Filtrelemek

Filtreler dis dunyadaki bir aksiyon hakkinda elde edilen gurultulu sinyalleri, tersine cevirerek arka plandaki aksiyon hakkinda hesaplama yapabilmemizi saglar. Mesela Kalman Filtreleri (KF) icin gizlenmis konum bir robotun nerede oldugu, bir senetin fiyati gibi bir sey olabilir, gizli konum bilgisi x_t degiskeninde o konum hakkindaki gurultulu olcum y_t icindedir. Hem gizli konumlar arasindaki gecis, hem de olcumun gurultusu lineer bir fonksiyon uzerindendir.

$x_{t+1} = Ax_t + v$

$y_t = Hx_t + w$

v ve w'in dagilimi Gaussian'dir ve kovaryans sirasiyla Q ve R icindedir.

Zaman faktorunu de dahil etmek gerekirse;

$\hat{x}_t^t = E[x_t|y_0, .., y_t]$

$P_t^t = E[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})'|y_0, ..., y_t]$

Filtremenin amaci x_{t+1} ve P_{t+1} hesabini yeni bir olcum y_{t+1} uzerinden yapmak olacak. "Gizli" x_t derken bunu kastediyorduk, bu deger bize verilmiyor, sadece xt ve x_{t+1} arasindaki gecisin nasil oldugunu biliyoruz, gurultunun nasil eklendigini biliyoruz, ama bunlarin bilsek bile elde bir suru bilinmeyen var. Filtrelemenin matematiksel numaralari sayesinde bunu hesaplayabiliyor olacagiz. Yani yapmamiz gereken "oku tersine cevirmek", yani x_t'nin y_t uzerindeki sartsal bagliligini (conditional dependence) ortaya cikartmak, bunu y_t'nin x_t'ye olan sartsal bagimliligini tersine cevirerek yapmak. Ana denklemin iki tarafinin da beklentisini (expectation) alalim:

$E\ x_{t+1} = \hat{x}_{t+1} = A\mu_t = A\hat{x}_t$

Simdi iki tarafin kovaryansini alalim ve P_t'yi $cov\ x(t)$ olarak belirtelim:

$P_{t+1} = AP_tA' + Q$

Bu gecis "zaman guncellemesi" olarak adlandirilir. Normal dagilimlari t anindan $t + 1$ anina gecirmemizi saglar. y iceren formullerde benzer bir durum var.

$\hat{x}_{t+1}^t = Ax_t^t$

$P_{t+1}^t = AP_t^t A' + Q$

$y_{t+1} = Cx_{t+1} + w_t$

$E[y_{t+1}|y_0, .., y_t] = E[Cx_{t+1} + w_t|y_0, .., y_t]$

$\hat{y}_{t+1}^t = C\hat{x}_{t+1}$

Kovaryans icin benzer durum

$E[(y_{t+1} - \hat{y}_{t+1}^t)(y_{t+1} - \hat{y}_{t+1}^t)'|y_0, ..., y_t] = C_{t+1}^t C' + R$

Simdi daha zor is olan oku tersini cevirmeye gelelim. Eger amacimiz p(xt —yt) denklemini elde etmek ise o zaman bu iki degiskeni iceren birlesik dagilimi (joint distribution) elde etmek zorundayiz. Iki Gaussian'in birlesiminin yeni bir Gaussian oldugunu biliyoruz, o zaman hem x_t hem de y_t'in kendisi cok boyutlu birer Gaussian olduklari icin onlarin birlesimi $p(x_t|y_t)$'in hakikaten devasa bir Gaussian olacagini tahmin edebiliriz.

x_t ve y_t'in birlesimi olan Gaussian'i bulmak demek, bu Gaussian'in ortalamasini (mean) ve kovaryansini bulmak demektir cunku bir Gaussian ortalama ve kovaryansi ile net bir sekilde tanimlanabilir bir seydir. Bir numara yapalim, ve $y_t = Cx_t + w_t$'yi $z = Hu$ seklinde yazalim. Sonra

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix}, H = \begin{bmatrix} I & 0 \\ C & I \end{bmatrix}, u = \begin{bmatrix} x_t \\ w_t \end{bmatrix}$$

