Variational learning

Solution | Agent-based modelling, Konstanz, 2024

Henri Kauhanen

30 April 2024

First, I repeat the definitions of the custom types `VariationalLearner` and `LearningEnvironment` and the associated functions we defined in the lecture. (Without these definitions, none of what follows will work!)

```
using Plots       # for drawing plots
using StatsBase   # for the sample() function

mutable struct VariationalLearner
  p::Float64
  gamma::Float64
end

struct LearningEnvironment
  P1::Float64
  P12::Float64
  P2::Float64
end

function sample_string(x::LearningEnvironment)
  sample(["S1", "S12", "S2"], Weights([x.P1, x.P12, x.P2]))
end

function pick_grammar(x::VariationalLearner)
  sample(["G1", "G2"], Weights([x.p, 1 - x.p]))
end

function learn!(x::VariationalLearner, y::LearningEnvironment)
  s = sample_string(y)
  g = pick_grammar(x)
```

```
  if g == "G1" && s == "S1"
    x.p = x.p + x.gamma * (1 - x.p)
  elseif g == "G1" && s == "S2"
    x.p = x.p - x.gamma * x.p
  elseif g == "G2" && s == "S2"
    x.p = x.p - x.gamma * x.p
  elseif g == "G2" && s == "S1"
    x.p = x.p + x.gamma * (1 - x.p)
  end

  return x.p
end
```

```
learn! (generic function with 1 method)
```

1. Five learners

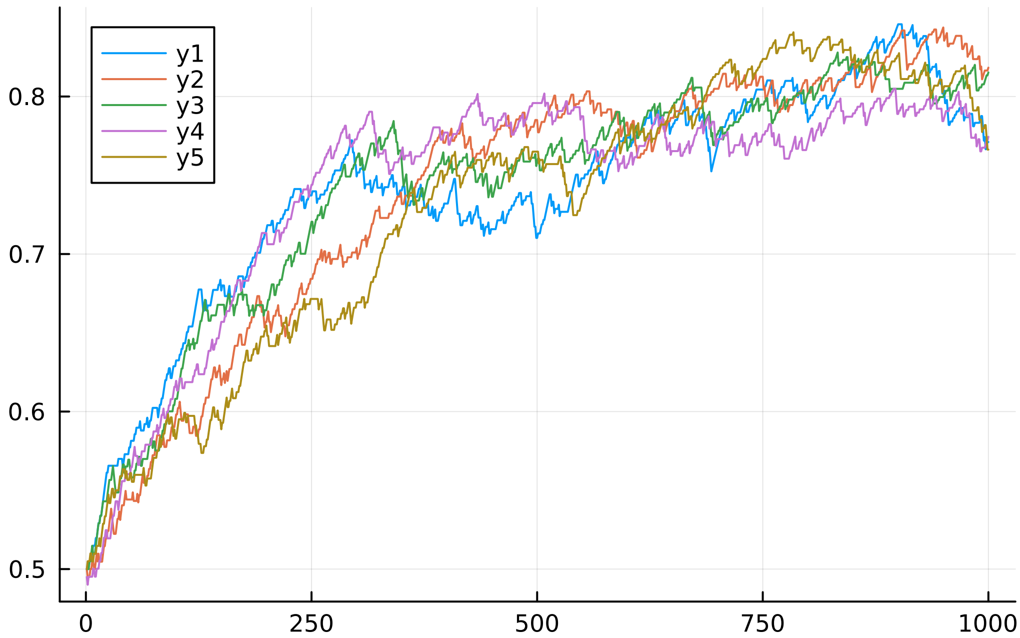
```
# create learning environment
london = LearningEnvironment(0.4, 0.5, 0.1)

# create 5 learners
gamma = 0.01
learner1 = VariationalLearner(0.5, gamma)
learner2 = VariationalLearner(0.5, gamma)
learner3 = VariationalLearner(0.5, gamma)
learner4 = VariationalLearner(0.5, gamma)
learner5 = VariationalLearner(0.5, gamma)

# simulate each learner for 1000 steps
trajectory1 = [learn!(learner1, london) for t in 1:1000]
trajectory2 = [learn!(learner2, london) for t in 1:1000]
trajectory3 = [learn!(learner3, london) for t in 1:1000]
trajectory4 = [learn!(learner4, london) for t in 1:1000]
trajectory5 = [learn!(learner5, london) for t in 1:1000]

# plot
plot(1:1000, trajectory1)
plot!(1:1000, trajectory2)
plot!(1:1000, trajectory3)
```

```
plot!(1:1000, trajectory4)
plot!(1:1000, trajectory5)
```



