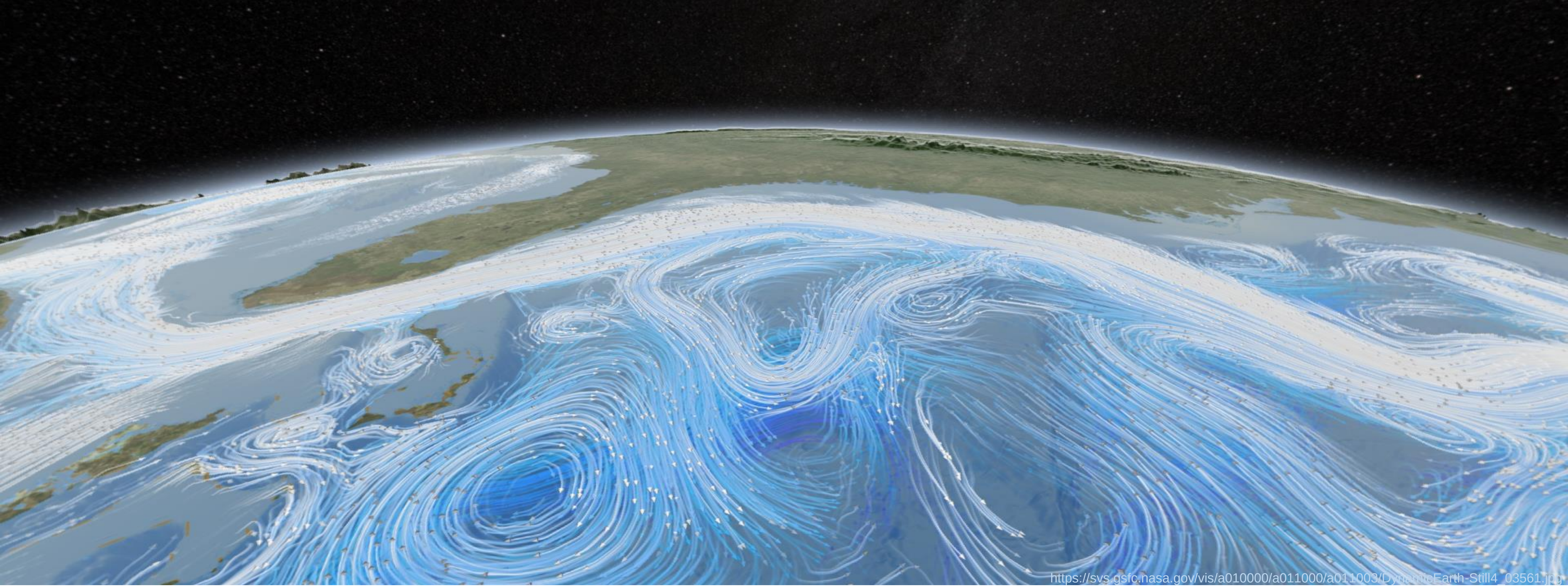


https://svs.gsfc.nasa.gov/vis/a010000/a011000/a011003/DynamicEarth-Still4_03561.jpg

Robotics for Future Industrial Applications

Learning Global and Local Dynamic Models

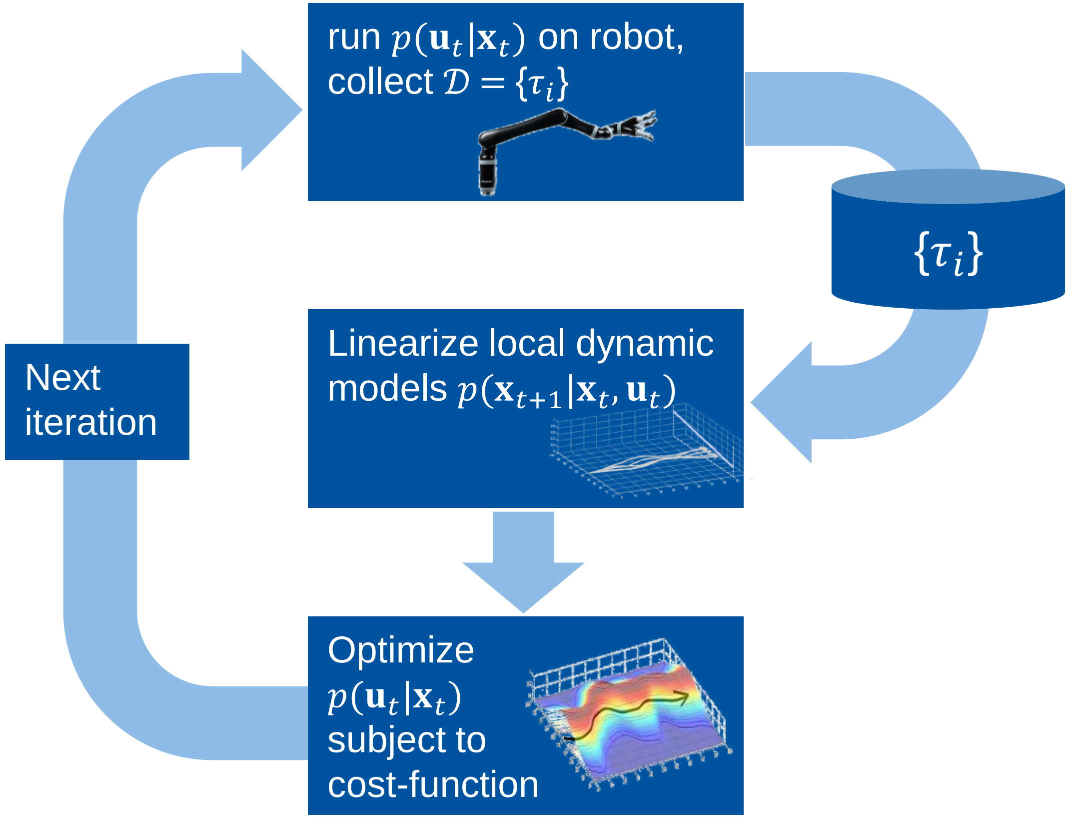
Philipp Ennen, M.Sc.

Learning a Policy

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$$

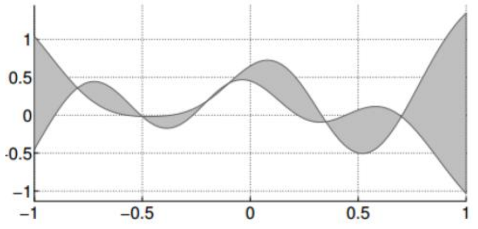
$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$

$$\mathbf{A}_t = \frac{\partial f}{\partial \mathbf{x}_t} \qquad \mathbf{B}_t = \frac{\partial f}{\partial \mathbf{u}_t}$$



What kind of models can we use?

Gaussian process



GP with input $(\mathbf{x}, \mathbf{u})$ and output $\mathbf{x}$'

Pro: very data-efficient

Con: not great with non-smooth dynamics

Con: very slow when dataset is big

Neural Network

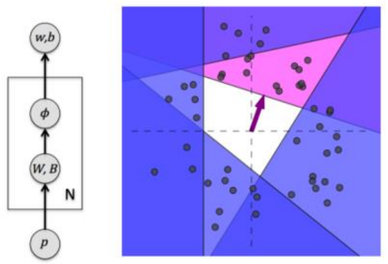


image: Punjani & Abbeel '14

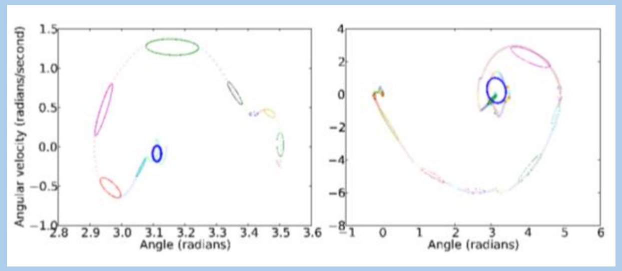
Input is $(\mathbf{x}, \mathbf{u})$, output ist $\mathbf{x}$'

Pro: very expressive, can use lots of data

Con: not so great I low data regimes

This weeks focus!

Gaussian Mixture Model

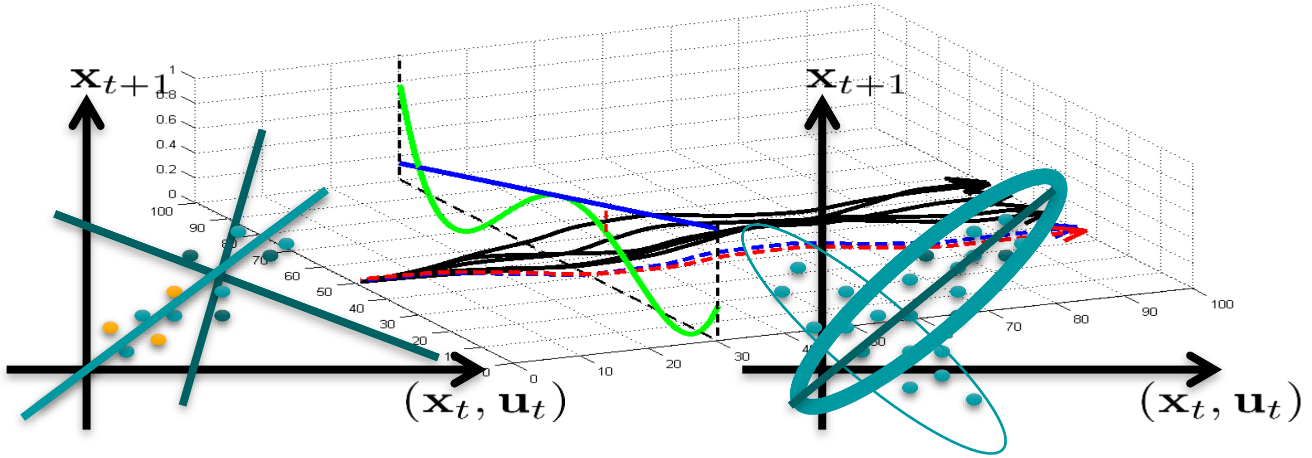


GMM over $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ tuples

Train on $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$, condition to get $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