Boylece daha basit bir denklemin kovaryansini alabiliriz

$cov(z) = H\ cos(u)H'$

$$cov(u) = \begin{bmatrix} P_t & 0 \\ 0 & R \end{bmatrix}$$

Tam carpim suna esit

$$\begin{bmatrix} I & 0 \\ C & I \end{bmatrix} \begin{bmatrix} P_t & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} I & C' \\ 0 & I \end{bmatrix}$$

bunun sonucu ise

$$\begin{bmatrix} P_t & P_t C' \\ CP_t & CP_t C' + R \end{bmatrix}$$

Bunu baglantisal denklem icin ve ortalamayi icerecek sekilde yazabiliriz

$$\left[\begin{array}{c} \hat{x}_t^t \\ C\hat{x}_t^t \end{array} \right], \left[\begin{array}{cc} P_t^t & P_t^t C' \\ CP_t^t & CP_t^t + R \end{array} \right]$$

Ayni sekilde x_{t+1}, y_{t+1} birlesik dagilim icin

$$\left[\begin{array}{c} \hat{x}_{t+1}^t \\ C\hat{x}_{t+1}^t \end{array} \right], \left[\begin{array}{cc} P_{t+1}^t & P_{t+1}^t C' \\ CP_{t+1}^t & CP_{t+1}^t C' + R \end{array} \right] \tag{1}$$

Simdi x_{t+1}^{t+1} 'in ortalama ve varyansi icin parcali Gaussian kavramini anlatmaliyiz. Bir n boyutlu Gaussian daha kucuk boyutlardaki p ve q alt Gaussian'lara parcalanabilir (tabii ki $n = p + q$). Yani su ifade kullanilabilir

$$\mu = \left[\begin{array}{c} \mu_1 \\ \mu_2 \end{array} \right], \Sigma = \left[\begin{array}{cc} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{array} \right] \tag{2}$$

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{(p+q)/2}|\Sigma|^{1/2}} exp\Big\{ -\frac{1}{2} \left(\begin{array}{c} x_1 - \mu_1 \\ x_2 - \mu_2 \end{array} \right)' \left[\begin{array}{cc} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{array} \right]^{-1} \left(\begin{array}{c} x_1 - \mu_1 \\ x_2 - \mu_2 \end{array} \right) \Big\}$$

Uzun cebirsel islemlerden sonra $p(x_1|x_2)$ ifadesini elde ederiz. Buradan sartlanmis (conditioned) μ ve Σ alinir.

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \tag{3}$$

$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

Simdi denklem 3'u alip 1'in icine koydugumuzda ve 2'deki yerlesim yapisini dikkate aldigimizda $\hat{x}_{t+1}^{t+1}$ ve P_{t+1}^{t+1} formullerini ortaya cikartabiliriz.

$$\hat{x}_{t+1}^{t+1} = x_{t+1}^t + P_{t+1}^t C'(CP_{t+1}^t C' + \Sigma_w)^{-1}(y_{t+1} - C\hat{x}_{t+1}^t)$$

$$P_{t+1}^{t+1} = P_{t+1}^t - P_{t+1}^t C'(CP_{t+1}^t C' + R)^{-1} CP_{t+1}^t$$

Eger $K = P_{t+1}^t C'(CP_{t+1}^t C' + \Sigma_w)^{-1}$ dersek

$$\hat{x}_{t+1}^{t+1} = \hat{x}_{t+1}^t + K_t(y_{t+1} - C\hat{x}_{t+1}^t$$

$$P_{t+1}^{t+1} = P_{t+1}^t - K_t CP_{t+1}^t$$

Ornek: Veriye Duz Cizgi Uydurmak (Line Fitting)

Eger elimizde bir cizgiye uydurmak icin kullanacagimiz tum veri olsaydi, uydurma islemi icin en az kareler (least sqaures) yontemini kullanabilirdik. Kalman Filtreleri bize yeni veri geldigi anda, her seferinde, azar azar bir

cizgiyi uydurmamizi sagliyor. Hatta matematiksel olarak ispalanmistir ki eger baslangic noktasi ayniysa, azar azar veriyi KF ile almanin sonunda, tum veriyi bir kerede en az karesel yontem ile uydurmak ayni sonucu verir.

Peki bu uydurma islemini nasil yapariz? Burada veriyi nasil temsil ettigimiz konusunda ufak bir numara kullanmamiz lazim.

