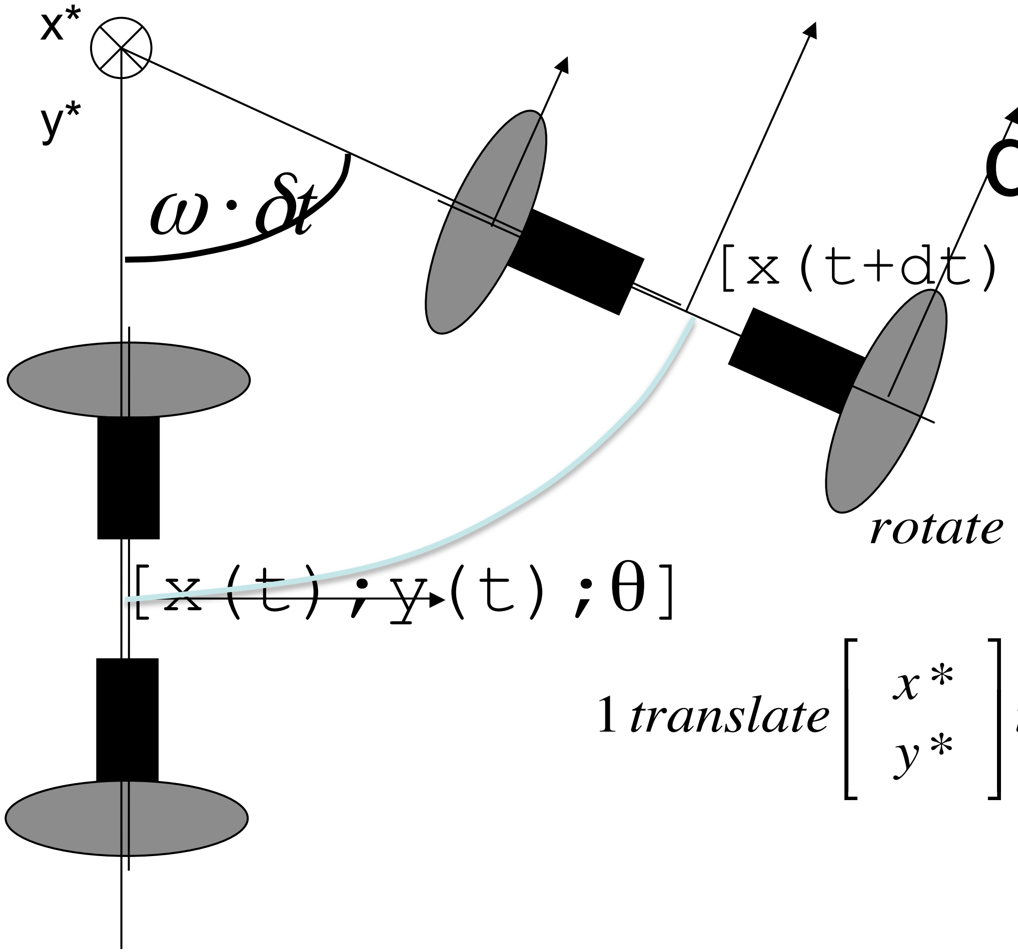
Motion model by Maximum Likelihood

Paul G. Plöger
SEE

Overview

- Introduce the motion model
  - inspect Thrun Algorithm 5.1 closely
- From the experiments to 3 Gaussians with different sigmas
- ML parameter estimation for Gaussians
- Find final $\alpha_1, \alpha_2, \ldots, \alpha_6$ via least mean squares

Geometry of diff.drive on circle



$$\textit{position of robot at time } t+\delta t\,?$$

$$\textit{rotate arround} \begin{bmatrix} x^* \\ y^* \end{bmatrix} \textit{by } \omega\,\delta t \textit{ in 3 steps:}$$

$$1 \textit{ translate} \begin{bmatrix} x^* \\ y^* \end{bmatrix} \textit{to origin, 2 rotate, 3 translate back}$$

$$t \longrightarrow t+\delta t$$

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{pmatrix} x-\lambda\sin(\theta) \\ y+\lambda\cos(\theta) \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} \cos(\omega\,\delta t) & -\sin(\omega\,\delta t) & 0 \\ \sin(\omega\,\delta t) & \cos(\omega\,\delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x-x^* \\ y-y^* \\ \theta \end{pmatrix} + \begin{pmatrix} x^* \\ y^* \\ \omega\,\delta t \end{pmatrix}$$