For i'th mixture element, $p_i(\mathbf{x}, \mathbf{u})$ gives region where the mode $p_i(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ holds

Pro: very expressive, if the dynamics can be assumed as piecewise linear



1. Run time-varying policy $q(\mathbf{u}_t|\mathbf{x}_t)$ on robot N times
2. Collect dataset $\mathcal{D} = \{\tau_i\}$ where $\tau_i = \{\mathbf{x}_{1i}, \mathbf{u}_{1i}, \ldots, \mathbf{x}_{Ti}, \mathbf{u}_{Ti}\}$
3. For each $t \in \{0, \ldots, T-1\}$, fit linear Gaussian $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$
4. Solve control problem to get new $q(\mathbf{u}_t|\mathbf{x}_t)$

Content

I. **Gaussians**
   I. Univariate Gaussian
   II. Multivariate Gaussian
   III. Conditioning (Bayes' rule)

II. **Gaussian Mixture Model**

III. **Learning Local Dynamic Models**

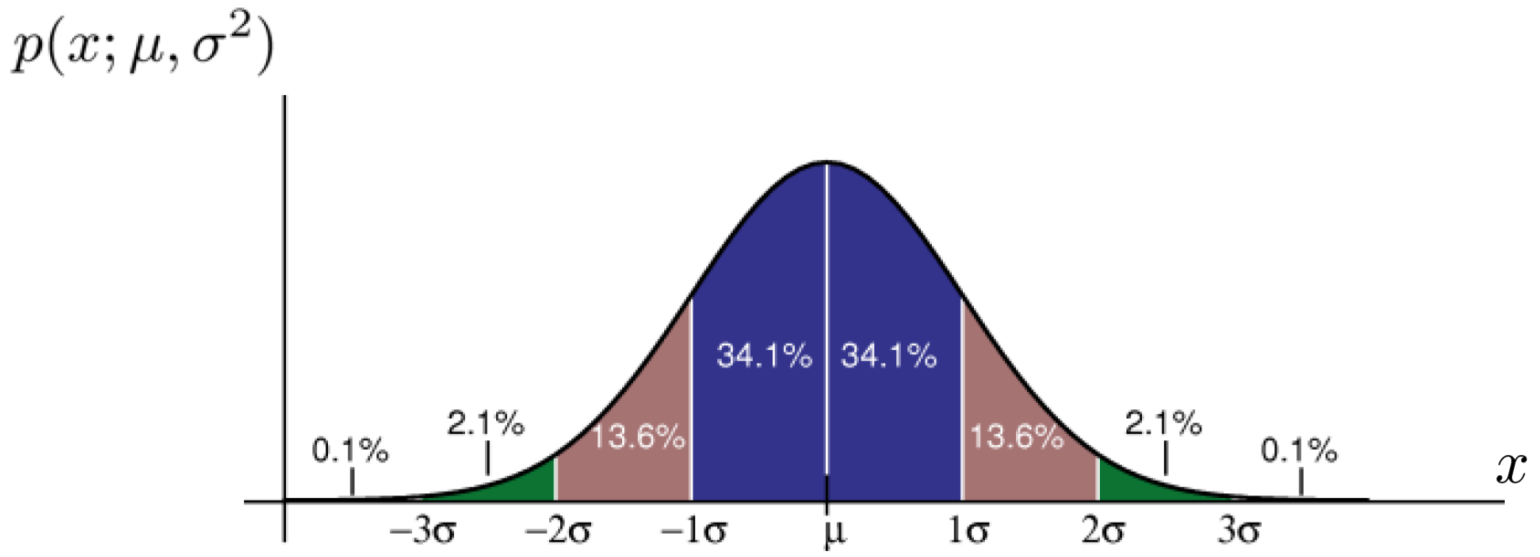
Disclaimer: lots of linear algebra now. In fact, pretty much all computations with Gaussians will be reduced to linear algebra!

Univariate Gaussian

- Gaussian distribution with mean μ, and standard deviation σ:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

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



$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \text{exp}(-\frac{(x-\mu)^2}{2\sigma^2})$$

Properties of Gaussians

- Densities integrate to one:

$$\int_{-\infty}^{\infty} p(x; \mu, \sigma^2) dx = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \text{exp}(-\frac{(x-\mu)^2}{2\sigma^2}) dx = 1$$

- Mean:

$$\begin{aligned} \text{E}_X[X] &= \int_{-\infty}^{\infty} x p(x; \mu, \sigma^2) dx \\ &= \int_{-\infty}^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} \text{exp}(-\frac{(x-\mu)^2}{2\sigma^2}) dx \\ &= \mu \end{aligned}$$

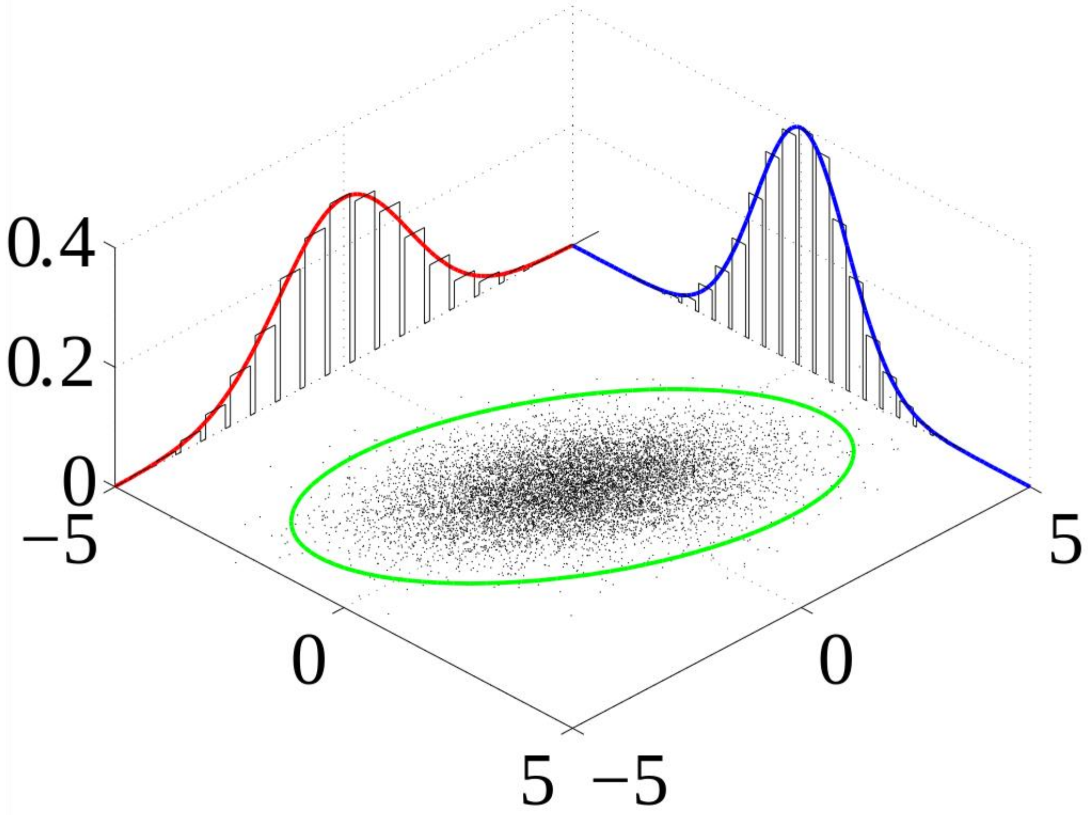
- Variance:

$$\begin{aligned} \text{E}_X[(X-\mu)^2] &= \int_{-\infty}^{\infty} (x-\mu)^2 p(x; \mu, \sigma^2) dx \\ &= \int_{-\infty}^{\infty} (x-\mu)^2 \frac{1}{\sigma\sqrt{2\pi}} \text{exp}(-\frac{(x-\mu)^2}{2\sigma^2}) dx \\ &= \sigma^2 \end{aligned}$$

Multivariate Gaussian

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

$$\int \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) dx = 1$$



- Remember: For a matrix $A \in \mathbb{R}^{n\times n}$, $|A|$ denotes the determinant of A
- Remember: For a matrix $A \in \mathbb{R}^{n\times n}$, A^{-1} denotes the inverse of A
  - Rule: $A^{-1}A = I = AA^{-1}$
  - $I \in \mathbb{R}^{n\times n}$ is the identity matrix with all diagonal entries equal to one, and all off-diagonal entries equal to zero

Multivariate Gaussian

- Mean:

$$\mathsf{E}_X[X_i] = \int x_i p(x;\mu,\Sigma)dx = \mu_i$$

$$\mathsf{E}_X[X] = \int x p(x;\mu,\Sigma)dx = \mu$$

(integral of vector = vector of integrals of each entry)

