object is what is the velocity is and even be able to predict a future location based on the estimate of the position and its velocity.

So only difference, class , is that our object now moves in a 2 dimensional space, where as in class it moved in a 1 – dimensional.

So my first question is what`s the dimension of the state vector in the Kalman filter


y
x
predict
$\begin{pmatrix} x \\ \dot{x} \end{pmatrix}$
DIMENSION OF THE STATE VECTOR
4

And the answer = 4

So rather than X and X node , we have X and Y and X node and Y node as our state vectors as our state vector and there`s 4 variables.

Quiz:

Now comes the tricky question, in the Kalman filter program study the 2-D Kalman filter, we had a matrix F.

And for delta t =0.1 and f matrix had a diagonal of 1 , which means is exportation our locations stays the same and our velocity stays the same.

د ة معناه ان اللوكيشن و السرعة فى نفس المكان علشان كدة كل واحد واخد وان

And we also knew that the velocity affected our state in the following way.

And you could now place 0.1 instead of the delta T to get specific F matrix.

Now I want to know from you for this 2-D case with a 4-dimensional state vector.

What is the new F ? it is a 4 by 4 so I want you to fill in all the above.

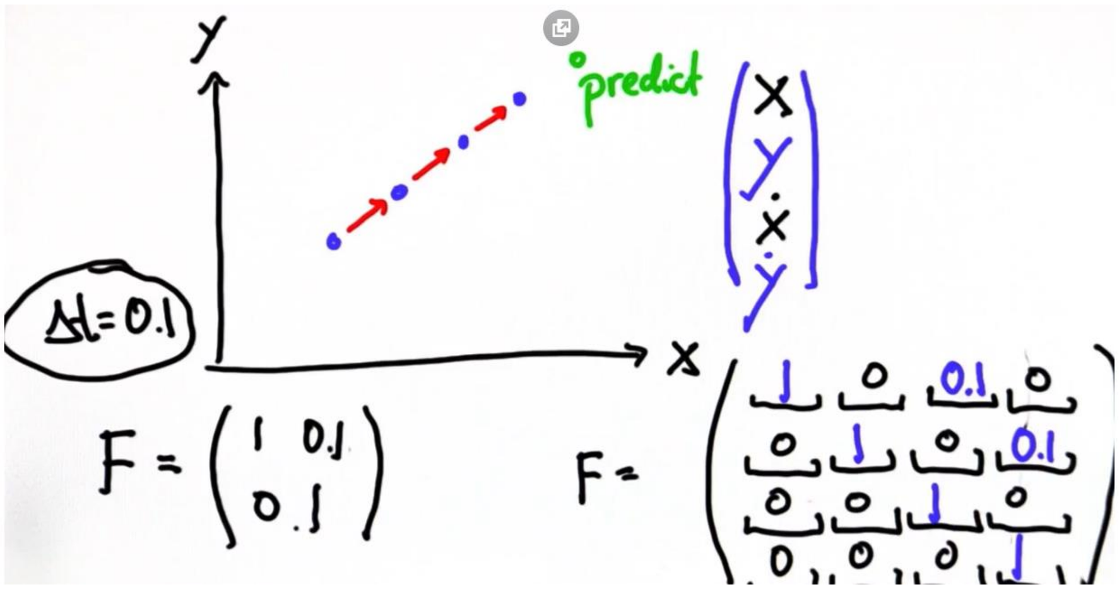
The main answer is the main diagonal remain one this express the effect that in absence of any velocity.

We expect the x-coordinate and the y-coordinate not to change.

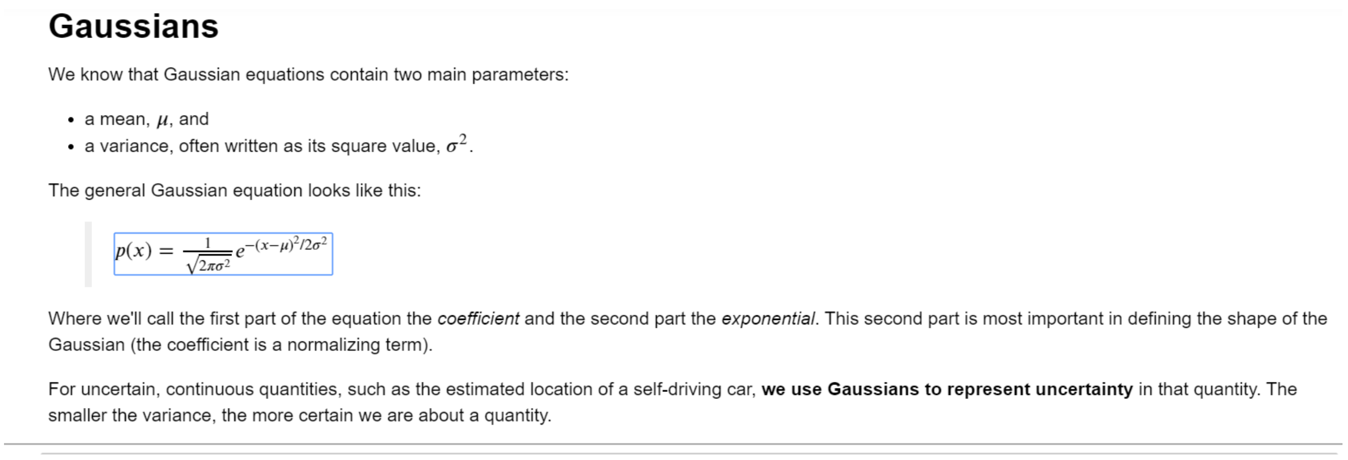
We also don`t expect the velocities to change but x node in 3 position through a 0.1 over here.

The third coordinate of velocity and the it affects the first coordinates.

And the all other values should be zeros so this is the 4-dimesnion translation of this 2 -dimensional state transition matrix over here.


y
x
predict
$\Delta t = 0.1$
$\begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix}$
$F = \begin{pmatrix} 1 & 0.1 \\ 0 & 1 \end{pmatrix}$
$F = \begin{pmatrix} 1 & 0 & 0.1 & 0 \\ 0 & 1 & 0 & 0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

1. *Gaussian Calculations*



Gaussians

We know that Gaussian equations contain two main parameters:

- a mean, μ, and
- a variance, often written as its square value, σ^2.

The general Gaussian equation looks like this:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Where we'll call the first part of the equation the *coefficient* and the second part the *exponential*. This second part is most important in defining the shape of the Gaussian (the coefficient is a normalizing term).

For uncertain, continuous quantities, such as the estimated location of a self-driving car, **we use Gaussians to represent uncertainty** in that quantity. The smaller the variance, the more certain we are about a quantity.