Velocity motion Model according to Thrun

Algorithm motion_model_velocity(x_t, u_t, x_{t-1})**:**

$$\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$

$$x^* = \frac{x + x'}{2} + \mu(y - y')$$

$$y^* = \frac{y + y'}{2} + \mu(x' - x)$$

$$r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$$

$$\Delta\theta = \operatorname{atan2}(y' - y^*, x' - x^*) - \operatorname{atan2}(y - y^*, x - x^*)$$

$$\hat{v} = \frac{\Delta\theta}{\Delta t}\, r^*$$

$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$

$$\boxed{\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}}$$

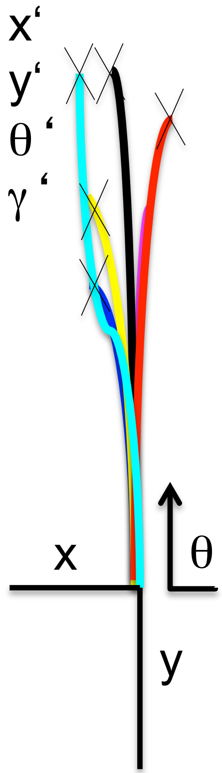
return $\mathbf{prob}(v - \hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \mathbf{prob}(\omega - \hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot \mathbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

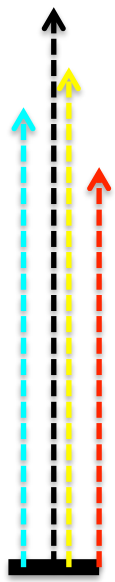
This is IMHO WRONG since this Gamma^ == 0 Always !!!

Table 5.1 Algorithm for computing $p(x_t \mid u_t, x_{t-1})$ based on velocity information. Here we assume x_{t-1} is represented by the vector $(x \;\; y \;\; \theta)^T$; x_t is represented by $(x' \;\; y' \;\; \theta')^T$; and u_t is represented by the velocity vector $(v \;\; \omega)^T$. The function **prob**(a, b) computes the probability of its argument a under a zero-centered distribution with standard deviation b. It may be implemented using any of the algorithms in Table 5.2.

case:run straight

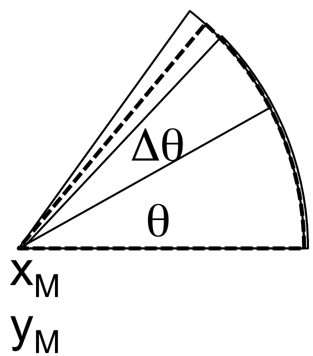
Observed Pathways

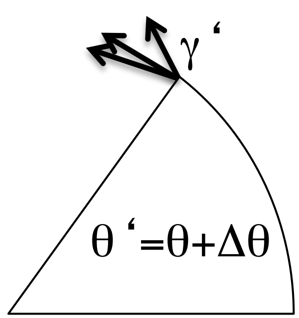




observed path lengths: distance using differential drive motion model:
Expected value: v
but observe:

$$\hat{v} = \hat{r}\,\frac{\Delta\hat{\theta}}{\Delta t}$$





case:run curves

$let\begin{pmatrix} x & y & \theta \end{pmatrix}^T be\ the\ start\ point$

$\begin{pmatrix} x' & y' & \theta' & \gamma' \end{pmatrix}^T be\ the\ observed\ end\ point$

$where\ \gamma'\ denotes\ heading\ angle\ and$

$\begin{pmatrix} x_M & y_M \end{pmatrix}^T is\ midpoint\ of\ circle\ of\ motion.$

$[N.B.: \gamma\ is\ different\ from\ expected\ tangential\ angle].$

$Then\ the\ distored\ speed\ is\ given\ by:$

$$\hat{u}_t = \begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} \frac{\hat{r}\cdot\Delta\hat{\theta}}{\Delta t} \\ \frac{\Delta\hat{\theta}}{\Delta t} \end{pmatrix}$$

$$\hat{r} = \sqrt{(x_M - x')^2 + (y_M - y')^2}$$

$$\Delta\hat{\theta} = \operatorname{atan2}(y' - y_M, x' - x_M) - \operatorname{atan2}(y - y_M, x - x_M)$$

$$dist = \hat{r}\cdot\Delta\hat{\theta}$$

$$\Delta\hat{\gamma} = \gamma' - \operatorname{atan2}(x' - x_M, y_M - y')$$

L6: Parameter estimation

- Introduction
- Parameter estimation
- Maximum likelihood
- Bayesian estimation
- Numerical examples