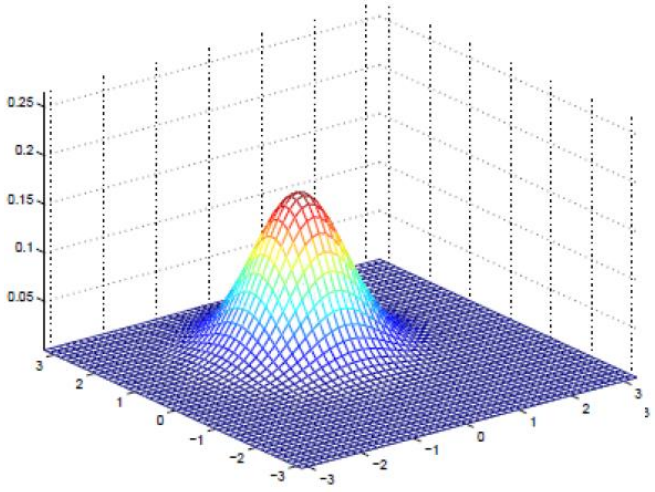
- Covariance:

$$\mathsf{E}_X[(X_i-\mu_i)(X_j-\mu_j)] = \int (x_i-\mu_i)(x_j-\mu_j)p(x;\mu,\Sigma)dx = \Sigma_{ij}$$

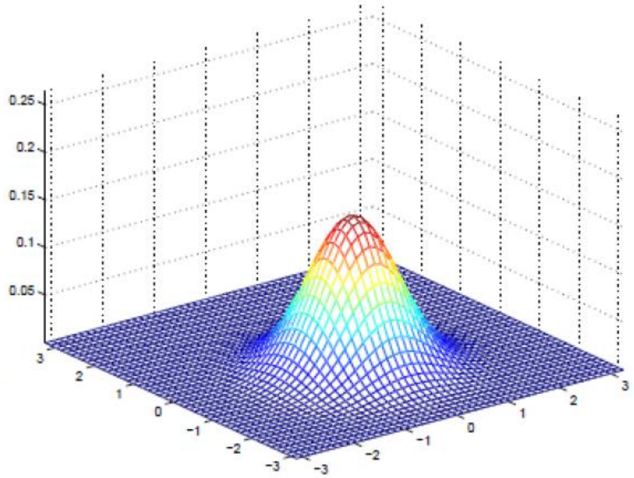
$$\mathsf{E}_X[(X-\mu)(X-\mu)^\top] = \int [(X-\mu)(X-\mu)^\top p(x;\mu,\Sigma)dx = \Sigma$$

(integral of matrix = matrix of integrals of each entry)

