# PS3

April 10, 2024

## 1 Problem set 3 – Neural Amortized Inference

Save the completed notebook as a pdf file and submit the pdf file to Canvas. Follow examples in the `Data-Driven Proposals in Gen` notebook (Section 5) covered in the lecture.

[71]:
```julia
using Random
using Gen
using Plots
using DelimitedFiles
using JLD2
```

### 1.1 Perception in a rectangle world

In this pset, you will develop a perception system that operates in a two-dimensional grayscale world where all objects are axis-aligned rectangular frames (i.e., unfilled rectangles) and there is just one such object in a given scene. An example scene in this world is illustrated below. Given such an input, the perception system should provide a posterior over where the object is, its constrast and size.

#### 1.1.1 Q 1A [2.5 pts]

Your first task is to write a generative model of this process. You will do this in the generative function `two_d_world`, below.

Here are the basic assumptions your generative model should reflect.

- Assume that the world size is 10x10 pixels.
- Assume that there is one object in each scene in this world.
- An object's position, in particular its bottom-left corner, can be anywhere in the world. So in a lot of the scenes, the object will only be partially visible.
- Each dimension of the objects in this world (width and height) follow a uniform distribution between 3 to 7 pixels. Notice that a rectangle cannot have a negative dimension.
- An object's overall brightness can vary between 0.1 and 1 with a uniform distribution, where the background brigthness is set to 0.
- Finally, assume that the observations are corrupted by some small Gaussian noise (mean, std = 0.05), i.e., adding a Gaussian noise to the brightness of each pixel.

To make our variational approximation less of a pain, we recommend setting up each of your priors to be uniform distributions `[0, 1]`, then scaling them before "rendering" your object.

We provide examples for two of the relevant random variables – the y-coordinate of the south-west (bottom-left) of the object and the height of the object.

```
# draw where the object's y coordinate will be
SW_row ~ uniform(0, 1)
# draw the height of the object
h ~ uniform(0, 1)

# scale the y-coordinate so that it is an integer (we will use this to index into a Matrix of
scaled_SW_row = ceil(Int64, SW_row * 10)
# scale the height so that it lies between 3 and 7 and is an integer
scaled_h = round(Int64, h * 4 + 3)
```

All random variables: x, y coordinate of the left corner, height, width, overall brightness of the object, brightness of each pixel (after considering whether it belongs to the object as well as the gaussian noise). Hint: brightness of each pixel can be sampled from `normal(mean brightness conditioned on whether it is part of the object or the background, 0.05)`.

In total, there are 100 (all pixels' brightness) + 5 (location, size, object brightness) random variables.

The generative function should return a 10x10 matrix as the rendered image.

[72]:
```
N_COLS = 10
N_ROWS = 10

@gen function two_d_world()
    # sample unscaled y coordinate (bottom-left corner)
    SW_row ~ uniform(0, 1)

    # sample unscaled x coordinate (bottom-left corner)
    SW_col ~ uniform(0, 1)

    # sample unscaled height
    h ~ uniform(0, 1)

    # sample unscaled width
    w ~ uniform(0, 1)

    # rescale x, y, height, and width to integers
    scaled_SW_row = ceil(Int64, SW_row * 10)
    scaled_SW_col = ceil(Int64, SW_col * 10)
    scaled_h = round(Int64, h * 4 + 3)
    scaled_w = round(Int64, w * 4 + 3)

    # sample overall brightness for the object
    b ~ uniform(0.1, 1)

    # create a blank image
    image = Matrix{Float64}(undef, N_ROWS, N_COLS)

    # sample brightness for each pixel with noise
    for row_id in 1:N_ROWS
        for col_id in 1:N_COLS
            if row_id  scaled_SW_row && row_id  (scaled_SW_row + scaled_h) &&␣
↪(col_id == scaled_SW_col || col_id == (scaled_SW_col + scaled_w))
                mean_brightness = b
            elseif col_id  scaled_SW_col && col_id  (scaled_SW_col + scaled_w)␣
↪&& (row_id == scaled_SW_row || row_id == (scaled_SW_row + scaled_h))
                mean_brightness = b
            else
                mean_brightness = 0  # background
            end

            image[row_id, col_id] = {:image => row_id => col_id => :brightness}␣
↪~ normal(mean_brightness, 0.05)
        end
    end

    # return image
    image
end
```

```
DynamicDSLFunction{Any}(Dict{Symbol, Any}(), Dict{Symbol, Any}(), Type[], false,␣
↪Union{Nothing, Some{Any}}[], var"##two_d_world#243", Bool[], false)
```