In previous lectures we showed how to build classifiers when the underlying densities are known

- Bayesian Decision Theory introduced the general formulation
- Quadratic classifiers covered the special case of unimodal Gaussian data

In most situations, however, the true distributions are unknown and must be estimated from data

- Two approaches are commonplace
  - Parameter Estimation (this lecture)
  - Non-parametric Density Estimation (the next two lectures)

Parameter estimation

- Assume a particular form for the density (e.g. Gaussian), so only the parameters (e.g., mean and variance) need to be estimated
  - Maximum Likelihood
  - Bayesian Estimation

Non-parametric density estimation

- Assume NO knowledge about the density
  - Kernel Density Estimation
  - Nearest Neighbor Rule

ML vs. Bayesian parameter estimation

Maximum Likelihood

- The parameters are assumed to be FIXED but unknown
- The ML solution seeks the solution that "best" explains the dataset X

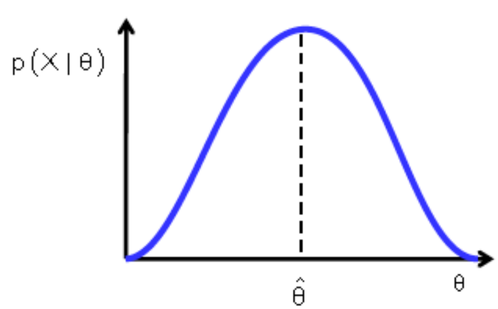
$$\hat{\theta} = argmax[p(X|\theta)]$$

Bayesian estimation

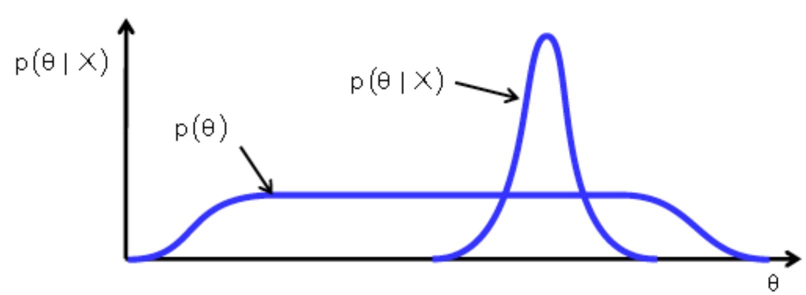
- Parameters are assumed to be random variables with some (assumed) known a priori distribution
- Bayesian methods seeks to estimate the posterior density $p(\theta|X)$
- The final density $p(x|X)$ is obtained by integrating out the parameters

$$p(x|X) = \int p(x|\theta)p(\theta|X)d\theta$$

Maximum Likelihood



Bayesian



Maximum Likelihood

Problem definition

- Assume we seek to estimate a density $p(x)$ that is known to depends on a number of parameters $\theta = [\theta_1, \theta_2, \dots \theta_M]^T$
  - For a Gaussian pdf, $\theta_1 = \mu$, $\theta_2 = \sigma$ and $p(x) = N(\mu, \sigma)$
  - To make the dependence explicit, we write $p(x|\theta)$
- Assume we have dataset $X = \{x^{(1}, x^{(2}, \dots x^{(N}\}$ drawn independently from the distribution $p(x|\theta)$ (an i.i.d. set)
  - Then we can write