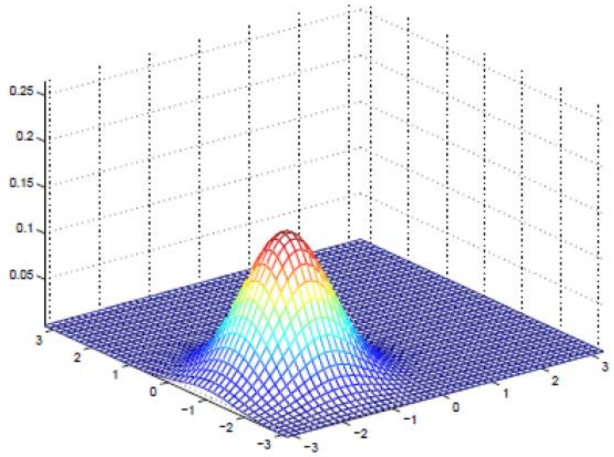
Multivariate Gaussian: examples



- $\mu = [1; 0]$
- $\Sigma = [1 \;\; 0; 0 \;\; 1]$

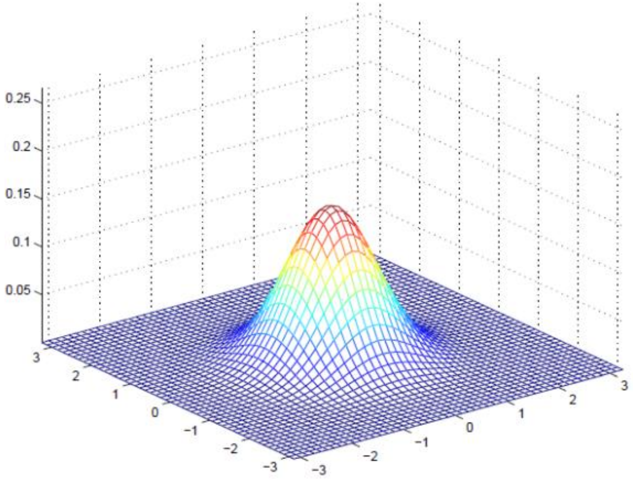


- $\mu = [-.5; 0]$
- $\Sigma = [1 \;\; 0; 0 \;\; 1]$

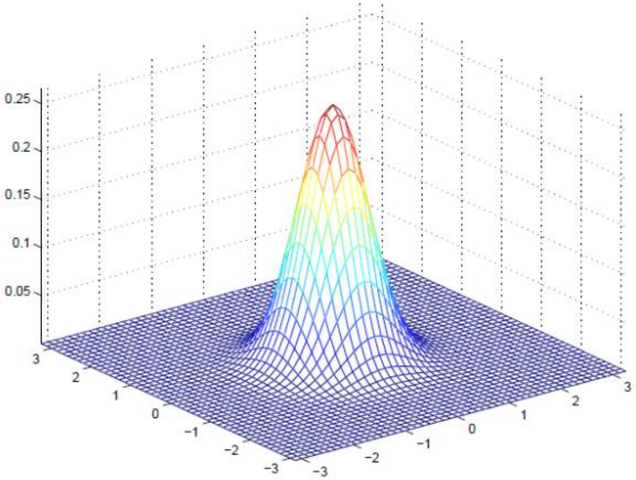


- $\mu = [-1; -1.5]$
- $\Sigma = [1 \;\; 0; 0 \;\; 1]$

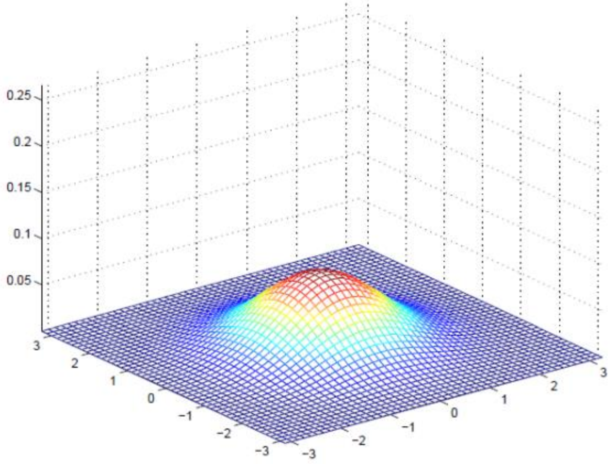
Multivariate Gaussian: examples



- $\mu = [0; 0]$
- $\Sigma = [1\ 0\ ;\ 0\ 1]$

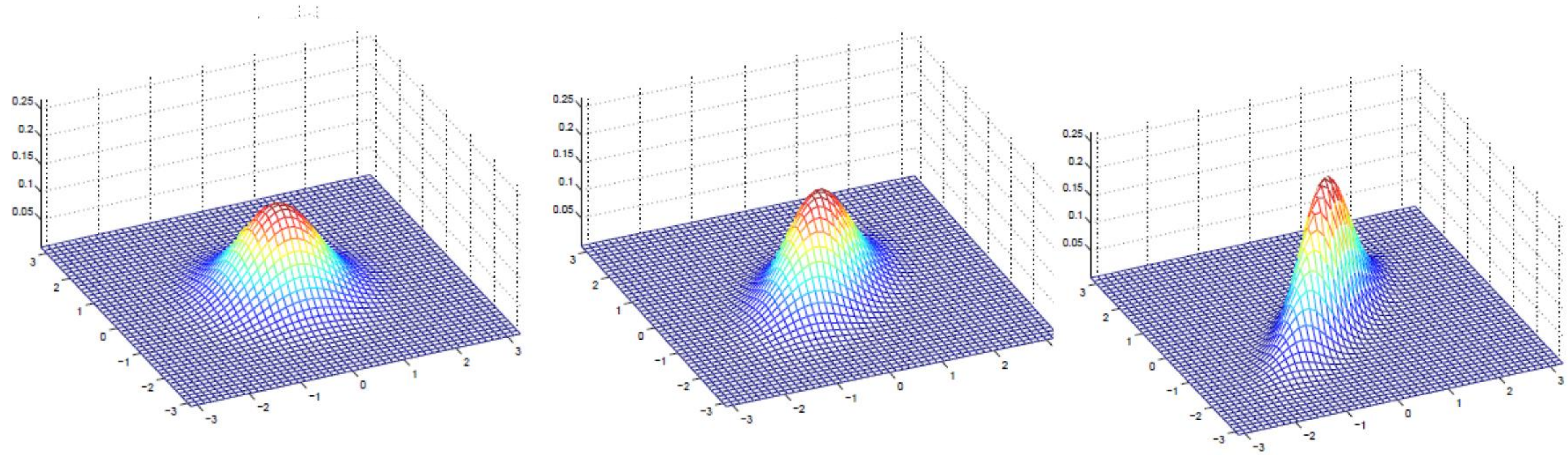


- $\mu = [0; 0]$
- $\Sigma = [.6\ 0\ ;\ 0\ .6]$



- $\mu = [0; 0]$
- $\Sigma = [2\ 0\ ;\ 0\ 2]$

Multivariate Gaussian: examples

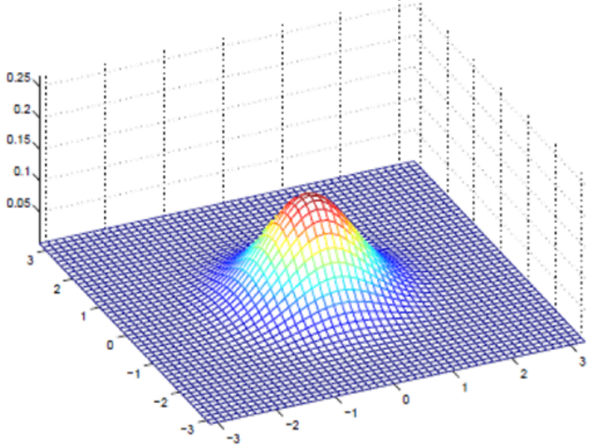


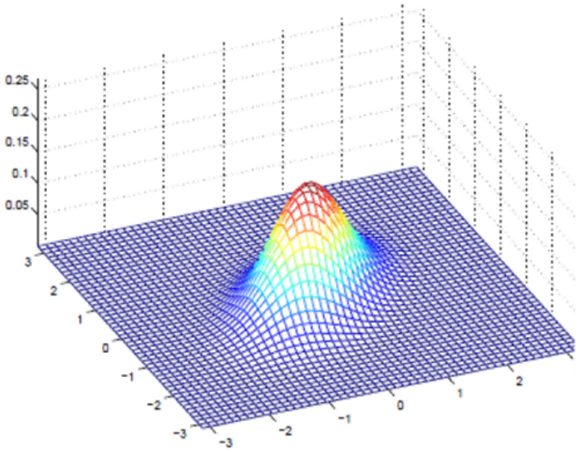
- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0; 0 \;\; 1]$

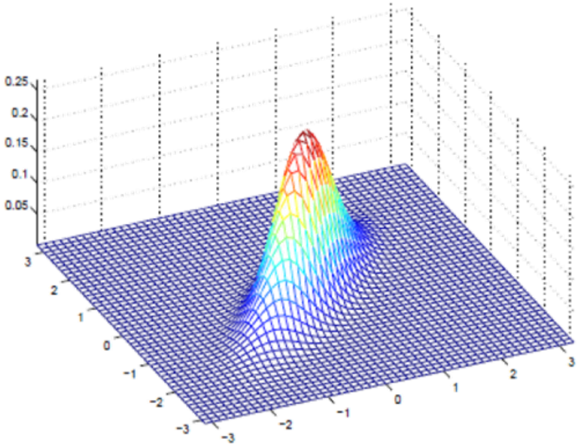
- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0.5; 0.5 \;\; 1]$

- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0.8; 0.8 \;\; 1]$

Multivariate Gaussian: examples



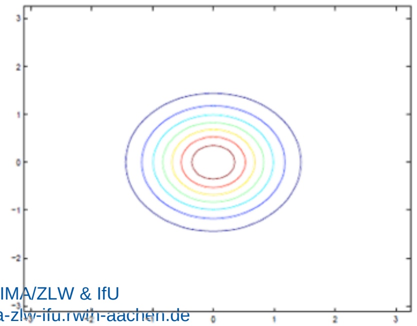


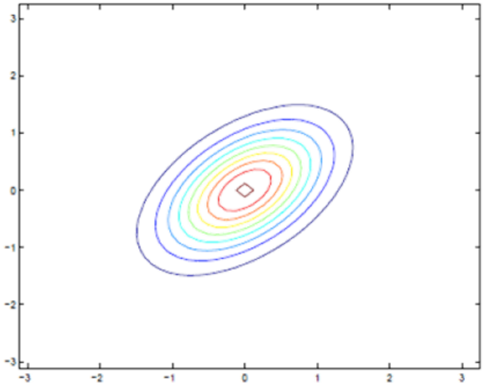


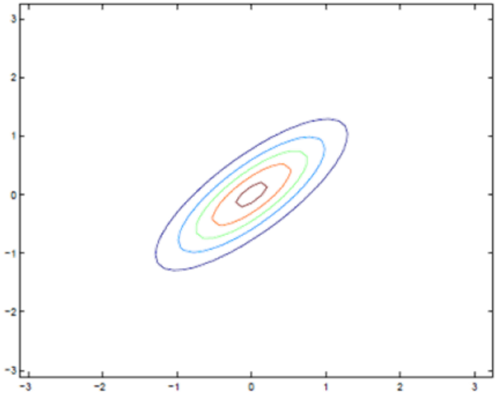
- $\mu = [0;\ 0]$
- $\Sigma = [1\ \ 0;\ 0\ \ 1]$

- $\mu = [0;\ 0]$
- $\Sigma = [1\ \ 0.5;\ 0.5\ \ 1]$

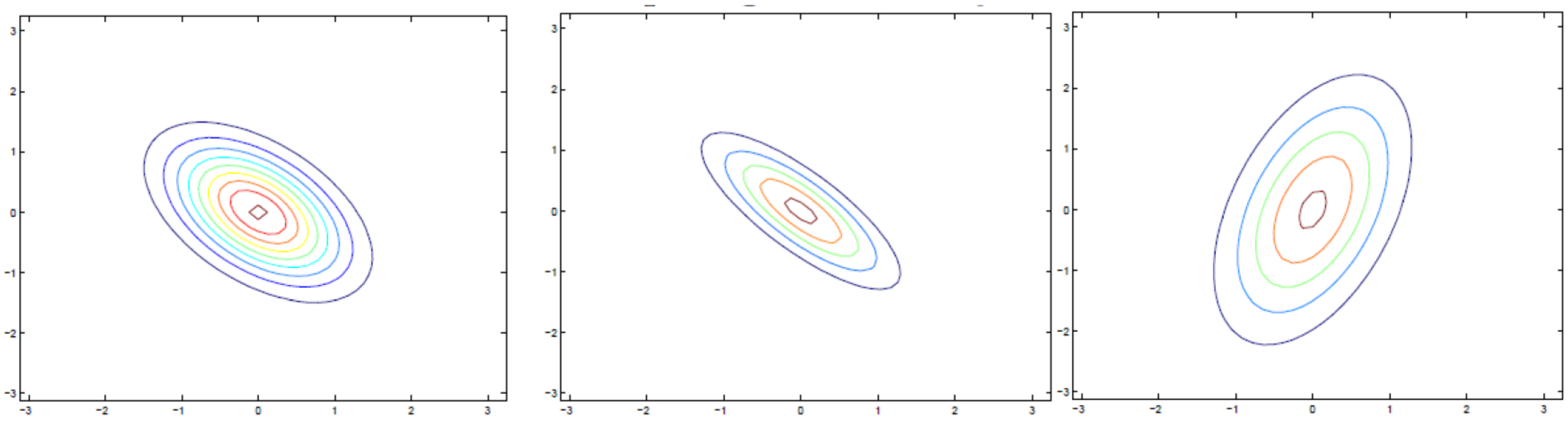
- $\mu = [0;\ 0]$
- $\Sigma = [1\ \ 0.8;\ 0.8\ \ 1]$







Multivariate Gaussian: examples



- $\mu = [0; 0]$
- $\Sigma = [1\ \ -0.5\ ;\ -0.5\ \ 1]$

- $\mu = [0; 0]$
- $\Sigma = [1\ \ -0.8\ ;\ -0.8\ \ 1]$

- $\mu = [0; 0]$
- $\Sigma = [3\ \ 0.8\ ;\ 0.8\ \ 1]$

Conditioning a multivariate Gaussian

- Consider a multi-variate Gaussian $$\mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

- And the precision matrix $$\Gamma = \Sigma^{-1} = \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}^{-1} = \begin{bmatrix}\Gamma_{XX} & \Gamma_{XY} \\ \Gamma_{YX} & \Gamma_{YY}\end{bmatrix}$$

- And the partitioned multi-variate Gaussian:

$$p(\begin{bmatrix}x \\ y\end{bmatrix}; \mu, \Sigma) = \frac{1}{(2\pi)^{(n/2)}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\left(\begin{bmatrix}x \\ y\end{bmatrix} - \begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}\right)^\top \begin{bmatrix}\Gamma_{XX} & \Gamma_{XY} \\ \Gamma_{YX} & \Gamma_{YY}\end{bmatrix}\left(\begin{bmatrix}x \\ y\end{bmatrix} - \begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}\right)\right)$$

- We have

$$\begin{aligned}
p(x|Y = y_0) &\propto p(\begin{bmatrix}x \\ y_0\end{bmatrix}; \mu, \Sigma) \\
&\propto \exp\left(-\frac{1}{2}(x-\mu_X)^\top\Gamma_{XX}(x-\mu_X) - (x-\mu_X)^\top\Gamma_{XY}(y_0-\mu_Y) - \frac{1}{2}(y_0-\mu_Y)^\top\Gamma_{YY}(y_0-\mu_Y)\right) \\
&\propto \exp\left(-\frac{1}{2}(x-\mu_X)^\top\Gamma_{XX}(x-\mu_X) - (x-\mu_X)^\top\Gamma_{XY}(y_0-\mu_Y)\right) \\
&= \exp\left(-\frac{1}{2}(x-\mu_X)^\top\Gamma_{XX}(x-\mu_X) - (x-\mu_X)^\top\Gamma_{XX}\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y) - \frac{1}{2}(y_0-\mu_Y)\Gamma_{YX}\Gamma_{XX}^{-1}\Gamma_{XX}\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y) + \frac{1}{2}(y_0-\mu_Y)\Gamma_{YX}\Gamma_{XX}^{-1}\Gamma_{XX}\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y)\right) \\
&= \exp\left(-\frac{1}{2}(x-\mu_X+\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y))^\top\Gamma_{XX}(x-\mu_X+\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y))\right)\exp\left(\frac{1}{2}(y_0-\mu_Y)\Gamma_{YX}\Gamma_{XX}^{-1}\Gamma_{XX}\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y)\right) \\
&\propto \exp\left(-\frac{1}{2}(x-\mu_X+\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y))^\top\Gamma_{XX}(x-\mu_X+\Gamma_{XX}^{-1}\Gamma_{XY}(y_0-\mu_Y))\right)
\end{aligned}$$