Kendimize bir soru soralim: bu sistemin konum bilgisi nedir? Bir robotu izliyorsak mesela soru cevabi basittir, onun x, y gibi kordinat bilgisi. Duz cizgi fit ederken takip edilen bunlar degil, bize gerekli olan bir cizginin "egimi (slope)". Yani hem bir cizginin y eksenini kestigi nokta, hem de cizginin egimi xt konum bilgisi icinde dahil edilecek. Burada KF literaturunden gelen x, y harfleri birbirine karismasin diye cizginin degerlerini xx_t ve yy_t olarak tanimlayacagiz. O zaman x_t vektoru suna benzer:

$$x_t = \begin{bmatrix} yy_t \\ a \end{bmatrix}$$

ki burada a harfi egimi temsil etmektedir. a bir sabit olduguna gore KF her zaman diliminde ayni kalacak bir degiskeni hesaplayacaktir. Cogunlukla KF ile her zaman diliminde degisik olan degerlerin hesaplandigini goruruz, bu uygulamaya gore degisen bir seydir, matematiksel bir mecburiyet degildir. A matrisimiz ile de biraz numara yapmamiz gerekli. Bu matris x_t'yi donusturup x_{t+1}'i elde etmemizi saglayan sey olduguna gore A'nin soyle olmasi gerekir:

$$A_t = \begin{bmatrix} 1 & \Delta xx \\ 0 & 1 \end{bmatrix}$$

Bu matrisi x_t ile carptigimizda $yy_t \cdot 1 + a \cdot \Delta xx$ degerini elde ediyoruz, ki bu deger bir cizgi uzerinde bir sonraki noktayi temsil ediyor. Dis olcumu veren gurultu matrisi H ise

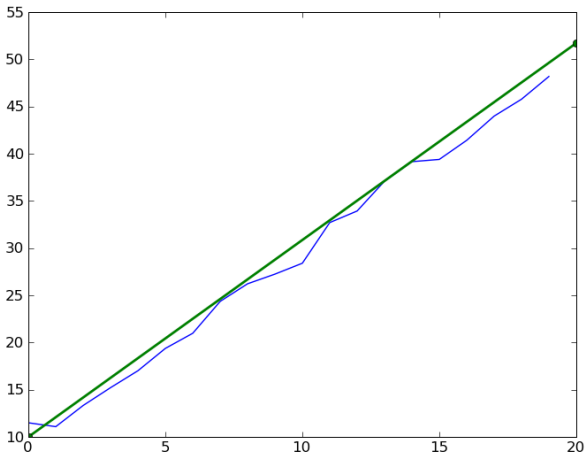
$$H = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

seklinde. Bunu x_t ile carptigimizda y_t'yi (iki kere) elde ettigimizi gorecegiz. Not: Niye iki kere? Kodlama sirasinda boyutlarin uyumlu olmasi icin boyle gerekti, cok buyuk bir rahatsizlik degil. Kod altta gorulebilir.

```
from pylab import *
from numpy import *
```

```
slope = 2

#
# x_{t+1} = A x_t + Q
# y_t = Hx_t + R
#
def Kalman(obs,x,mu_init,nsteps):

    ndim = shape(mu_init)[0]

    Q = zeros((ndim, ndim))
    A = eye(ndim)
    H = array([[1, 0], [1, 0]])

    mu_hat = mu_init
    cov = ones((ndim, ndim))
    R = eye(ndim) * 10

    m = zeros((ndim,nsteps),dtype=float)
    ce = zeros((ndim,ndim,nsteps),dtype=float)

    for t in range(1,nsteps):
        # Make prediction
        # A is transformation matrix, equals to
        # TR: Tahmini yap
        # A transofmrasyon matrisi ve suna esit
        # | 1 delta_x |
        # | 0     1   |
        A = array([[1, x[t]-x[t-1]], [0, 1]])
        mu_hat_est = dot(A,mu_hat)
        cov_est = dot(A,dot(cov,transpose(A))) + Q

        # Update estimate
        # TR: tahmini guncelle
        error_mu = obs[:,t] - dot(H,mu_hat_est)
        error_cov = dot(H,dot(cov,transpose(H))) + R
        K = dot(dot(cov_est,transpose(H)),linalg.inv(error_cov))
        mu_hat = mu_hat_est + dot(K,error_mu)
```

```
            m[:,t] = mu_hat
            cov = dot((eye(ndim) - dot(K,H)),cov_est)
            ce[:,:,t] = cov
            print "mu_hat="+str(mu_hat)
        return mu_hat

N = 20

#
# create sample data
# TR: ornek veri yarat
#

obs = zeros((2, N))
x = xrange(N)
for i in xrange(N):
    obs[0, i] = obs[1, i]  = (slope*i)+random.normal(10)

print "obs="+str(obs.shape)

mu_hat = Kalman(obs, x, mu_init=array([0, 0]),nsteps=N)

plot(obs[0, :])
plot([0,N], [10,N*mu_hat[1]], 'go-', label='line 1', linewidth=2)
show()
```



Ornek: Obje Takibi

Daha degisik bir ornekten bahsedelim. Bu ornekte OpenCV kutuphanesinden elde ettigimiz 2 boyutlu degerleri y_t icin kullanacagiz. Degerler OpenCV'nin bir satranc tahtasi seklinin kose noktalarini otomatik olarak bulabilen cvFindChessboardCorners cagrisinden gelecek (ayrica cvDrawChessboardCorners ile bu noktalari ekranda aninda gosterebilecegiz).