$$p(X|\theta) = \Pi_{k=1}^{N} p\left(x^{(k} | \theta\right)$$

  - The ML estimate of θ is the value that maximizes the likelihood $p(X|\theta)$

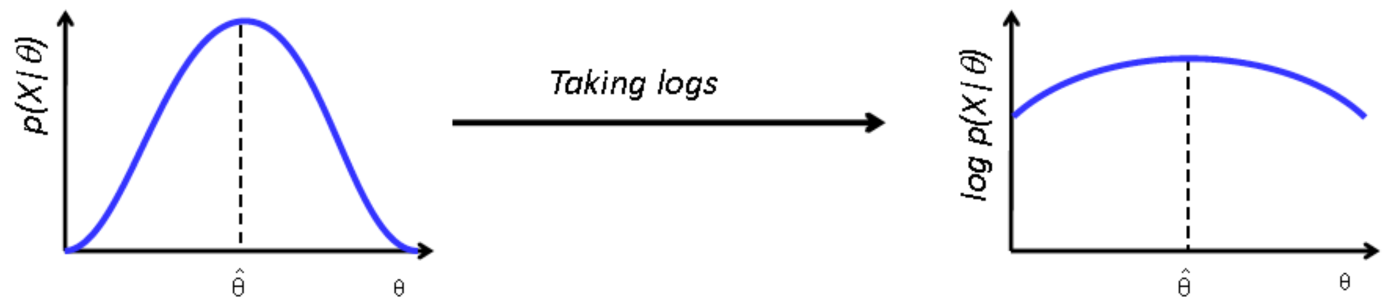
$$\hat{\theta} = argmax[p(X|\theta)]$$

  - This corresponds to the intuitive idea of choosing the value of θ that is most likely to give rise to the data

For convenience, we will work with the log likelihood

- Because the log is a monotonic function, then:

$$\hat{\theta} = argmax[p(X|\theta)] = argmax[\log p(X|\theta)]$$



- Hence, the ML estimate of θ can be written as:

$$\hat{\theta} = argmax\left[\log \Pi_{k=1}^{N} p\left(x^{(k}|\theta\right)\right] = argmax\left[\Sigma_{k=1}^{N} \log p\left(x^{(k}|\theta\right)\right]$$

  - This simplifies the problem, since now we have to maximize a sum of terms rather than a long product of terms
  - An added advantage of taking logs will become very clear when the distribution is Gaussian

Example: Gaussian case, μ unknown

Problem statement

- Assume a dataset $X = \{x^{(1}, x^{(2}, \ldots x^{(N}\}$ and a density of the form $p(x) = N(\mu, \sigma)$ where σ is known
- What is the ML estimate of the mean?

$$\theta = \mu \Rightarrow \hat{\theta} = \arg max \Sigma_{k=1}^{N} log p\left(x^{(k} | \theta\right) =$$

$$= \arg max \Sigma_{k=1}^{N} log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}\left(x^{(k} - \mu\right)^2\right)\right) =$$

$$= \arg max \Sigma_{k=1}^{N}\left[log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2\sigma^2}\left(x^{(k} - \mu\right)^2\right]$$

- The maxima of a function are defined by the zeros of its derivative

$$\frac{\partial \Sigma_{k=1}^{N} log p\left(x^{(k} | \theta\right)}{\partial \theta} = \frac{\partial}{\partial \theta} \Sigma_{k=1}^{N} log p(\cdot) = 0 \Rightarrow$$

$$\mu = \frac{1}{N} \Sigma_{k=1}^{N} x^{(k}$$

- So the ML estimate of the mean is the average value of the training data, a very intuitive result!

Example: Gaussian case, both μ and σ unknown

A more general case when neither μ nor σ is known

- Fortunately, the problem can be solved in the same fashion
- The derivative becomes a gradient since we have two variables

$$\hat{\theta} = \begin{bmatrix} \theta_1 = \mu \\ \theta_2 = \sigma^2 \end{bmatrix} \Rightarrow \nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \Sigma_{k=1}^{N} logp\left(x^{(k}|\theta\right) \\ \frac{\partial}{\partial \theta_2} \Sigma_{k=1}^{N} logp\left(x^{(k}|\theta\right) \end{bmatrix} = \Sigma_{k=1}^{N} \begin{bmatrix} \frac{1}{\theta_2}\left(x^{(k} - \theta_1\right) \\ -\frac{1}{2\theta_2} + \frac{\left(x^{(k} - \theta_1\right)^2}{2\theta_2^2} \end{bmatrix} = 0$$

- Solving for θ_1 and θ_2 yields

$$\hat{\theta}_1 = \frac{1}{N}\Sigma_{k=1}^{N} x^{(k}; \quad \hat{\theta}_2 = \frac{1}{N}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\theta}_1\right)^2$$

  - Therefore, the ML of the variance is the sample variance of the dataset, again a very pleasing result

