EINDHOVEN UNIVERSITY OF TECHNOLOGY

Schedule-free variational message-passing for Bayesian filtering

Wouter M. Kouw
2020-03-31

neuromatch

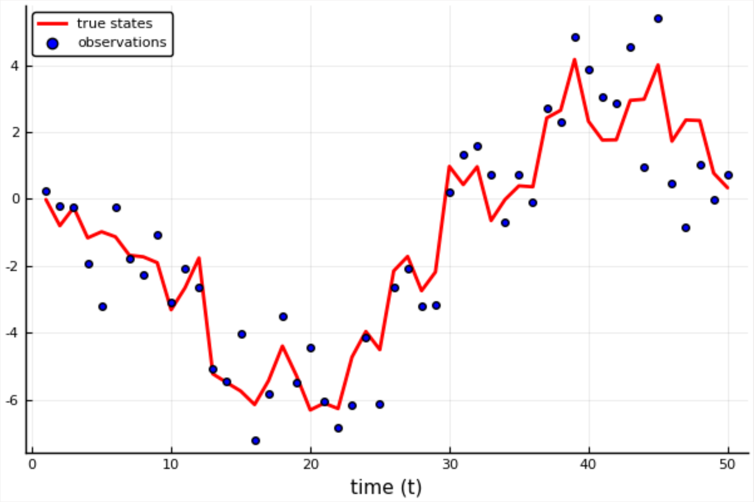
Bayesian Intelligent Autonomous Systems lab

Bayesian filtering

Goal: recover underlying states and parameters from a noisy signal.

State-space model:

$$p(y_{1:T}, x_{1:T}) = \underbrace{p(x_0)}_{\text{prior}} \prod_{t=1}^{T} \underbrace{p(x_t \mid x_{t-1})}_{\text{state transition}} \underbrace{p(y_t \mid x_t)}_{\text{likelihood}}$$



Free Energy Principle

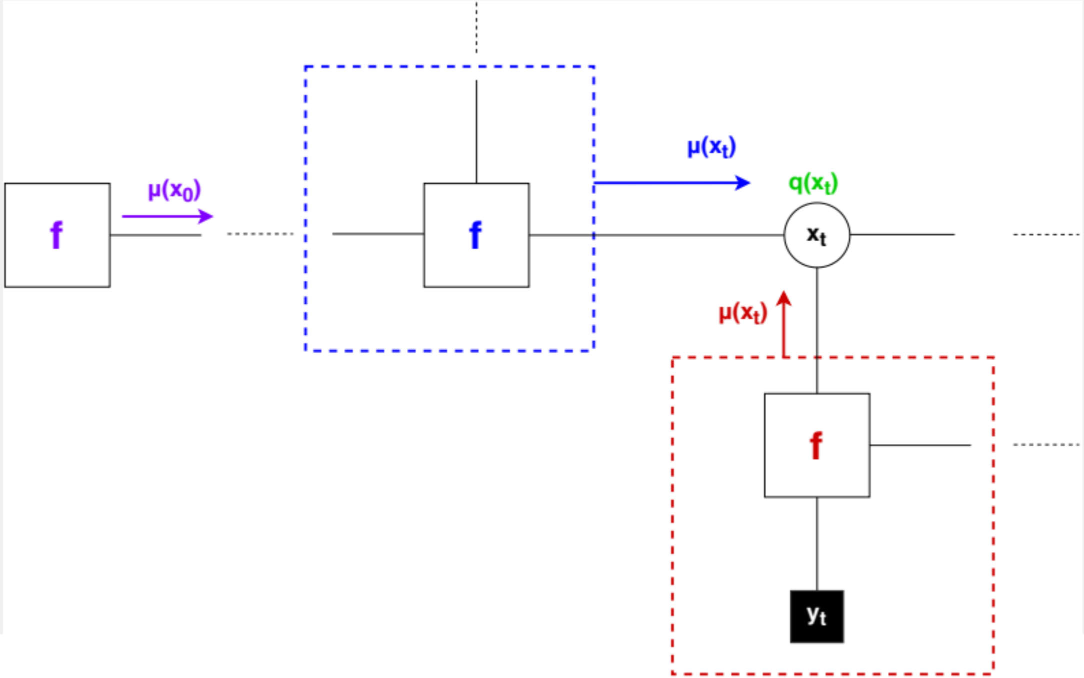
Form a Free Energy function with beliefs *q* that approximate the generative model *p*:

$$\mathcal{F}[q] = \int q(x_{1:T}) \log \frac{q(x_{1:T})}{p(y_{1:T}, x_{1:T})} \, \mathrm{d}x_{1:T}$$

→ Minimising Free Energy = updating beliefs *q* to match the posterior *p*.