2. Different learning rates

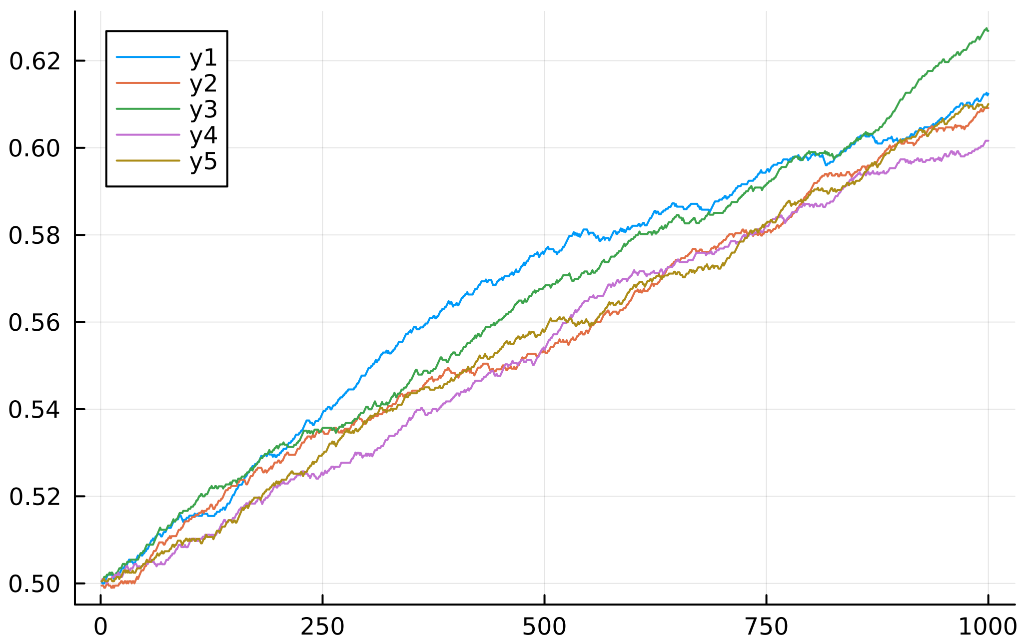
```
# create learning environment
london = LearningEnvironment(0.4, 0.5, 0.1)

# create 5 learners
gamma = 0.001
learner1 = VariationalLearner(0.5, gamma)
learner2 = VariationalLearner(0.5, gamma)
learner3 = VariationalLearner(0.5, gamma)
learner4 = VariationalLearner(0.5, gamma)
learner5 = VariationalLearner(0.5, gamma)

# simulate each learner for 1000 steps
trajectory1 = [learn!(learner1, london) for t in 1:1000]
trajectory2 = [learn!(learner2, london) for t in 1:1000]
trajectory3 = [learn!(learner3, london) for t in 1:1000]
```

```
trajectory4 = [learn!(learner4, london) for t in 1:1000]
trajectory5 = [learn!(learner5, london) for t in 1:1000]

# plot
plot(1:1000, trajectory1)
plot!(1:1000, trajectory2)
plot!(1:1000, trajectory3)
plot!(1:1000, trajectory4)
plot!(1:1000, trajectory5)
```



