Computational Cognitive Science 2026/27

Week 8

Frank Keller Chris Lucas

Solutions for Tutorial 5: Active learning

This tutorial provides some hands-on exposure to concepts from the active learning lecture.

Imagine you come across a curiosity shop that has four devices for sale at £100 each. Each device can automatically create a delicious cake when one pulls its lever, but the devices vary in their reliability:

- one device never works,
- one device works exactly 20 percent of the time,
- one works 30 percent of the time, and
- one works 75 percent of the time.

We don't know which device is which, but the shopkeeper will let us use the devices a few times to choose one.

Our task is to find out which machine is which.

Let's take an information-theoretic approach to this active learning problem.

Here are some helper functions.

```
plogp <- function(p) {
  # bits, not nats
  if(p==0) 0 else p*log2(p) # Consider why we can't just use p*log2(p).
}

# entropy for a categorical random variable
cat_entropy <- function(catv) {
  -sum(sapply(catv,plogp))
}
```

Let's define our hypothesis space as all of the different cake probabilities the machines before us might have from left to right. Fore example `c(0,.2,.3,.75)` means the machines increase in their probabilities from left to right, and `c(.75,.3,.2,0)` means they decrease from left to right.

Question 1 How does the code below define that hypothesis space?

```
require(combinat) # all list permutations
```

```
## Loading required package: combinat
```

```
##
## Attaching package: 'combinat'
```

```
## The following object is masked from 'package:utils':
##
##     combn
```

```
probs <- c(0,.2,.3,.75)
hyps <- permn(probs)
```

Solution: *Our hypotheses capture which machine is assigned which cake probability by exhaustively permuting a single probability list, yielding 4!=24 hypotheses.*

Question 2 What is our entropy, assuming we know nothing about the order of the machines?

Solution:

```
prior <- rep(1/24,24) # uniform prior over all hypotheses
cat_entropy(prior)
```

```
## [1] 4.584963
```

Question 3 Suppose we turn the lever on the first machine and get nothing. What probability should we now assign our different hypotheses? Edit the code below.

Solution::

```
update_probs <- function(prior,choice,outcome) {
  # prior (vector): one prior probability per hypothesis
  # choice (int): index of the machine (1-4) on which the lever was turned
  # outcome (int): whether we get cake (1) or not (0)

  likelihood <- function(this_hyp,this_choice,this_outcome) {
    dbinom(this_outcome, 1, this_hyp[this_choice])
    # alternative implementation:
    # p <- this_hyp[this_choice]
    # if (this_outcome == 1) p else (1 - p)
  }

  likes <- sapply(hyps,function(h) {likelihood(h,choice,outcome)})  # likelihoods
  unnp <- likes*prior # unnormalized posterior
  z <- sum(unnp) # normalization constant
  unnp/z
}

post <- update_probs(prior,choice=1,outcome=0)
print(post)
```

```
##  [1] 0.06060606 0.06060606 0.06060606 0.01515152 0.01515152 0.06060606
##  [7] 0.06060606 0.06060606 0.04242424 0.04242424 0.04242424 0.01515152
## [13] 0.01515152 0.04242424 0.04242424 0.04242424 0.04848485 0.04848485
## [19] 0.04848485 0.01515152 0.01515152 0.04848485 0.04848485 0.04848485
```

Question 4 What is our new entropy? Did we learn much? Why or why not?

Solution:

```
print(cat_entropy(post))
```

```
## [1] 4.450846
```

```
print(cat_entropy(prior) - cat_entropy(post))
```

```
## [1] 0.1341161
```

We didn't learn very much! It makes sense, because 3/4 of our machines are unlikely to produce a cake, so most hypotheses are quite consistent with what we've seen.

Question 5 What is the expected probability getting a cake if we choose machine 2 next?

Solution:

```
# First we want the probability of each outcome in each hyp
# Then we take the weighted sum over hypothesis probabilities (Bayesian hypothesis averaging)
expected_outcome <- function(i,ph) {
  # ph (vector): posterior, one prior probability per hypothesis
  # i (int): machine index
  sum(sapply(hyps,function(x) {x[i]})*ph)
}
print(expected_outcome(2,post))
```

```
## [1] 0.3490909
```

Question 6 What is the expected entropy if we choose machine 2 next?

Solution: *We take the expectation over possible outcomes.*

```
expected_entropy <- function(choice,p_hyps) {
  pwin <- expected_outcome(choice,p_hyps)
  if_win <- update_probs(p_hyps,choice,1)
  if_lose <- update_probs(p_hyps,choice,0)
  cat_entropy(if_win)*pwin+cat_entropy(if_lose)*(1-pwin)
}
print(expected_entropy(2,post))
```

```
## [1] 4.15306
```

Question 7 Suppose we do choose machine 2 and it yields a cake. What is our current entropy and what are the expected entropies after each of our next possible choices?

Solution:

```
post2 <- update_probs(post,2,1)
print(cat_entropy(post2))
```

```
## [1] 3.781135
```

```
sapply(c(1,2,3,4),function(x) {expected_entropy(x,post2)})
```

```
## [1] 3.595719 3.613545 3.537257 3.537257
```

Question 8 What does this imply we should do next?

Solution: *It implies we should try a new arm – either 3 or 4.*

Question 9 Is this at all counter-intuitive? What might we do differently if we are only interested in knowing which is the most reliable machine?

Solution: *It seems odd to learn about a machine that is probably bad, if we want to learn which machine is good – our learning objective isn't matched to our goals. We can instead focus on the entropy of the random variable that matters: The identity of the machine with the highest probability.*

```r
is_75_ent <- function(probs) {
  oh_75 <- sapply(hyps,function(x) {sapply(x,function(y) {if(y==.75) 1.0 else 0})})
  pr <- matrix(rep(probs,4),ncol=4)
  p75<-colSums(t(oh_75) * pr)
  cat_entropy(p75)
}

ee_75 <- function(choice,prior) {
  pwin <- expected_outcome(choice,prior)
  if_win <- update_probs(prior,choice,1)
  if_lose <- update_probs(prior,choice,0)
  is_75_ent(if_win)*pwin+is_75_ent(if_lose)*(1-pwin)
}

ents_75 <- sapply(c(1,2,3,4),function(x) {ee_75(x,post2)})
ents_75
```

```
## [1] 1.367541 1.263344 1.280055 1.280055
```