Gaussian

Conditioning a multivariate Gaussian

- If

$$(X,Y) \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

- Then:

$$\begin{aligned} X|Y = y_0 \quad &\sim \quad \mathcal{N}(\mu_X - \Gamma_{XX}^{-1}\Gamma_{XY}(y_0 - \mu_Y), \Gamma_{XX}) \\ &= \quad \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y_0 - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}) \end{aligned}$$

  – Mean moved according to correlation and variance on measurement
  – Covariance $\Sigma_{XX\,|\,Y=y0}$ does not depend on y_0

Content

I. Gaussians

   I. Univariate Gaussian
   II. Multivariate Gaussian
   III. Conditioning (Bayes' rule)

II. Gaussian Mixture Model

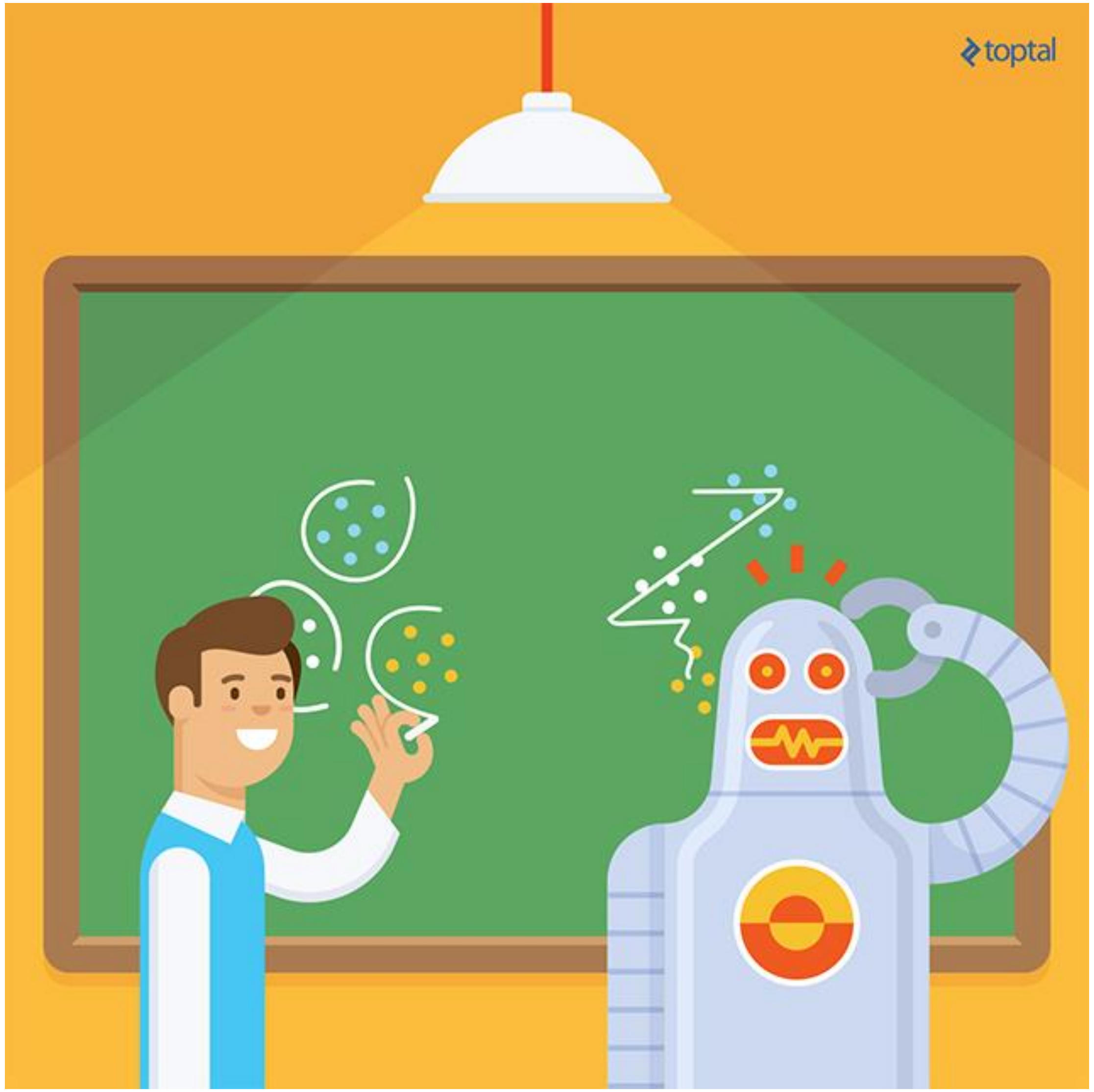
III. Learning Local Dynamic Models

Disclaimer: lots of linear algebra now. In fact, pretty much all computations with Gaussians will be reduced to linear algebra!

Gaussian Mixture Model

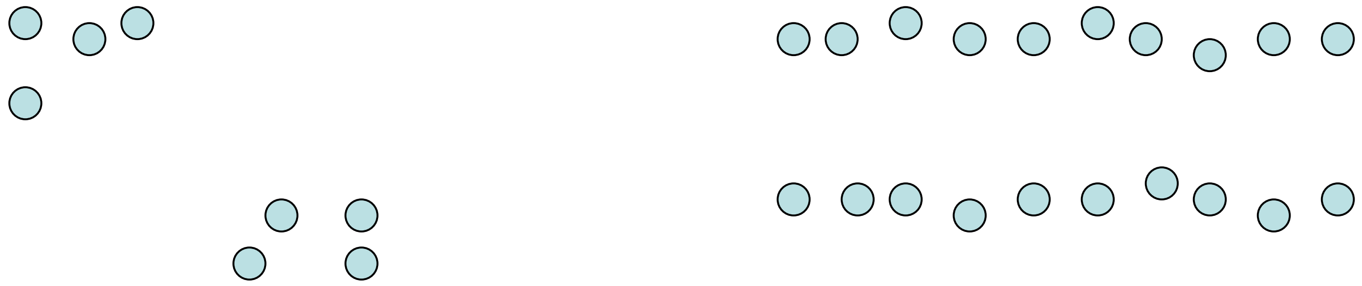
Clustering

- Unsupervised Learning
- Detect patterns in unlabelled data
- Useful if you do not know what to look for
- Requires data, but no labels



Clustering

- Fundamental approach: Group similiar instances
- Example: 2D pattern



- What does similarity mean?

What does similarity mean?

- Similarity is hard to define
  – But we know it if we see it

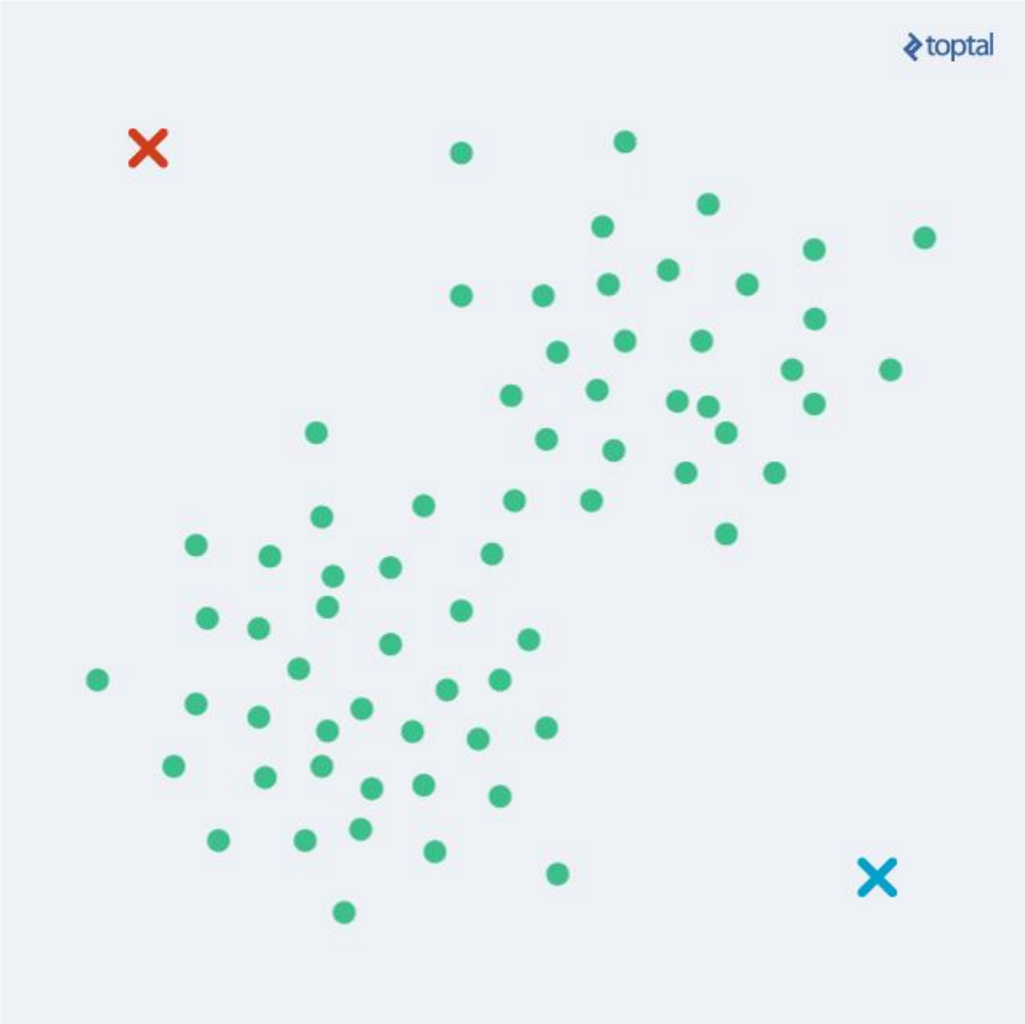
- The true meaning of similarity is a philosophical question. We will therefore choose a more pragmatic approach: we think of **distances** (rather than similarities) between vectors or correlations between random variables

Hard Clustering with K-Means

K-Means Algorithm
1. Choose K random points as cluster centers 2. Assign datapoint to the nearest cluster center 3. Adjust cluster center so that it get’s the mean value of the associated points

Problem: correct assignment is hard!
– Sometimes distances can be deceiving!
– Cluster may overlap!