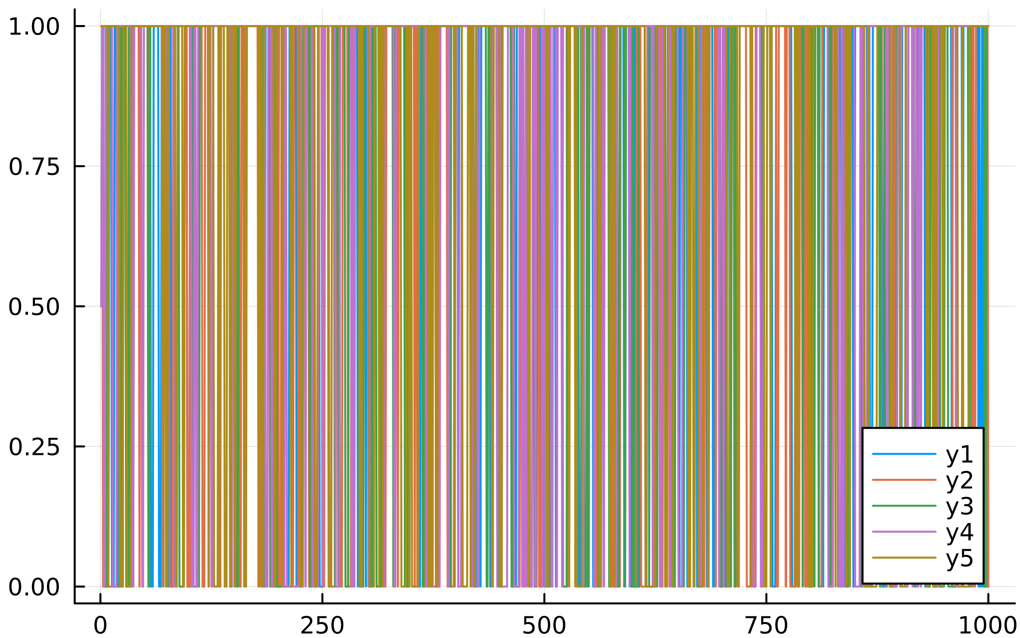

```
# create learning environment
london = LearningEnvironment(0.4, 0.5, 0.1)

# create 5 learners
gamma = 1.0
learner1 = VariationalLearner(0.5, gamma)
learner2 = VariationalLearner(0.5, gamma)
learner3 = VariationalLearner(0.5, gamma)
learner4 = VariationalLearner(0.5, gamma)
learner5 = VariationalLearner(0.5, gamma)

# simulate each learner for 1000 steps
```

```
trajectory1 = [learn!(learner1, london) for t in 1:1000]
trajectory2 = [learn!(learner2, london) for t in 1:1000]
trajectory3 = [learn!(learner3, london) for t in 1:1000]
trajectory4 = [learn!(learner4, london) for t in 1:1000]
trajectory5 = [learn!(learner5, london) for t in 1:1000]

# plot
plot(1:1000, trajectory1)
plot!(1:1000, trajectory2)
plot!(1:1000, trajectory3)
plot!(1:1000, trajectory4)
plot!(1:1000, trajectory5)
```