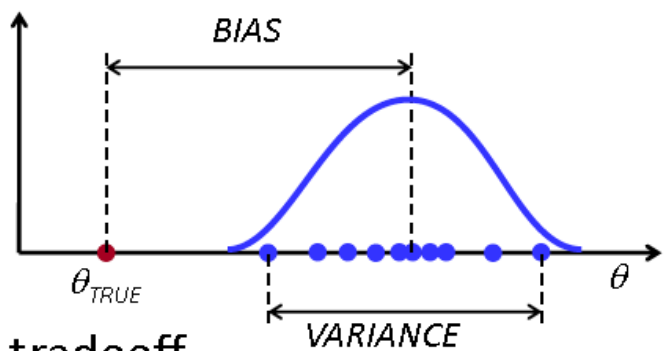
- Similarly, it can be shown that the ML estimates for the multivariate Gaussian are the sample mean vector and sample covariance matrix

$$\hat{\mu} = \frac{1}{N}\Sigma_{k=1}^{N} x^{(k}; \quad \hat{\Sigma} = \frac{1}{N}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\mu}\right)\left(x^{(k} - \hat{\mu}\right)^T$$

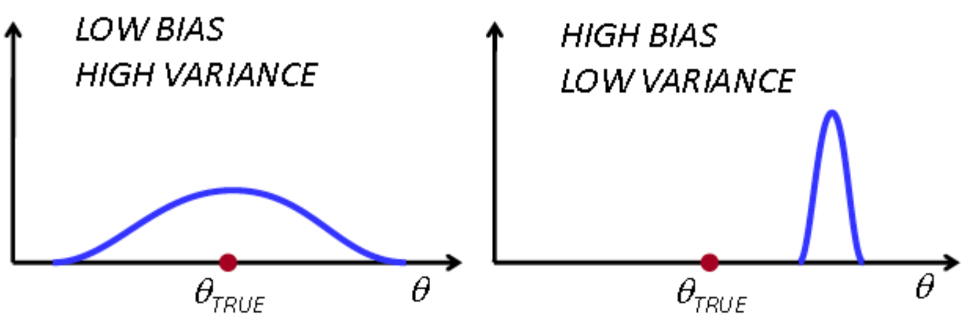
Bias and variance

How good are these estimates?

- Two measures of “goodness” are used for statistical estimates
- **BIAS:** how close is the estimate to the true value?
- **VARIANCE:** how much does it change for different datasets?



- The bias-variance tradeoff
  - In most cases, you can only decrease one of them at the expense of the other



What is the bias of the ML estimate of the mean?

$$E[\hat{\mu}] = E\left[\frac{1}{N}\Sigma_{k=1}^{N} x^{(k}\right] = \frac{1}{N}\Sigma_{k=1}^{N} E\left[x^{(k}\right] = \mu$$

- Therefore the mean is an unbiased estimate

What is the bias of the ML estimate of the variance?

$$E[\hat{\sigma}^2] = E\left[\frac{1}{N}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\mu}\right)^2\right] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$

- Thus, the ML estimate of variance is BIASED
  - This is because the ML estimate of variance uses $\hat{\mu}$ instead of μ
- How "bad" is this bias?
  - For $N \to \infty$ the bias becomes zero asymptotically
  - The bias is only noticeable when we have very few samples, in which case we should not be doing statistics in the first place!
- Notice that MATLAB uses an unbiased estimate of the covariance

$$\hat{\Sigma}_{UNBIAS} = \frac{1}{N-1}\Sigma_{k=1}^{N}\left(x^{(k} - \hat{\mu}\right)\left(x^{(k} - \hat{\mu}\right)^T$$

Recipe

1) Fix $v = v_{0/1/2}$ and $\omega_0 = 0$, rehearse 10 (each) times => use ML to get $\sigma_{1/2/3,0}$
2) Fix v_1 and vary $\omega_{1/2}$ (left and right!) thus getting $\sigma_{1,1/2/-1/-1}$
3) Least square fit all observations to find best $\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6$ by:

$$v: experiment: \min_{\alpha_1,\alpha_2} \sum_{i,j} \left({}^{v}\sigma_{ij} - (\alpha_1 v_i + \alpha_2 \omega_j)\right)^2$$

$$\omega: experiment: \min_{\alpha_3,\alpha_4} \sum_{i,j} \left({}^{\omega}\sigma_{ij} - (\alpha_3 v_i + \alpha_4 \omega_j)\right)^2$$

$$\gamma: experiment: \min_{\alpha_5,\alpha_6} \sum_{i,j} \left({}^{\gamma}\sigma_{ij} - (\alpha_5 v_i + \alpha_6 \omega_j)\right)^2$$