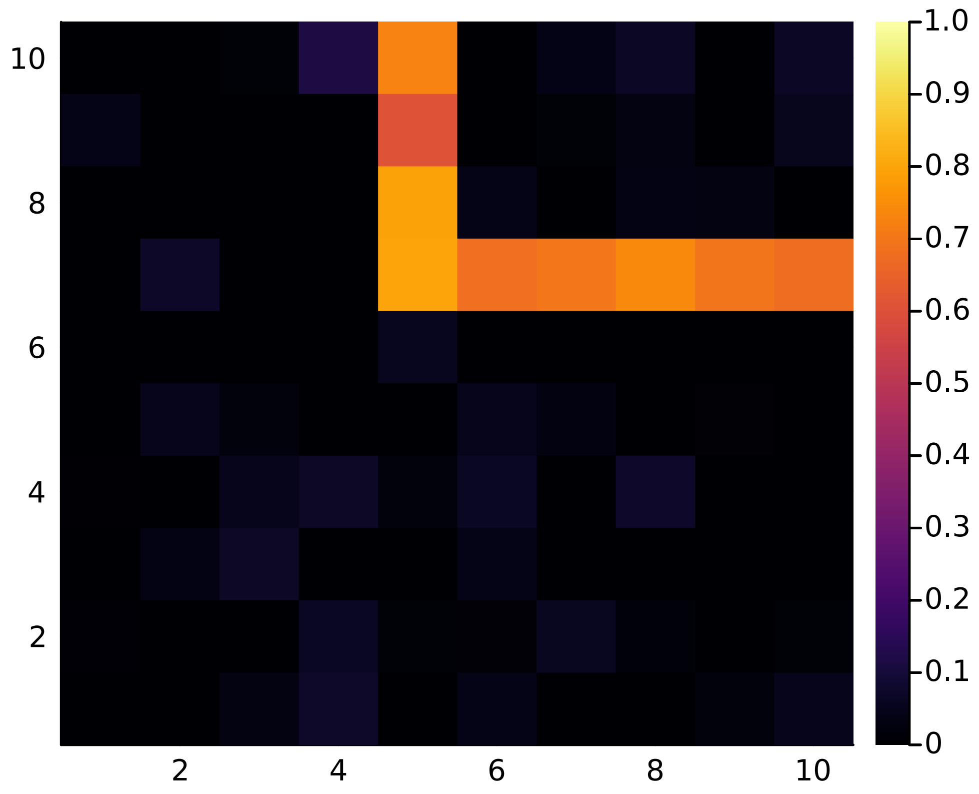
Below is a function to visualize a given draw from your generative model.

[73]:
```
function visualize(input::Matrix{<:Real})
    heatmap(input, clim=(0,1), thickness_scaling=3.5, size=(1600, 1300),␣
↪aspect=:equal)
end
```

```
visualize (generic function with 1 method)
```

Draw a sample from your generative model and visualize it (using the `visualize` function above).

[74]:
```
Random.seed!(42)
visualize(two_d_world())
```



### 1.1.2 Q 1B [2 pts]

Now implement an amortized variational approximation of this generative model, parametrized with a deep neural network conditioning this approximation to input observations. You will do this in the generative function `neural_amortized_inference`, below.