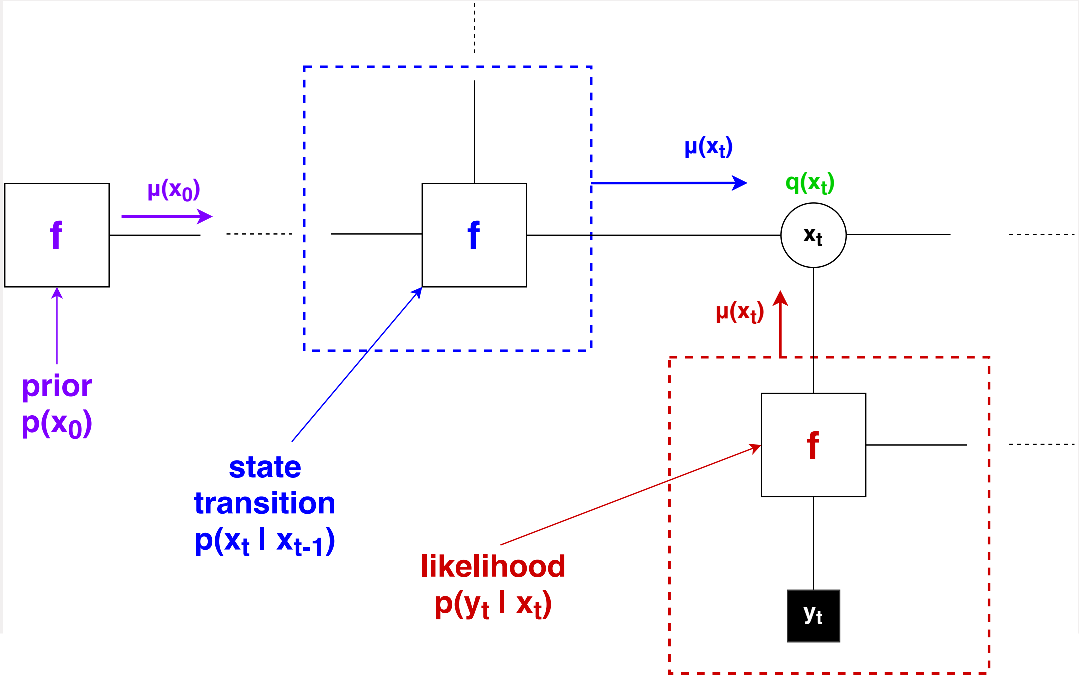
Message passing

Model factorises → build factor graph and perform variational message passing:



Message passing

Model factorises → build factor graph and perform variational message passing:



Scheduler

Currently, automatic message passing tools such as ForneyLab.jl employ a master scheduler:

```
function stepSMF!(data::Dict, marginals::Dict=Dict(), messages::Vector{Message}=Array{Message}(undef, 40))

    messages[1] = ruleVBGaussianMeanVarianceOut(nothing, ...
    messages[2] = ruleVBGaussianMeanPrecisionM(marginals[:s_1], nothing, marginals[:w])
    messages[3] = ruleVBGaussianMeanPrecisionOut(nothing, marginals[:s_0], marginals[:w])
    messages[4] = ruleVBGaussianMeanPrecisionM(marginals[:s_2], nothing, marginals[:w])
    messages[5] = ruleVBGaussianMeanVarianceM(ProbabilityDistribution(Univariate, PointMass, m=data[:x][1]), ...
            :
            :
    marginals[:s_0] = messages[1].dist * messages[2].dist
    marginals[:s_1] = messages[3].dist * messages[6].dist
    marginals[:s_2] = messages[7].dist * messages[10].dist
    marginals[:s_3] = messages[11].dist * messages[14].dist
    marginals[:s_4] = messages[15].dist * messages[18].dist
            :
            :

    return marginals
end
```

But such an algorithm / compiler is biologically implausible.

TU/e

Scheduler

Currently, automatic message passing tools such as ForneyLab.jl employ a master scheduler:

```
function stepSMF!(data::Dict, marginals::Dict=Dict(), messages::Vector{Message}=Array{Message}(undef, 40))

    messages[1] = ruleVBGaussianMeanVarianceOut(nothing, ...
    messages[2] = ruleVBGaussianMeanPrecisionM(marginals[:s_1], nothing, marginals[:w])
    messages[3] = ruleVBGaussianMeanPrecisionOut(nothing, marginals[:s_0], marginals[:w])
    messages[4] = ruleVBGaussianMeanPrecisionM(marginals[:s_2], nothing, marginals[:w])
    messages[5] = ruleVBGaussianMeanVarianceM(ProbabilityDistribution(Univariate, PointMass, m=data[:x][1]), ...
            :
            :
    marginals[:s_0] = messages[1].dist * messages[2].dist
    marginals[:s_1] = messages[3].dist * messages[6].dist
    marginals[:s_2] = messages[7].dist * messages[10].dist
    marginals[:s_3] = messages[11].dist * messages[14].dist
    marginals[:s_4] = messages[15].dist * messages[18].dist
            :
            :

    return marginals
end
```

But such an algorithm / compiler is biologically implausible.

Research Question: How can we perform message passing without a scheduler?

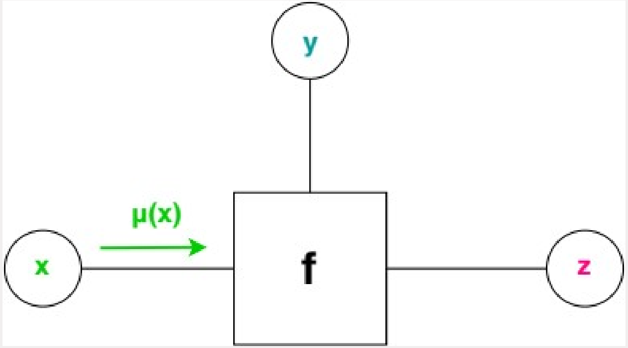
Schedule-free message passing

Consider the following set-up:

Schedule-free message passing

Consider the following set-up:

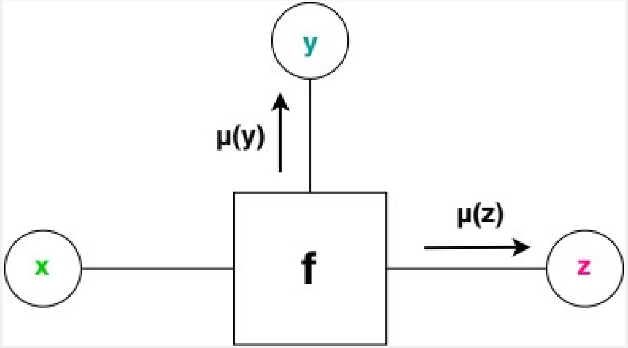
1. A factor node responds to an incoming belief by passing messages to all other variables:



Schedule-free message passing

Consider the following set-up:

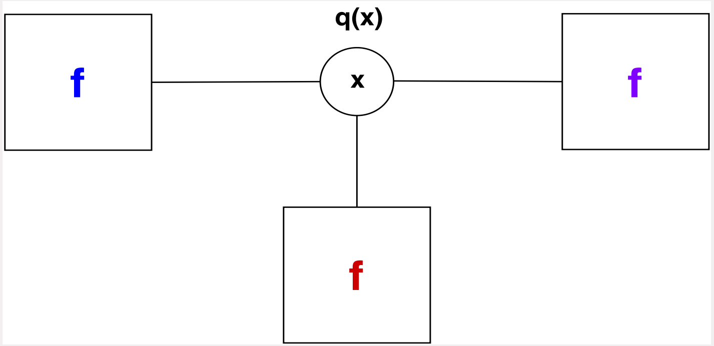
1. A factor node responds to an incoming belief by passing messages to all other variables:



Schedule-free message passing

Consider the following set-up:

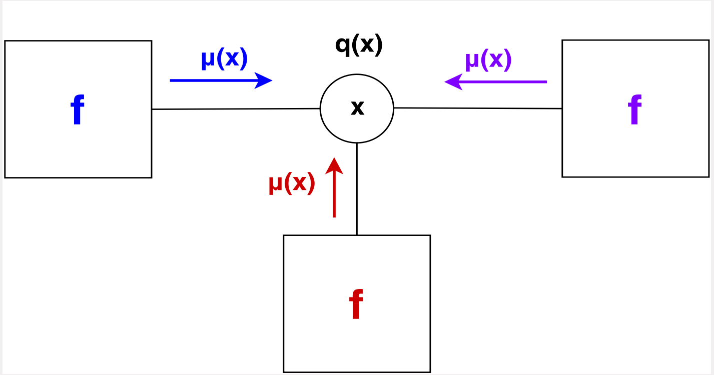
2. A variable node is updated based on the product of incoming messages:

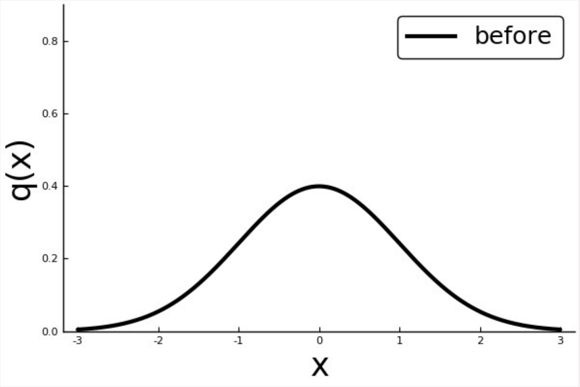


Schedule-free message passing

Consider the following set-up:

2. A variable node is updated based on the product of incoming messages:

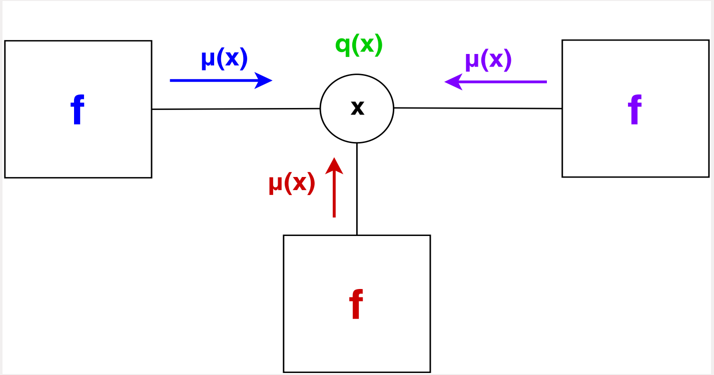


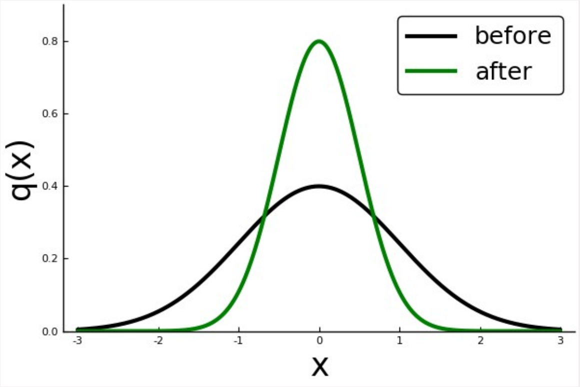


Schedule-free message passing

Consider the following set-up:

2. A variable node is updated based on the product of incoming messages:

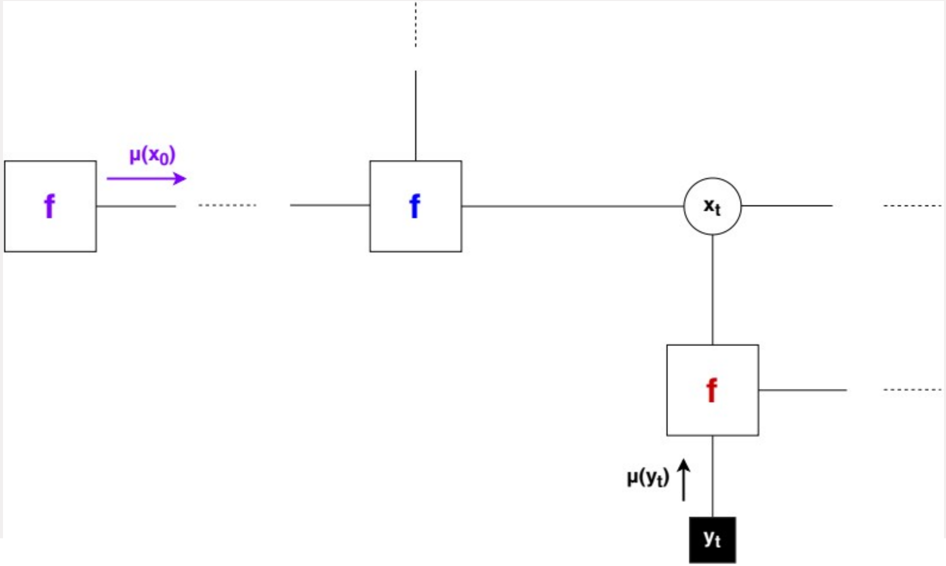




Schedule-free message passing

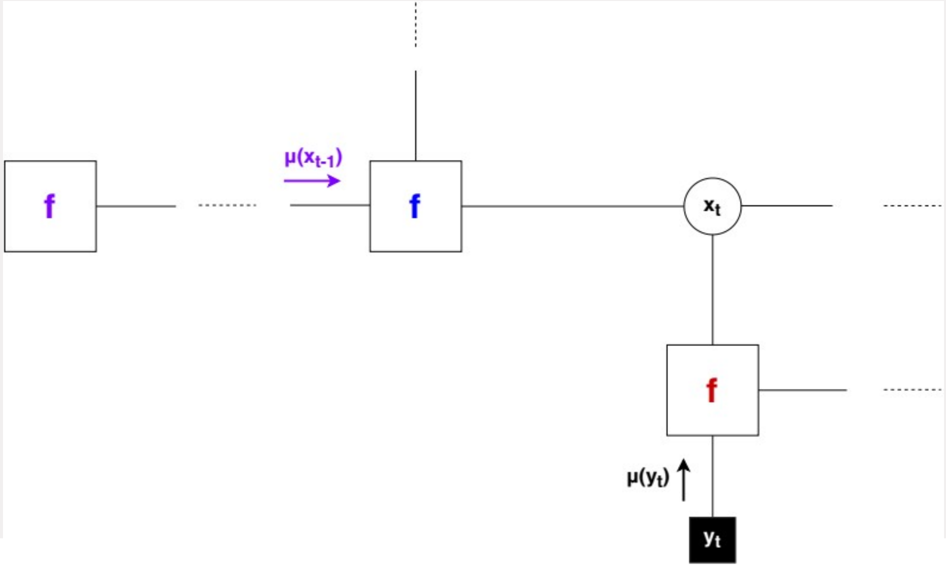
Consider the following set-up:

3. Inital state prior and observed variables start flow of messages through factor graph.



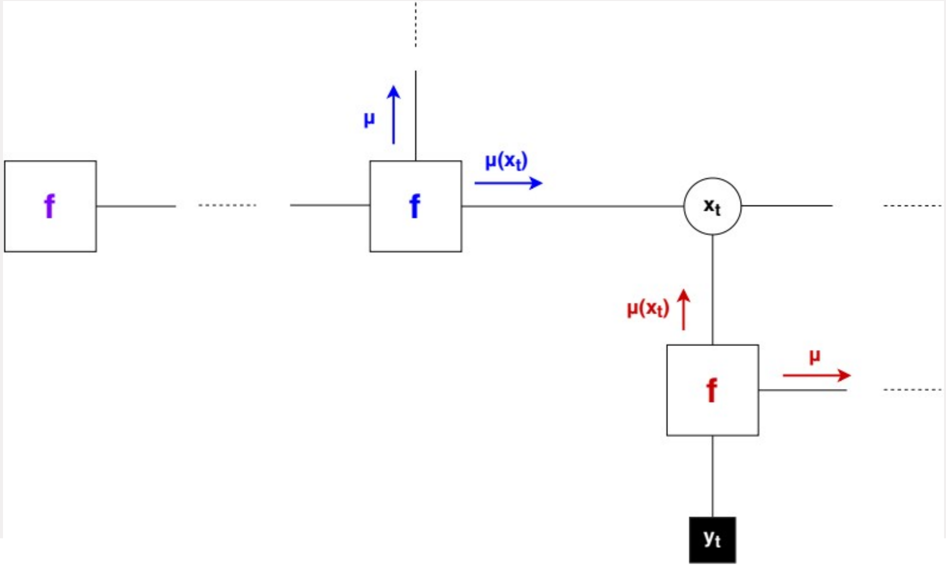
Schedule-free message passing

Messages now flow freely through the graph



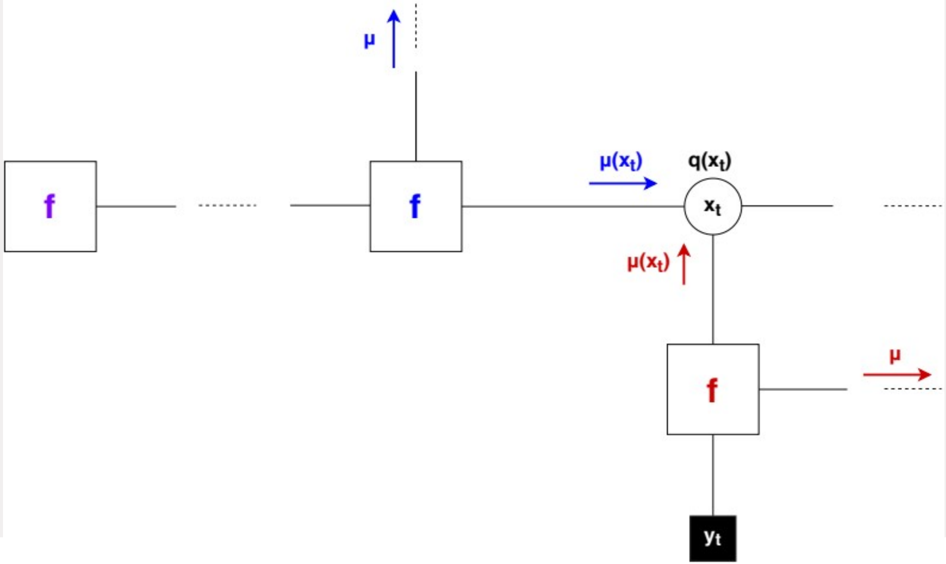
Schedule-free message passing

Messages now flow freely through the graph



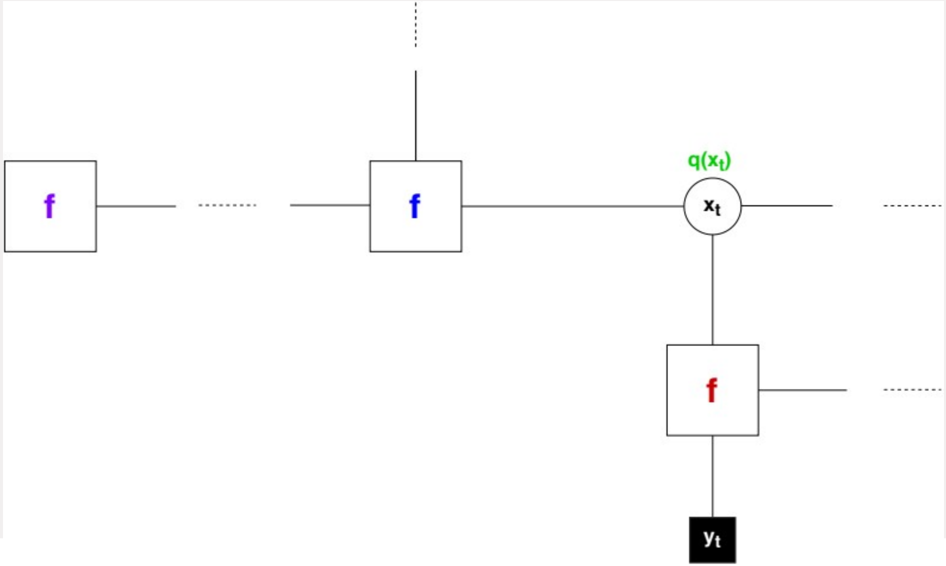
Schedule-free message passing

Messages now flow freely through the graph



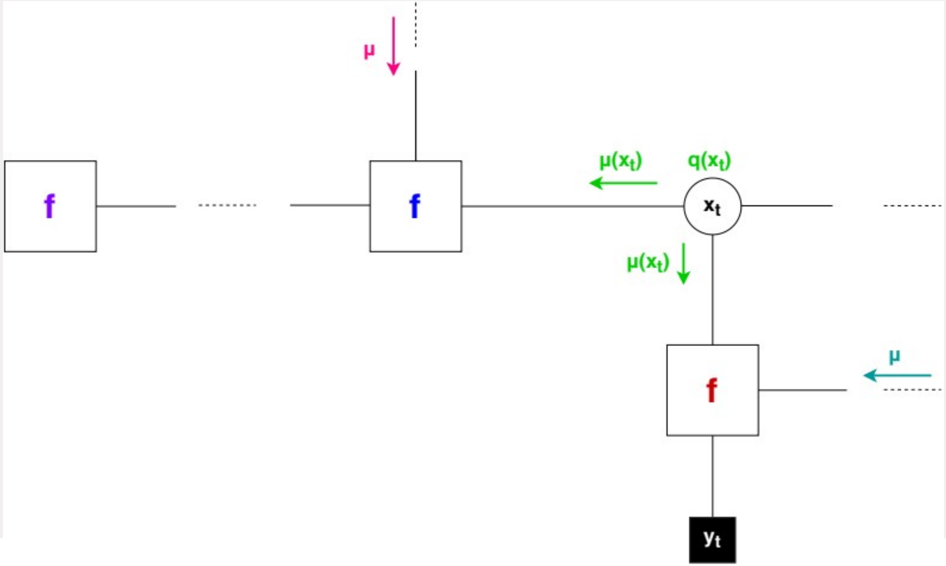
Schedule-free message passing

Messages now flow freely through the graph



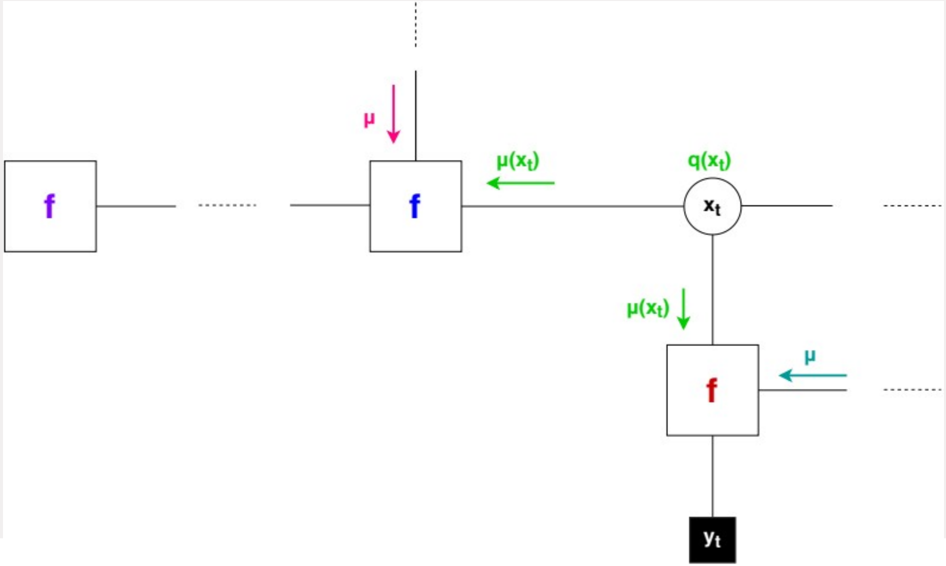