3. Varying the environment

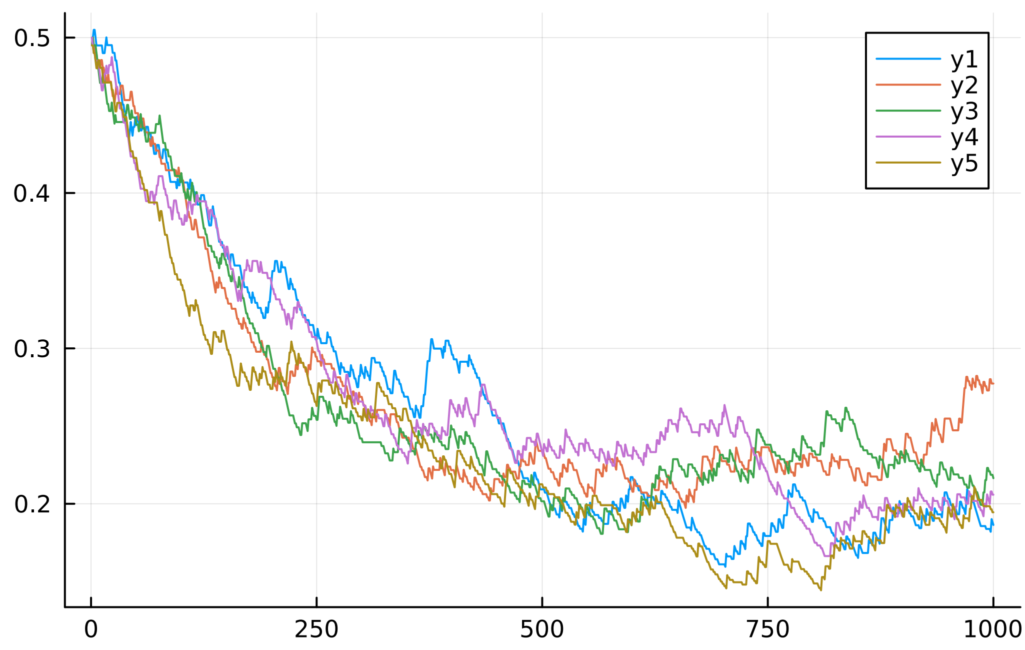
Here I'm only doing this with the value of the learning rate $\gamma = 0.01$; the principle will be exactly the same for any other value.

```
# create learning environment
prague = LearningEnvironment(0.1, 0.5, 0.4)
```

```
# create 5 learners
gamma = 0.01
learner1 = VariationalLearner(0.5, gamma)
learner2 = VariationalLearner(0.5, gamma)
learner3 = VariationalLearner(0.5, gamma)
learner4 = VariationalLearner(0.5, gamma)
learner5 = VariationalLearner(0.5, gamma)

# simulate each learner for 1000 steps
trajectory1 = [learn!(learner1, prague) for t in 1:1000]
trajectory2 = [learn!(learner2, prague) for t in 1:1000]
trajectory3 = [learn!(learner3, prague) for t in 1:1000]
trajectory4 = [learn!(learner4, prague) for t in 1:1000]
trajectory5 = [learn!(learner5, prague) for t in 1:1000]

# plot
plot(1:1000, trajectory1)
plot!(1:1000, trajectory2)
plot!(1:1000, trajectory3)
plot!(1:1000, trajectory4)
plot!(1:1000, trajectory5)
```



4. What does this mean?

The effect of the learning rate parameter γ (`gamma`) is quite straightforward: the smaller the value of this parameter, the slower learning is. In the extreme case $\gamma = 1$, learning is in a sense as "fast" as it can possibly be: each learner switches between probability 1 of using G_1 and probability 0 of using G_1 at parsing failure.

The effect of the the probabilities P1, P12 and P2 of the learning environment is a bit less straightforward. In fact, a formula exists that allows one to predict the expected value of p (probability of using G_1) a variational learner ends up with after a long period of learning, and that formula only depends on the environment's probability parameters. We might look at this in detail later. For now, suffice it to say that the parameters have *some* such effect. More precisely, in the above simulations we see that

- When P1 = 0.4 and P2 = 0.1 (environment `london`), the learners end up with $p \approx 0.8$.
- When P1 = 0.1 and P2 = 0.4 (environment `prague`), the learners end up with $p \approx 0.2$.

5. Prettifying the plots

```
# create learning environment
prague = LearningEnvironment(0.1, 0.5, 0.4)

# create 5 learners
gamma = 0.01
learner1 = VariationalLearner(0.5, gamma)
learner2 = VariationalLearner(0.5, gamma)
learner3 = VariationalLearner(0.5, gamma)
learner4 = VariationalLearner(0.5, gamma)
learner5 = VariationalLearner(0.5, gamma)

# simulate each learner for 1000 steps
trajectory1 = [learn!(learner1, prague) for t in 1:1000]
trajectory2 = [learn!(learner2, prague) for t in 1:1000]
trajectory3 = [learn!(learner3, prague) for t in 1:1000]
trajectory4 = [learn!(learner4, prague) for t in 1:1000]
trajectory5 = [learn!(learner5, prague) for t in 1:1000]

# plot
plot(1:1000, trajectory1, seriestype=:scatter)
plot!(1:1000, trajectory2, seriestype=:scatter)
plot!(1:1000, trajectory3, seriestype=:scatter)
plot!(1:1000, trajectory4, seriestype=:scatter)
```

```
plot!(1:1000, trajectory5, seriestype=:scatter)
xlabel!("learning iteration")
ylabel!("probability of G1")
title!("A variational learning trajectory")
```