What about probabilistic clustering?



The General Gaussian Mixture Model Assumption

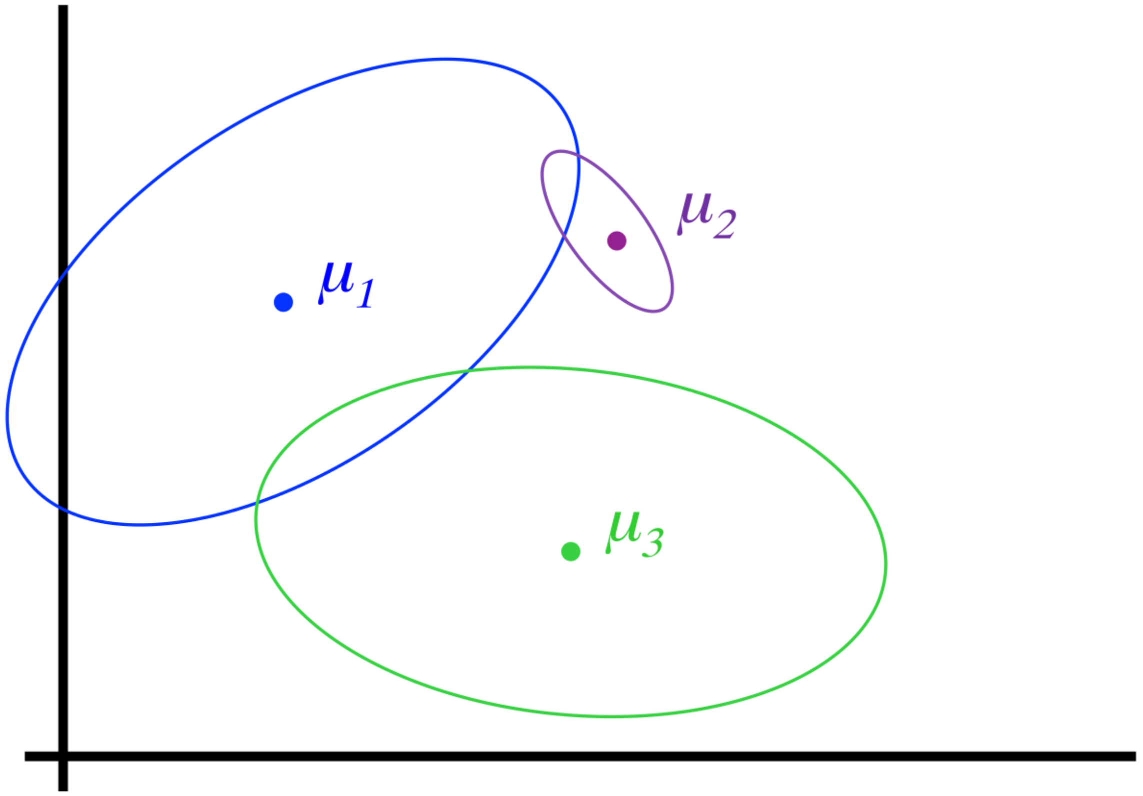
- “The probabilistic version of K-Means”
- Each cluster has not only mean μ_i but also associated covariance matrix Σ_i
- GMM is a linear combination of *K* Gaussians given by

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}|k)P(k)$$

where $P(k)$ is mixture weight subject to constraints

$$0 \leq P(k) \leq 1 \quad \textbf{and} \quad \sum_{k=1}^{K} P(k) = 1$$

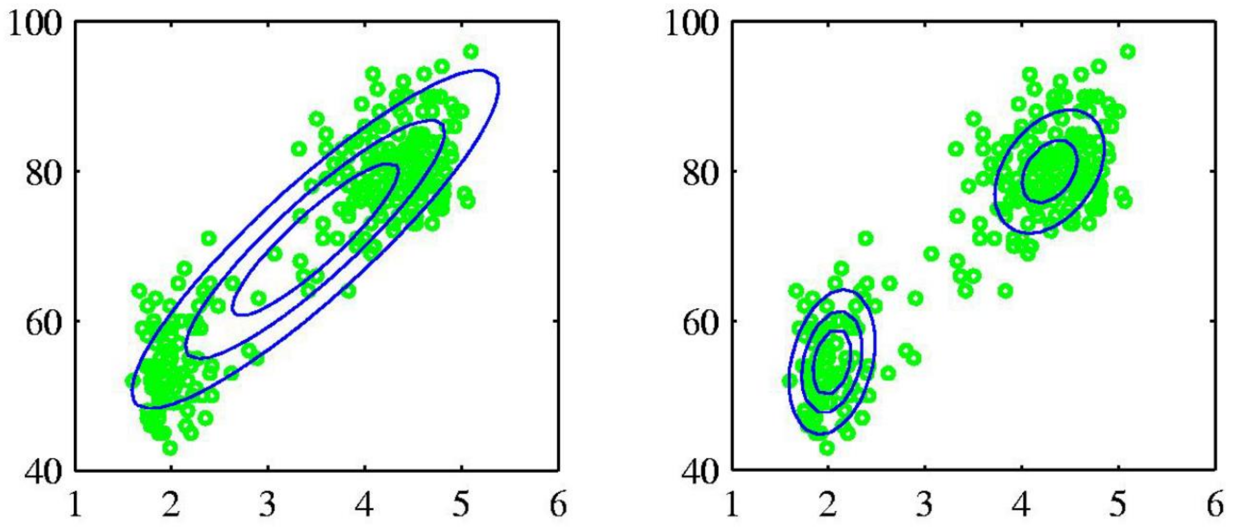
and $p(x|k)$ is height of $k\text{'}th$ Gaussian at datapoint x



The General Gaussian Mixture Model Assumption

- Gaussian Mixture Model
  - $P(Y)$ is multinomial
  - $p(x|k)$ is a multivariate Gaussian Distribution

$$P(X=\mathbf{x}_j \mid Y=i) = \frac{1}{(2\pi)^{m/2} \|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu_i\right)^T \Sigma_i^{-1}\left(\mathbf{x}_j-\mu_i\right)\right]$$



Single Gaussian

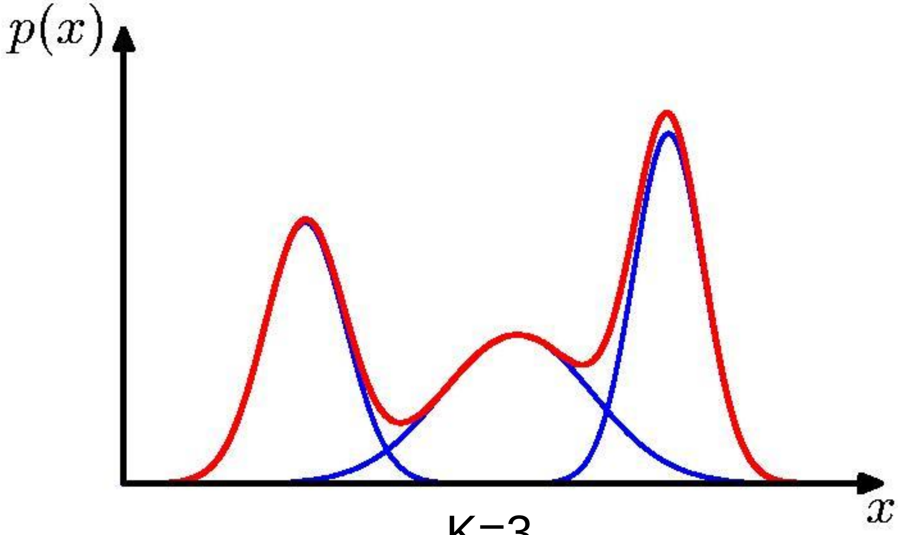
Mixture of two Gaussians

Combine simple models into a complex model

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \underbrace{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