Elimizdeki "gurultulu" olcumler iki boyutlu noktasal degerler. Gurultulu cunku kamera bize bu imajlari aktarirken hata eklemis olabilir, OpenCV fonksiyonu hesabi yaparken hata eklemis olabilir, bir suru olasilik var.

Bu ornekte, ayrica, ilk kez KF ortaminda boyut degisikligi olasiligini net bir sekilde gorebiliyoruz. Gizli konum bilgisi x_t 3 boyutlu bir nokta, ama elimizdeki olcum 2 boyutlu bir "yansima". Yansima sirasinda kacinilmaz olarak deger kaybediliyor, bir boyutun bilgisi ortadan yokoluyor. Ama tum bu bilinmezlere ragmen Kalman filtresinin bizim icin gizli bilgiyi hesaplamasini istiyoruz.

Bu problemde A matrisi ne olacaktir? Obje takibi konularinda A'nin ne oldugunu hayal etmek daha kolay, A matrisi iki zaman dilimi arasindaki "hareketi" temsil edecek. Bu problemdeki ek bir kolaylik bu hareketi onceden bildigimiz, ve hareketin tek yonde oldugu. Yani resimde benim tuttugum kartonu ne kadar hizla hareket ettirdigimi ben onceden probleme bildiriyo-

rum. Yer degisikligini d olarak betimledim, ve A soyle oldu:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dikkat edersek A 4x4 boyutunda, 3x3 degil. 3 boyutlu kordinatlari temsil etmek icin homojen kordinat sistemini kullandigimiz icin boyle oldu, o sebeple zaten x_t de 4x1 oldu, ona uymak icin A'nin degismesi gerekiyordu. Ax_t carpiminin hakikaten kartonu hareket ettirdigini gostermek icin bu carpimi bir ornek uzerinde yapalim: Diyelim ki $x_t = [a_1\ a_2\ a_3\ a_4]$ o zaman Ax_t ya da x_{t+1} su hale gelir: $[a_1\ a_2\ a_3 + d\ a_4]$.

Bakiyoruz, hakikaten de d kadarlik bir yer degisimi z kordinati, yani derinlik uzerinde eklenmis. Test amaclarimiz icin d = -0.5 aldik, yani satranc tahta kartonunun her zaman diliminde kameraya dogru 0.5 cm ilerledigini belirttik. Tabii bu da kabaca bir tahmindi (her ne kadar hareketi yaptiran ben olsam bile!), ama filrelemenin gucunu burada goruyoruz. Benim tahminimde "gurultu" yani "hata payi" var, olcumde gurultu var, tum bunlar ust uste konsa bile filtre yine de gizli konumu bulacak.

Olcumsel donusumu temsil eden H'e ben onun temeli olan yansima (projection) kelimesinden gelen P matrisinden bahsedelim. Yansima matrisi goruntu (vision) literaturunde tek delikli kamera (pinhole camera) modelinden ileri gelen bir matristir ve bu matrisi hesaplamak ayarlama / kalibrasyon (calibration) denen apayri bir islemin parcasidir. OpenCV icinde kalibrasyon icin fonksiyonlar var, biz de bunlari denedik, kalibrasyon icin kullandigimiz resimlerle alakali olmali, elde edilen sonuclardan memnun kalmadik. Alternatif olarak sunu yaptik; resimde gorulen yesil yuzey bizim programin olusturdugu hayali bir yuzey. Filtrenin o anki tahminini P uzerinden goruntuye yansitarak bu yuzeyi olusturduk, boylece deneme / yanilma yontemiyle pek cok P degerini deneyerek, yuzeyin resimde gorulen masanin sonunda cikacak sekilde olmasini sagladik. O noktaya gelince istedigimiz P degerini bulmus oluyorduk. Yansitma matrisleri 3x3 olur, KF buna bir dorduncu [0 0 0] satiri