Schedule-free message passing

Messages now flow freely through the graph



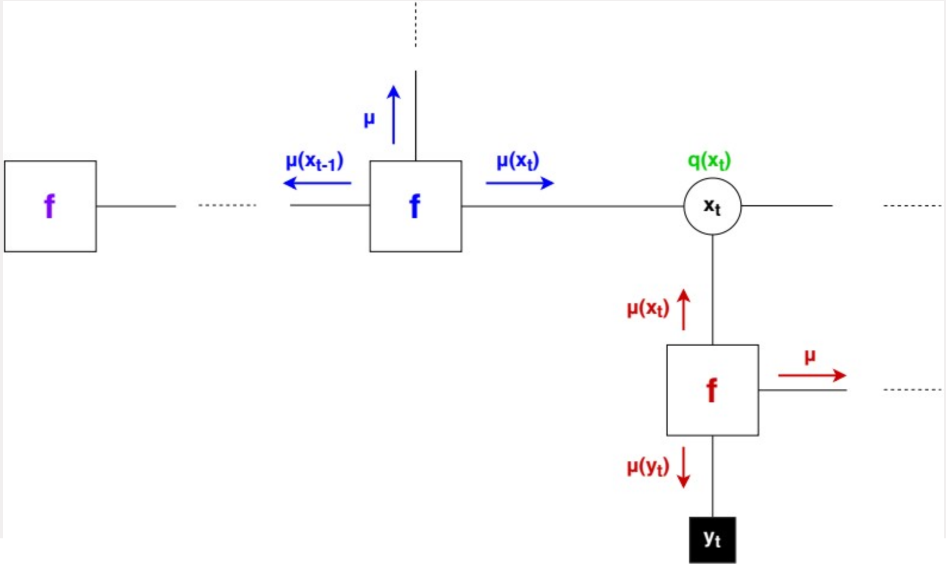
Schedule-free message passing

Messages now flow freely through the graph



Schedule-free message passing

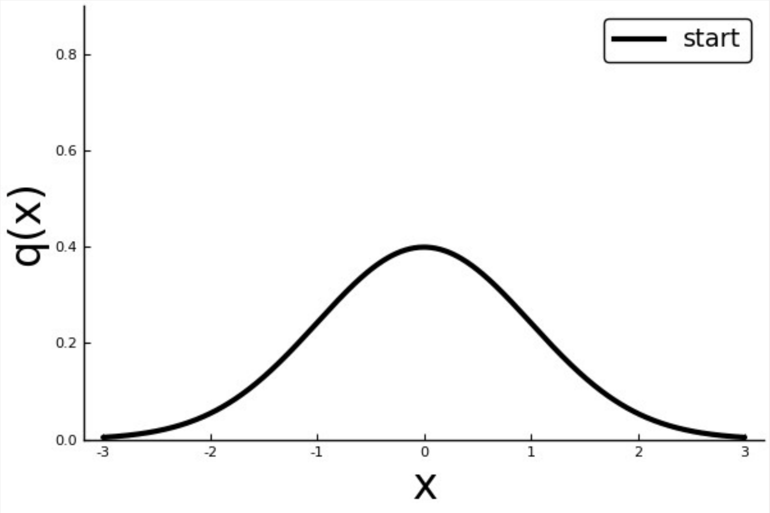
Messages now flow freely through the graph



Schedule-free message passing

How should this procedure be terminated?

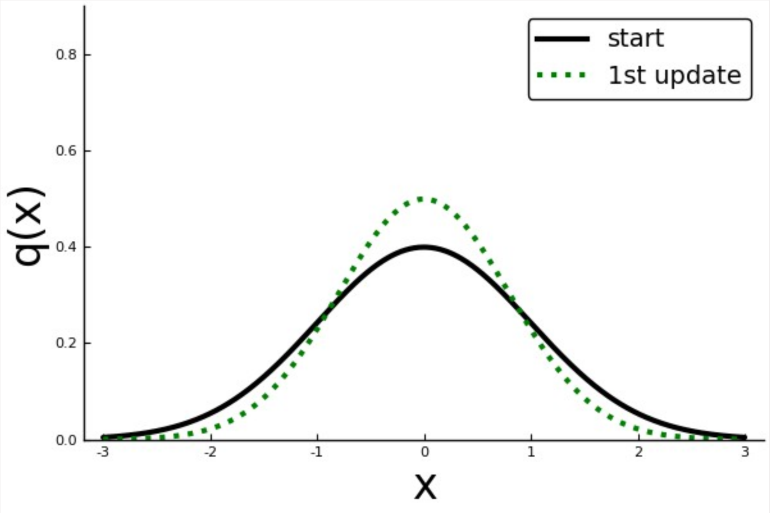
→ Updated beliefs are shaped by priors and likelihoods and will converge.



Schedule-free message passing

How should this procedure be terminated?