Assume that the neural network takes as input a vector – so, the observations should be flattened to vectors (from 2D matrices). Your network architecture should be rather simple: one hidden layer and one output layer. The hidden layer should be activated with a `tanh` non-linearity (provided in the code block below).

The output layer should consist of all of the variational family parameters.

As for your variational approximation, for a random variable `x ~ uniform(0,1)` in your generative model (`two_d_world`), a reasonable choice would be `x ~ beta(shape, scale)`. Your neural network would be outputting the parameters of the beta, but you'd need to be careful to ensure that these parameters are possitive. E.g., you can use exp() as the activation function for the output layer: `exp.(x)`.

HINT: There are 5 (location, size, object brightness) random variables we want to infer based on an observed image. So there should be 10 parameters from the output layer of the neural network

(2 parameters for the beta distribution for each random variable).

[75]:
```
 (x) = tanh.(x)

@gen function neural_amortized_inference(input::Vector{Float64})
    @param W1::Matrix{Float64}
    @param b1::Vector{Float64}

    @param W2::Matrix{Float64}
    @param b2::Vector{Float64}

    # non-linear hidden layer
    hidden_layer =  (W1 * input + b1)

    # output layer
    output_layer = exp.(W2 * hidden_layer + b2)

    # feature extraction
    SW_row_shape = output_layer[1]
    SW_row_scale = output_layer[2]

    SW_col_shape = output_layer[3]
    SW_col_scale = output_layer[4]

    w_shape = output_layer[5]
    w_scale = output_layer[6]

    h_shape = output_layer[7]
    h_scale = output_layer[8]

    b_shape = output_layer[9]
    b_scale = output_layer[10]

    SW_row ~ beta(SW_row_shape, SW_row_scale)
    SW_col ~ beta(SW_col_shape, SW_col_scale)
    w ~ beta(w_shape, w_scale)
    h ~ beta(h_shape, h_scale)
    b ~ beta(b_shape, b_scale)

    return nothing
end
```

```
DynamicDSLFunction{Any}(Dict{Symbol, Any}(), Dict{Symbol, Any}(),␣
 ↪Type[Vector{Float64}], false, Union{Nothing, Some{Any}}[nothing],␣
 ↪var"##neural_amortized_inference#245", Bool[0], false)
```

### 1.1.3 Q 1C [1 pts]

Next create a data generator function, called `data_generator`. Notice that this function takes no arguments. In each call, it will simulate the generative model of our world once. This will yield a pair of input and output for training the neural network based estimator `neural_amortized_inference`.

```
[76]: function data_generator()
          tr = Gen.simulate(two_d_world, ())

          # record the "observations" (inputs to the NN model, i.e., brightness of␣
       ↪each pixel)
          obs_matrix = Matrix{Float64}(undef, N_ROWS, N_COLS)

          for row_id in 1:N_ROWS
              for col_id in 1:N_COLS
                  obs_matrix[row_id, col_id] = tr[:image => row_id => col_id => :
       ↪brightness]
              end
          end

          obs = vec(obs_matrix)

          # record the random choices of the 5 latent variables (outputs of the NN␣
       ↪model)
          constraints = Gen.choicemap()
          constraints[:SW_row] = tr[:SW_row]
          constraints[:SW_col] = tr[:SW_col]
          constraints[:w] = tr[:w]
          constraints[:h] = tr[:h]
          constraints[:b] = tr[:b]

          return ((obs,), constraints)
      end
```

```
data_generator (generic function with 1 method)
```

### 1.1.4 Q 1D [1.5 pts]

Initialize the `params` in the `neural_amortized_inference`. You will have to pay attention to your dimensions.