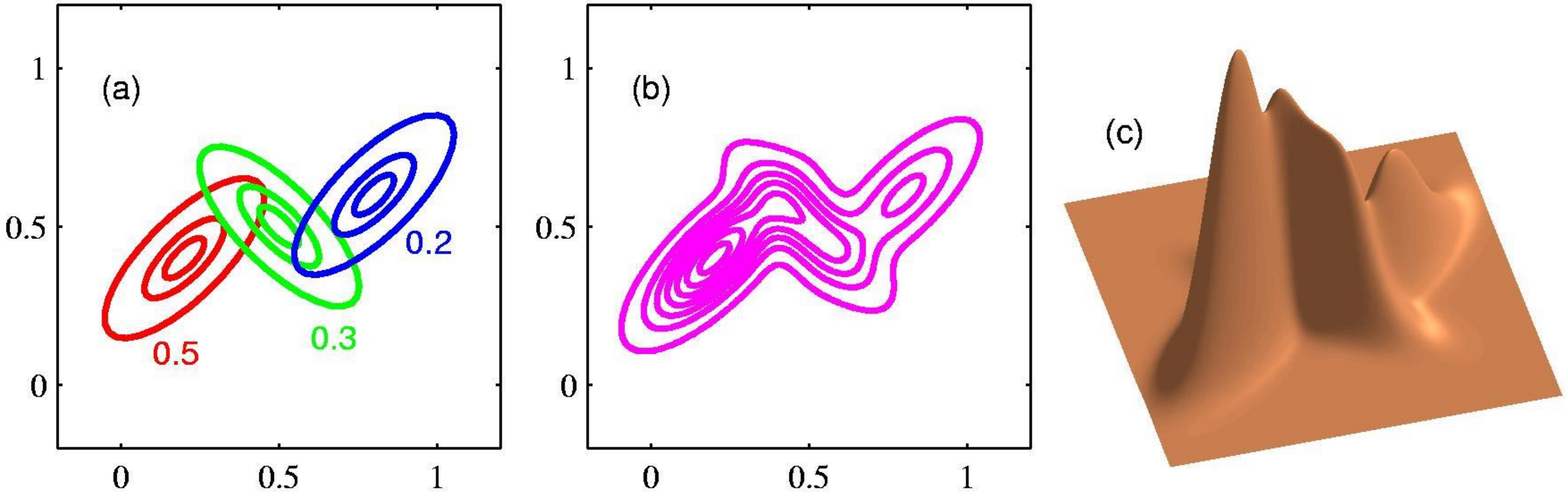
Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



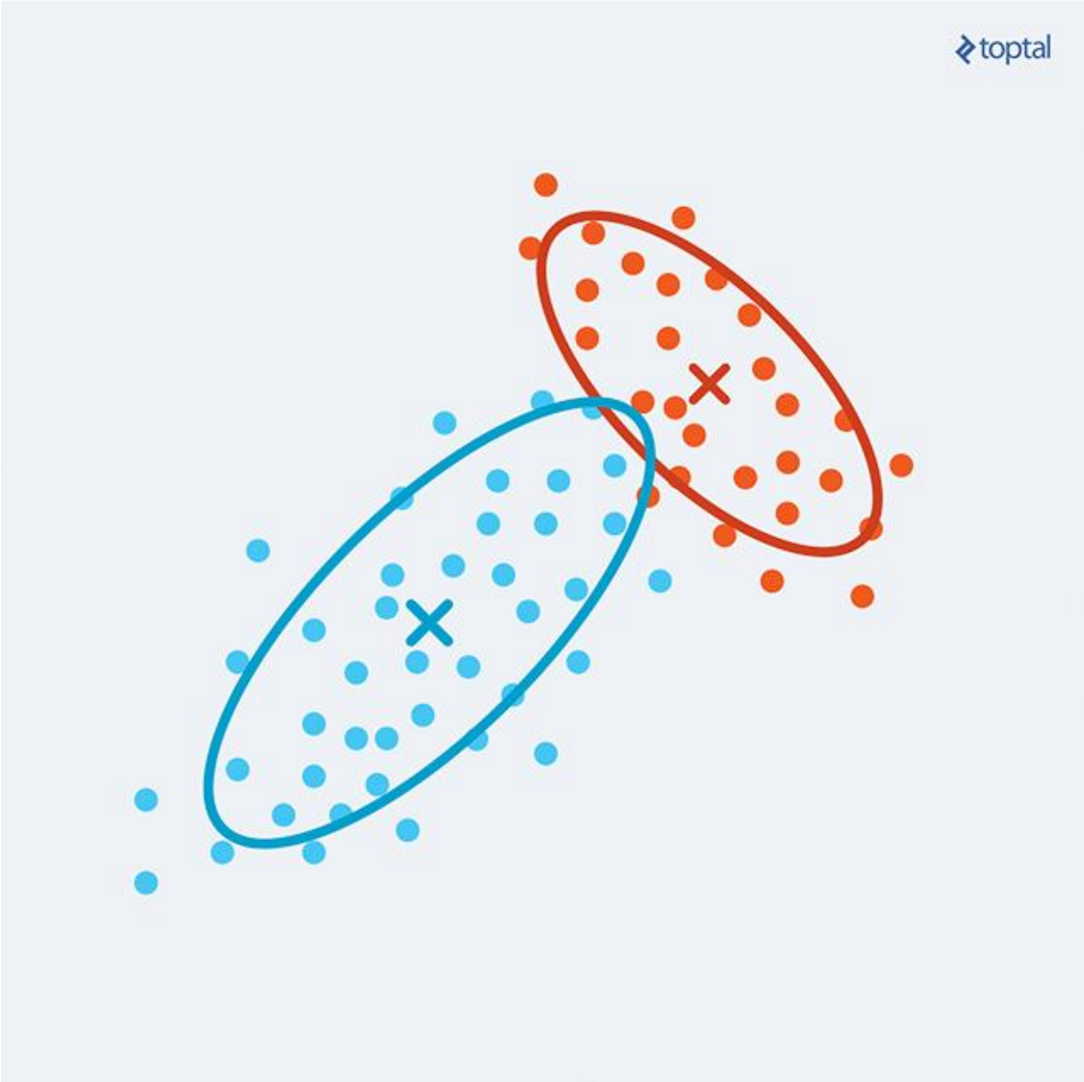
K=3

Combine simple models into a complex model: Example



Soft-Clustering with Expectation Maximization and GMM

Expectation-Maximization Algorithm
1. Choose K random mean and covariances as clusters 2. E-Step: Calculate, for each point, the probabilities of it belonging to each of the clusters 3. M-Step: recalculate mean and covariance of each cluster, using the probability of belonging to each cluster

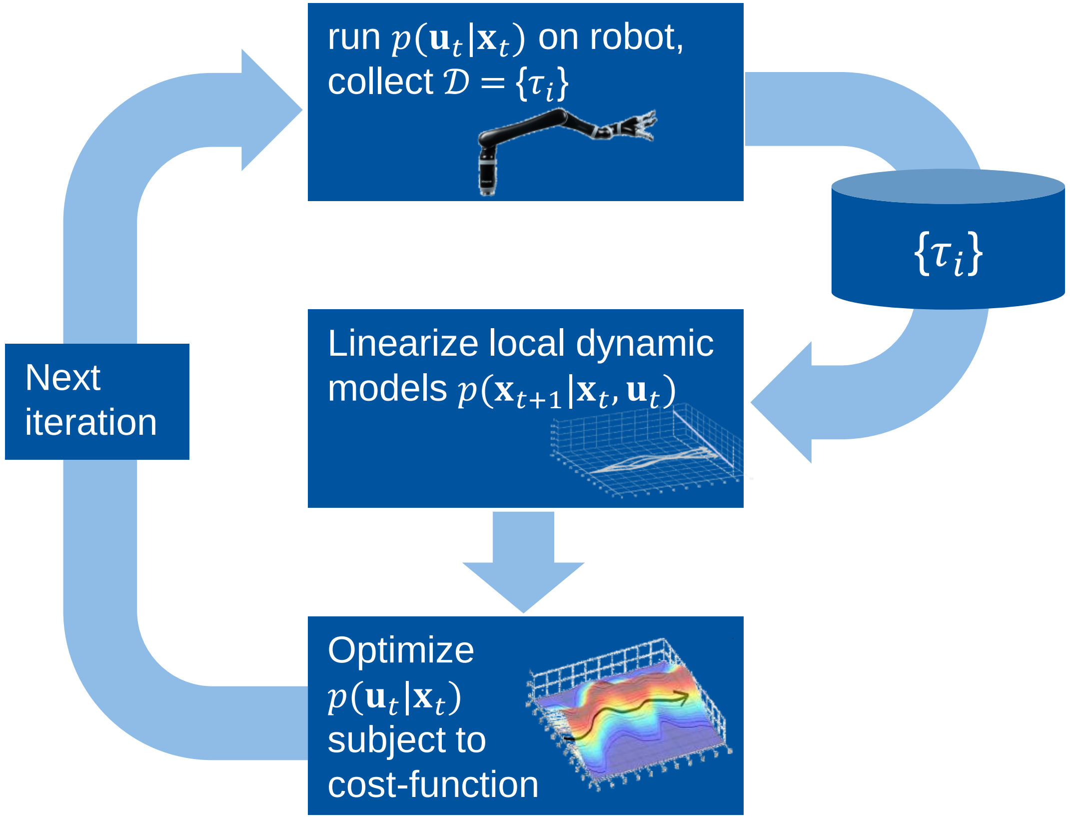


Learning a Policy

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$$

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$

$$\mathbf{A}_t = \frac{\partial f}{\partial \mathbf{x}_t} \qquad \mathbf{B}_t = \frac{\partial f}{\partial \mathbf{u}_t}$$



Linearized local dynamics

Goal: get the system dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ for each timestep t

Data: samples generated by the previous controller $\widehat{p}_i(\mathbf{u}_t|\mathbf{x}_t) \rightarrow \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})_i\}$

Linear Gaussian Dynamics are defined as

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f_{xt}\mathbf{x}_t + f_{ut}\mathbf{u}_t + f_{ct}), \mathbf{F}_t)$$

How can we determine linear Gaussian dynamics from few samples?