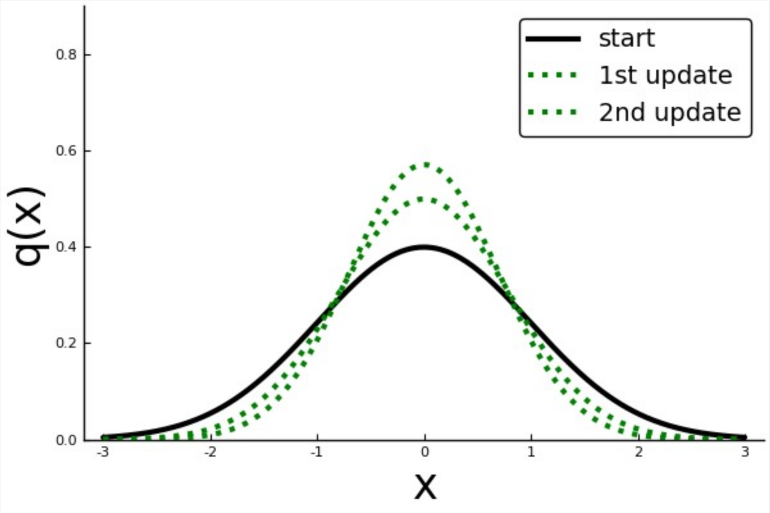
→ Updated beliefs are shaped by priors and likelihoods and will converge.



Schedule-free message passing

How should this procedure be terminated?

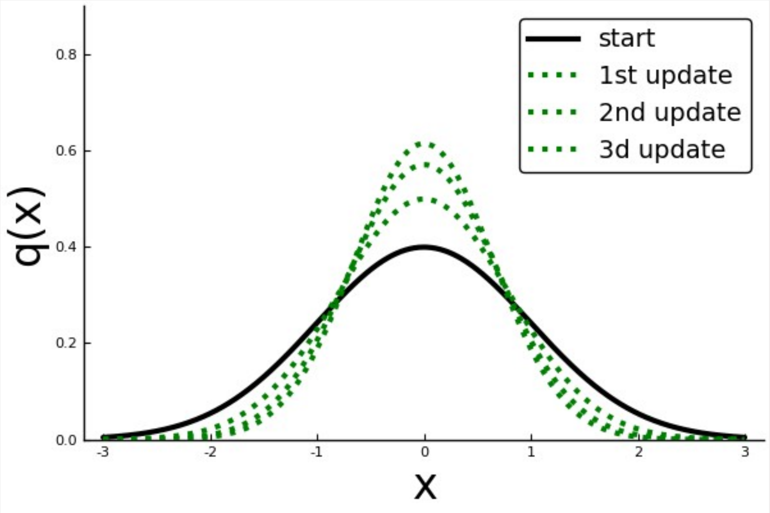
→ Updated beliefs are shaped by priors and likelihoods and will converge.



Schedule-free message passing

How should this procedure be terminated?

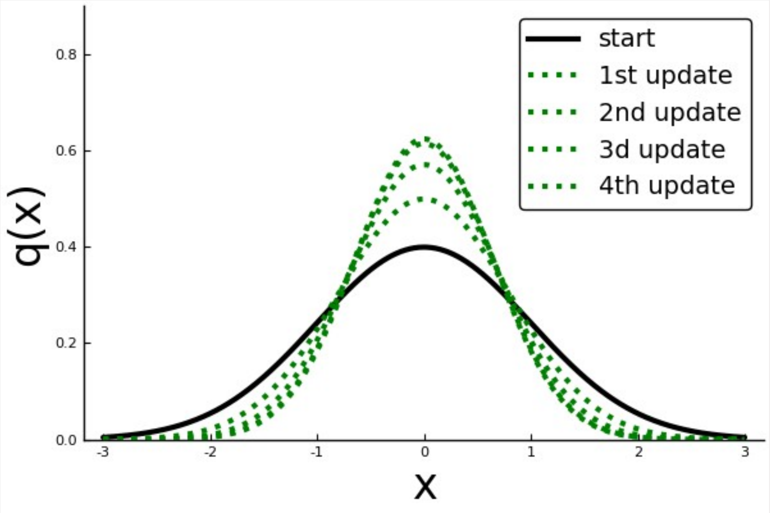
→ Updated beliefs are shaped by priors and likelihoods and will converge.



Schedule-free message passing

How should this procedure be terminated?

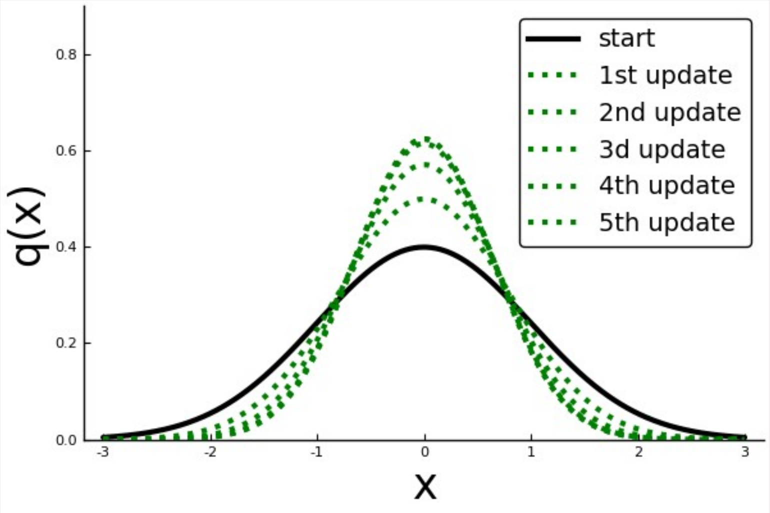
→ Updated beliefs are shaped by priors and likelihoods and will converge.



Schedule-free message passing

How should this procedure be terminated?

→ Updated beliefs are shaped by priors and likelihoods and will converge.