Choose the dimensionality of the hidden layer to be 200. Use the `init_weight` function (provided in the code block below) to initialize your weight matrices.

```
[77]: Random.seed!(42)
      # a function for randomly initializing the weight matrices
      init_weight(shape...) = (1. / sqrt(shape[2])) * randn(shape...)

      # input, hidden, and output dimensions of the network
```

```
input_dim = 100
hidden_dim = 200
output_dim = 10

# create and initialize W1 and W2
init_W1 = init_weight(hidden_dim, input_dim)
init_W2 = init_weight(output_dim, hidden_dim)

# set params of the data-driven proposal function
init_param!(neural_amortized_inference, :W1, init_W1)
init_param!(neural_amortized_inference, :b1, zeros(hidden_dim))
init_param!(neural_amortized_inference, :W2, init_W2)
init_param!(neural_amortized_inference, :b2, zeros(output_dim));
```

### 1.1.5 Q 1E [1 pt]

Create an optimizer for updating the weights using `Gen.FixedStepGradientDescent` with a learning rate of `1e-5`.

Train your amortized estimator using this optimizer using `Gen.train!`.

Use the following arguments for the `train!` function:

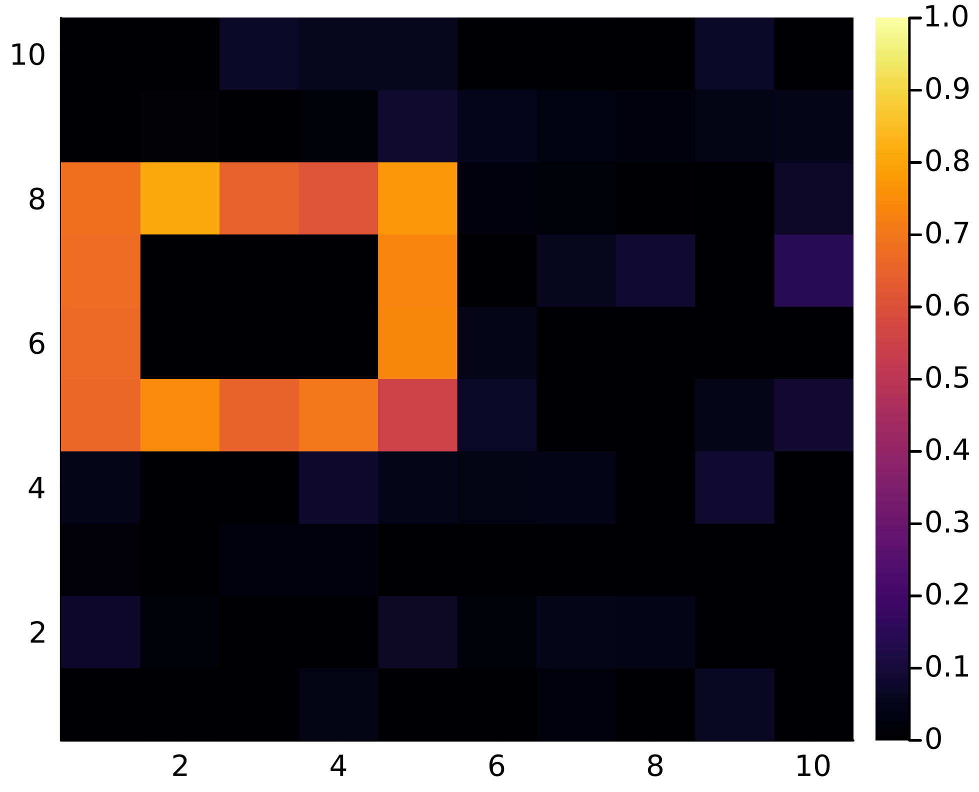
```
num_epoch=200
epoch_size=1000
num_minibatch=100
minibatch_size=10
evaluation_size=100
verbose=true
```