Learning Local and Global Models

run $p(\mathbf{u}_t|\mathbf{x}_t)$ on robot, collect $\mathcal{D} = \{\tau_i\}$

$\{\tau_i\}$

Data contains samples from all previous iteration

Train GMM: **Global Dynamic Model**
- Uses data from nearby timesteps
- Uses data from prior iterations

Train GMM on all $\{\tau_i\} = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})_i\}$

Obtain prior parameters from GMM (mean μ_0, covariance $\boldsymbol{\Phi}$, degree of freedom n_0, m (number of datapoints))

Get normal-inverse-Wishart ($\mathcal{NIW}$) prior from GMM for a given tuple $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Local data $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$
From last iteration

Linearize: **Local Dynamic Model**
- Uses prior from local dynamic model
- Uses data from last iteration at timestep t
- Condition on given $(\mathbf{x}_t, \mathbf{u}_t)$

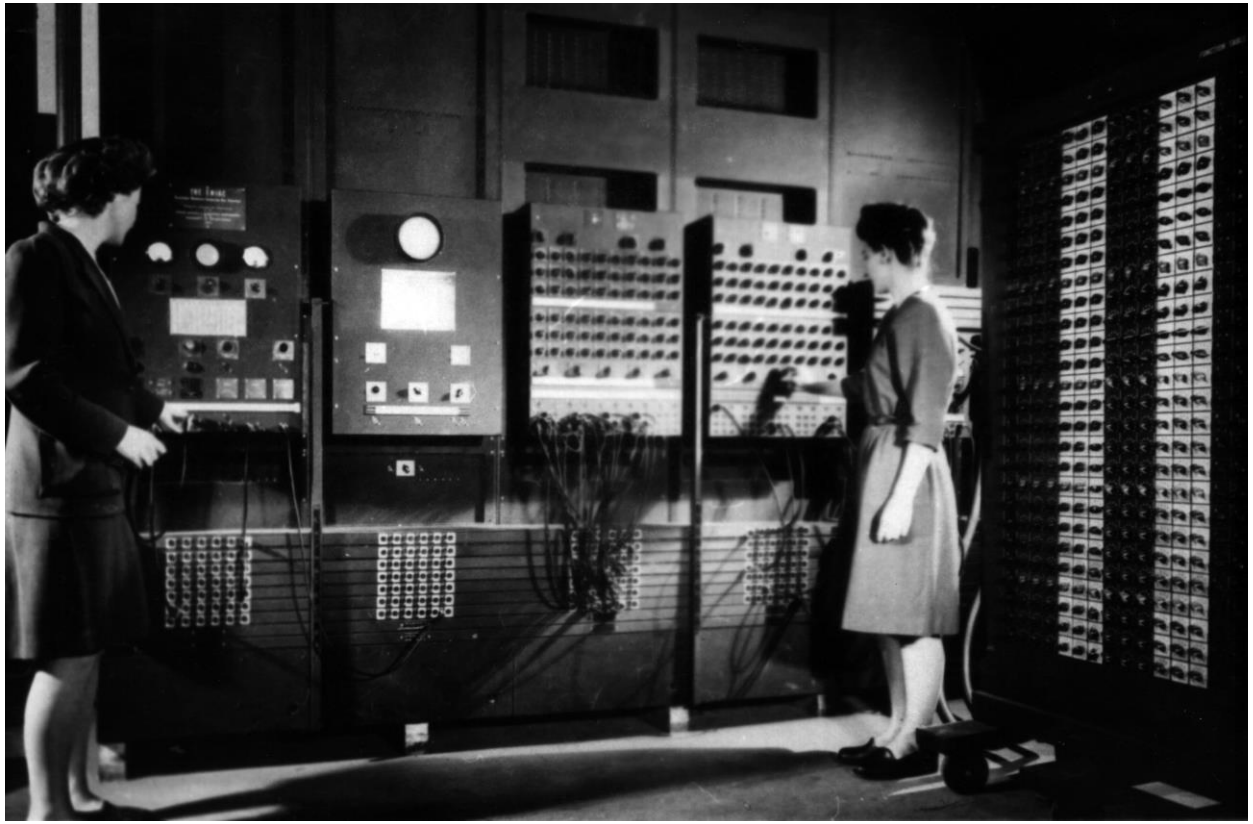
Calculate posterior covariance using $\mathcal{NIW}$-Priorparameters
Calculate posterior mean using sampled data

Condition on given $(\mathbf{x}_t, \mathbf{u}_t)$

$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ Local linear Gaussian dynamics

Its your turn!

Visit the website and implement it!



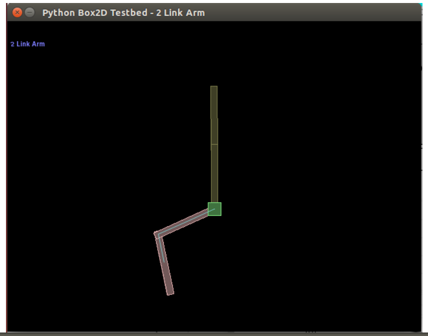
Introduction to the tasks

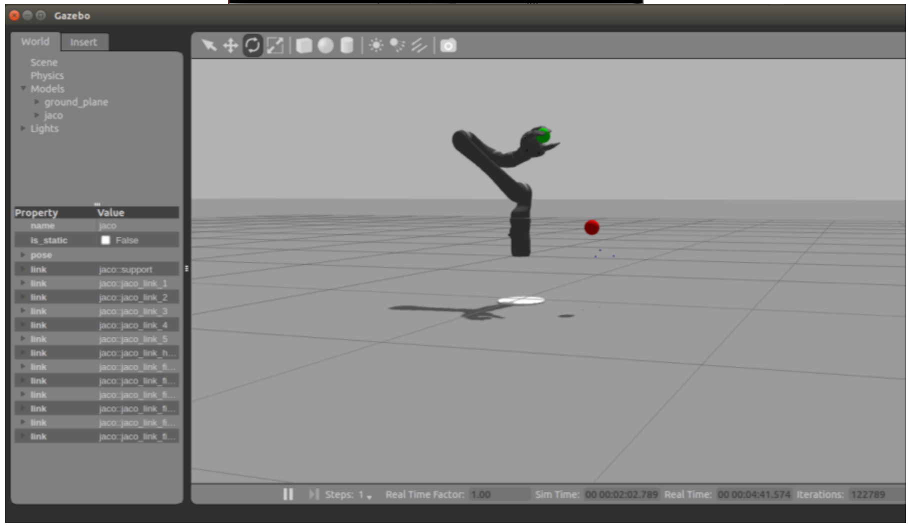
Tasks for today and tomorrow

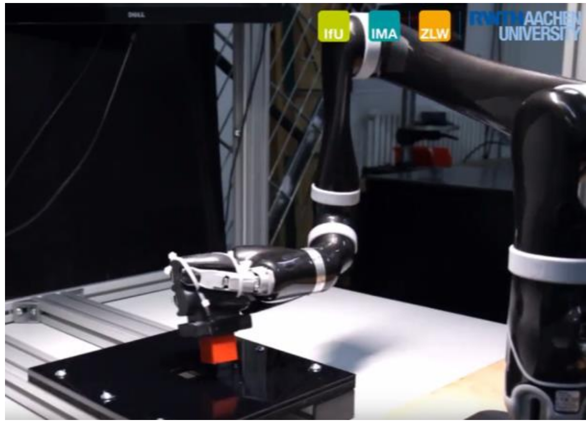
- Task 1:
  – Implement an LQR Backward and Forward pass
  – Try to understand it!
  – Test it with our test method

- Task 2:
  – Implement linearization of the dynamic model
  – Try to understand it!
  – Test it with our test-method
  – Test it on the Box2D Scenario

- Task 3:
  – Test it with Kinova Jaco 2 in simulation
  – Adjust cost function







Task 2 – Installation procedure

Download source code (do it in your home directory: `cd ~`):

```
git clone https://github.com/philippente/task2_dynamics.git
```

Edit .bashrc to set environment variables:

```
gedit ~/.bashrc
```

At the end of file, the lines should look like this:

```
source /opt/ros/indigo/setup.bash
source /home/<USERNAME>/catkin_ws/devel/setup.bash
export
ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/opt/ros/indigo/share:/opt/ros/indigo/stacks:/home/<USERNAME>/task2_dynamics:/home/<USERNAME>/task2_dynamics/src/gps_agent_pkg
```

Check if the blue part of the source folder and ROS_PACKAGE_PATH is correct!

Then save it and close it. Source the .bashrc (load the environment variables):

```
source ~/.bashrc
```

Now, compile some stuff:

```
cd task2_dynamics
catkin_make
```

Task 2 – Installation procedure

- Open PyCharm
- Import the folder *task2_dynamics* as a new project
- Open within PyCharm: python/gps/algorithm/dynamics/dynamics_lr_prior.py
- **Task: Implement dynamics learning! Look at the website for advices: https://philippente.github.io/irobotics.html**

- You can test your implementation with a little test program
  - using a terminal, open the directory *task2_dynamics*
  - Start the program with: `python python/gps/dynamics_test.py`
  - Was it successful?

- If it was successful, lets look at the Box2D scenario!
  - using a terminal, open the directory *task2_dynamics*
  - Start it with `python python/gps/gps_main.py box2d_arm_example`
  - How is the performance?