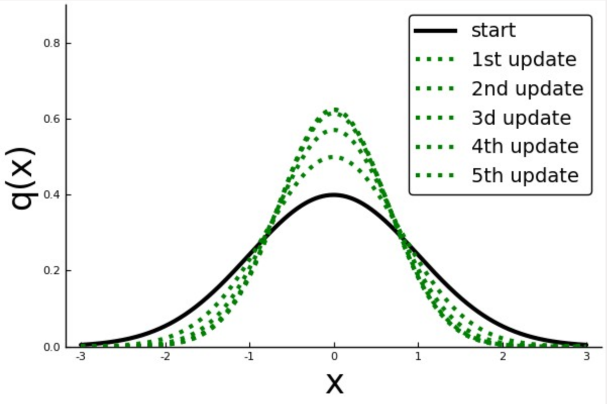


Schedule-free message passing

How should this procedure be terminated?

→ Updated beliefs are shaped by priors and likelihoods and will converge.

We can measure local Free Energy, $F[q(x_t)]$, to check whether beliefs are being updated significantly enough.

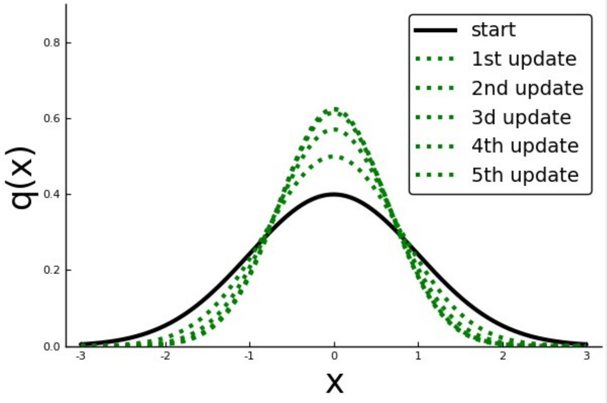


Schedule-free message passing

How should this procedure be terminated?

→ Updated beliefs are shaped by priors and likelihoods and will converge.

We can measure local Free Energy, $F[q(x_t)]$, to check whether beliefs are being updated significantly enough.

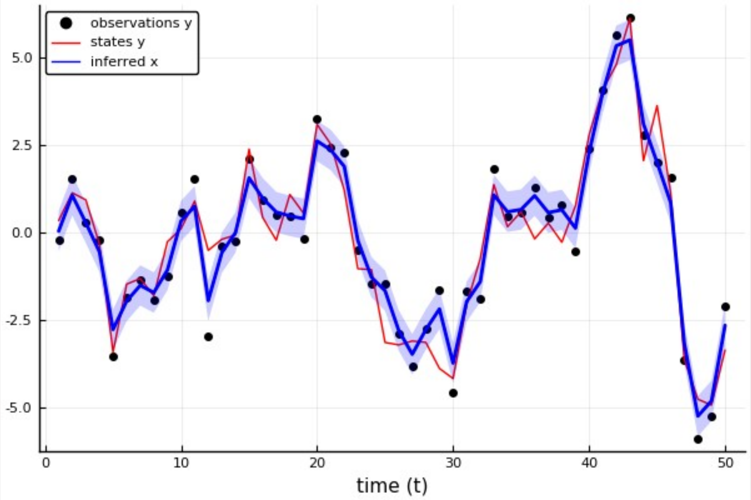


If a message arrives at a factor node from a variable that has not changed *enough*, we can tell the node *not* to react.

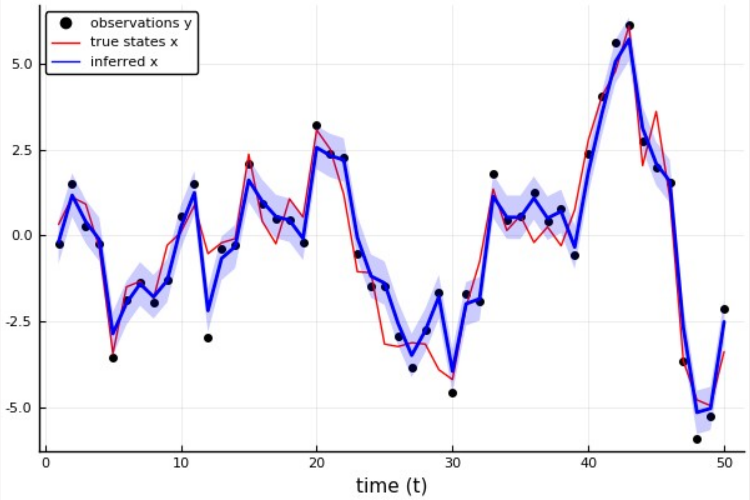
TU/e

Experiment

Comparing a scheduled message passing procedure with the schedule-free algorithm:



scheduled
MSE = 0.454



schedule-free
MSE = 0.611

Open questions

– Will this message passing procedure produce suboptimal estimates?

→ It is entirely possible that suboptimality propagates through the graph.

Open questions

– Will this message passing procedure produce suboptimal estimates?

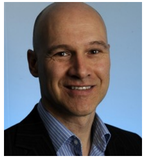
→ It is entirely possible that suboptimality propagates through the graph.

– Implementation can be made more efficient with Functional Reactive Programming.

→ Asychronous message passing is a core feature.

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Thanks to my fellow lab members:


Bert de Vries

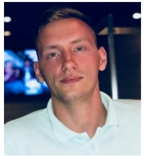

Thijs van de Laar

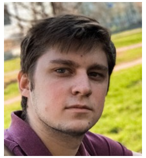

Ivan Bocharov


Ismail Senoz


Semih Akbayrak


Magnus Koudahl


Albert Podusenko


Dmitry Bagaev

Questions?

neuromatch

Bayesian Intelligent Autonomous Systems lab