[78]:
```
# get a gradient-based optimizer and train!
# your code here
update = Gen.ParamUpdate(Gen.FixedStepGradientDescent(1e-5),␣
↪neural_amortized_inference);
Gen.train!(neural_amortized_inference, data_generator, update, num_epoch=200,␣
↪epoch_size=1000, num_minibatch=100, minibatch_size=10, evaluation_size=100,␣
↪verbose=true)
# save trained parameters
let data = Dict()
    for name in [:W1, :b1, :W2, :b2]
        data[(:param, name)] = Gen.get_param(neural_amortized_inference, name)
    end
    save("neural_amortized_inference_trained.jld2", "data", data)
end
```

```
epoch 1: generating 1000 training examples…
epoch 1: training using 100 minibatches of size 10…
epoch 1: evaluating on 100 examples…
epoch 1: est. objective value: 0.0
epoch 2: generating 1000 training examples…
epoch 2: training using 100 minibatches of size 10…
epoch 2: evaluating on 100 examples…
epoch 2: est. objective value: 0.0
epoch 3: generating 1000 training examples…
epoch 3: training using 100 minibatches of size 10…
epoch 3: evaluating on 100 examples…
epoch 3: est. objective value: 0.0
epoch 4: generating 1000 training examples…
epoch 4: training using 100 minibatches of size 10…
epoch 4: evaluating on 100 examples…
epoch 4: est. objective value: 0.0
epoch 5: generating 1000 training examples…
epoch 5: training using 100 minibatches of size 10…
epoch 5: evaluating on 100 examples…
epoch 5: est. objective value: 0.0
epoch 6: generating 1000 training examples…
epoch 6: training using 100 minibatches of size 10…
epoch 6: evaluating on 100 examples…
epoch 6: est. objective value: 0.0
epoch 7: generating 1000 training examples…
epoch 7: training using 100 minibatches of size 10…
epoch 7: evaluating on 100 examples…
epoch 7: est. objective value: 0.0
epoch 8: generating 1000 training examples…
epoch 8: training using 100 minibatches of size 10…
epoch 8: evaluating on 100 examples…
epoch 8: est. objective value: 0.0
epoch 9: generating 1000 training examples…
epoch 9: training using 100 minibatches of size 10…
epoch 9: evaluating on 100 examples…
epoch 9: est. objective value: 0.0
epoch 10: generating 1000 training examples…
epoch 10: training using 100 minibatches of size 10…
epoch 10: evaluating on 100 examples…
epoch 10: est. objective value: 0.0
epoch 11: generating 1000 training examples…
epoch 11: training using 100 minibatches of size 10…
epoch 11: evaluating on 100 examples…
epoch 11: est. objective value: 0.0
epoch 12: generating 1000 training examples…
epoch 12: training using 100 minibatches of size 10…
epoch 12: evaluating on 100 examples…
epoch 12: est. objective value: 0.0
epoch 13: generating 1000 training examples…
epoch 13: training using 100 minibatches of size 10…
epoch 13: evaluating on 100 examples…
epoch 13: est. objective value: 0.0
epoch 14: generating 1000 training examples…
epoch 14: training using 100 minibatches of size 10…
epoch 14: evaluating on 100 examples…
epoch 14: est. objective value: 0.0
epoch 15: generating 1000 training examples…
epoch 15: training using 100 minibatches of size 10…
epoch 15: evaluating on 100 examples…
epoch 15: est. objective value: 0.0
epoch 16: generating 1000 training examples…
epoch 16: training using 100 minibatches of size 10…
epoch 16: evaluating on 100 examples…
epoch 16: est. objective value: 0.0
epoch 17: generating 1000 training examples…
epoch 17: training using 100 minibatches of size 10…
epoch 17: evaluating on 100 examples…
epoch 17: est. objective value: 0.0
epoch 18: generating 1000 training examples…
epoch 18: training using 100 minibatches of size 10…
epoch 18: evaluating on 100 examples…
epoch 18: est. objective value: 0.0
epoch 19: generating 1000 training examples…
epoch 19: training using 100 minibatches of size 10…
epoch 19: evaluating on 100 examples…
epoch 19: est. objective value: 0.0
epoch 20: generating 1000 training examples…
epoch 20: training using 100 minibatches of size 10…
epoch 20: evaluating on 100 examples…
epoch 20: est. objective value: 0.0
epoch 21: generating 1000 training examples…
epoch 21: training using 100 minibatches of size 10…
epoch 21: evaluating on 100 examples…
epoch 21: est. objective value: 0.0
epoch 22: generating 1000 training examples…
epoch 22: training using 100 minibatches of size 10…
epoch 22: evaluating on 100 examples…
epoch 22: est. objective value: 0.0
epoch 23: generating 1000 training examples…
epoch 23: training using 100 minibatches of size 10…
epoch 23: evaluating on 100 examples…
epoch 23: est. objective value: 0.0
epoch 24: generating 1000 training examples…
epoch 24: training using 100 minibatches of size 10…
epoch 24: evaluating on 100 examples…
epoch 24: est. objective value: 0.0
epoch 25: generating 1000 training examples…
epoch 25: training using 100 minibatches of size 10…
epoch 25: evaluating on 100 examples…
epoch 25: est. objective value: 0.0
epoch 26: generating 1000 training examples…
epoch 26: training using 100 minibatches of size 10…
epoch 26: evaluating on 100 examples…
epoch 26: est. objective value: 0.0
epoch 27: generating 1000 training examples…
epoch 27: training using 100 minibatches of size 10…
epoch 27: evaluating on 100 examples…
epoch 27: est. objective value: 0.0
epoch 28: generating 1000 training examples…
epoch 28: training using 100 minibatches of size 10…
epoch 28: evaluating on 100 examples…
epoch 28: est. objective value: 0.0
epoch 29: generating 1000 training examples…
epoch 29: training using 100 minibatches of size 10…
epoch 29: evaluating on 100 examples…
epoch 29: est. objective value: 0.0
epoch 30: generating 1000 training examples…
epoch 30: training using 100 minibatches of size 10…
epoch 30: evaluating on 100 examples…
epoch 30: est. objective value: 0.0
epoch 31: generating 1000 training examples…
epoch 31: training using 100 minibatches of size 10…
epoch 31: evaluating on 100 examples…
epoch 31: est. objective value: 0.0
epoch 32: generating 1000 training examples…
epoch 32: training using 100 minibatches of size 10…
epoch 32: evaluating on 100 examples…
epoch 32: est. objective value: 0.0
epoch 33: generating 1000 training examples…
epoch 33: training using 100 minibatches of size 10…
epoch 33: evaluating on 100 examples…
epoch 33: est. objective value: 0.0
epoch 34: generating 1000 training examples…
epoch 34: training using 100 minibatches of size 10…
epoch 34: evaluating on 100 examples…
epoch 34: est. objective value: 0.0
epoch 35: generating 1000 training examples…
epoch 35: training using 100 minibatches of size 10…
epoch 35: evaluating on 100 examples…
epoch 35: est. objective value: 0.0
epoch 36: generating 1000 training examples…
epoch 36: training using 100 minibatches of size 10…
epoch 36: evaluating on 100 examples…
epoch 36: est. objective value: 0.0
epoch 37: generating 1000 training examples…
epoch 37: training using 100 minibatches of size 10…
epoch 37: evaluating on 100 examples…
epoch 37: est. objective value: 0.0
epoch 38: generating 1000 training examples…
epoch 38: training using 100 minibatches of size 10…
epoch 38: evaluating on 100 examples…
epoch 38: est. objective value: 0.0
epoch 39: generating 1000 training examples…
epoch 39: training using 100 minibatches of size 10…
epoch 39: evaluating on 100 examples…
epoch 39: est. objective value: 0.0
epoch 40: generating 1000 training examples…
epoch 40: training using 100 minibatches of size 10…
epoch 40: evaluating on 100 examples…
epoch 40: est. objective value: 0.0
epoch 41: generating 1000 training examples…
epoch 41: training using 100 minibatches of size 10…
epoch 41: evaluating on 100 examples…
epoch 41: est. objective value: 0.0
epoch 42: generating 1000 training examples…
epoch 42: training using 100 minibatches of size 10…
epoch 42: evaluating on 100 examples…
epoch 42: est. objective value: 0.0
epoch 43: generating 1000 training examples…
epoch 43: training using 100 minibatches of size 10…
epoch 43: evaluating on 100 examples…
epoch 43: est. objective value: 0.0
epoch 44: generating 1000 training examples…
epoch 44: training using 100 minibatches of size 10…
epoch 44: evaluating on 100 examples…
epoch 44: est. objective value: 0.0
epoch 45: generating 1000 training examples…
epoch 45: training using 100 minibatches of size 10…
epoch 45: evaluating on 100 examples…
epoch 45: est. objective value: 0.0
epoch 46: generating 1000 training examples…
epoch 46: training using 100 minibatches of size 10…
epoch 46: evaluating on 100 examples…
epoch 46: est. objective value: 0.0
epoch 47: generating 1000 training examples…
epoch 47: training using 100 minibatches of size 10…
epoch 47: evaluating on 100 examples…
epoch 47: est. objective value: 0.0
epoch 48: generating 1000 training examples…
epoch 48: training using 100 minibatches of size 10…
epoch 48: evaluating on 100 examples…
epoch 48: est. objective value: 0.0
epoch 49: generating 1000 training examples…
epoch 49: training using 100 minibatches of size 10…
epoch 49: evaluating on 100 examples…
epoch 49: est. objective value: 0.0
epoch 50: generating 1000 training examples…
epoch 50: training using 100 minibatches of size 10…
epoch 50: evaluating on 100 examples…
epoch 50: est. objective value: 0.0
epoch 51: generating 1000 training examples…
epoch 51: training using 100 minibatches of size 10…
epoch 51: evaluating on 100 examples…
epoch 51: est. objective value: 0.0
epoch 52: generating 1000 training examples…
epoch 52: training using 100 minibatches of size 10…
epoch 52: evaluating on 100 examples…
epoch 52: est. objective value: 0.0
epoch 53: generating 1000 training examples…
epoch 53: training using 100 minibatches of size 10…
epoch 53: evaluating on 100 examples…
epoch 53: est. objective value: 0.0
epoch 54: generating 1000 training examples…
epoch 54: training using 100 minibatches of size 10…
epoch 54: evaluating on 100 examples…
epoch 54: est. objective value: 0.0
epoch 55: generating 1000 training examples…
epoch 55: training using 100 minibatches of size 10…
epoch 55: evaluating on 100 examples…
epoch 55: est. objective value: 0.0
epoch 56: generating 1000 training examples…
epoch 56: training using 100 minibatches of size 10…
epoch 56: evaluating on 100 examples…
epoch 56: est. objective value: 0.0
epoch 57: generating 1000 training examples…
epoch 57: training using 100 minibatches of size 10…
epoch 57: evaluating on 100 examples…
epoch 57: est. objective value: 0.0
epoch 58: generating 1000 training examples…
epoch 58: training using 100 minibatches of size 10…
epoch 58: evaluating on 100 examples…
epoch 58: est. objective value: 0.0
epoch 59: generating 1000 training examples…
epoch 59: training using 100 minibatches of size 10…
epoch 59: evaluating on 100 examples…
epoch 59: est. objective value: 0.0
epoch 60: generating 1000 training examples…
epoch 60: training using 100 minibatches of size 10…
epoch 60: evaluating on 100 examples…
epoch 60: est. objective value: 0.0
epoch 61: generating 1000 training examples…
epoch 61: training using 100 minibatches of size 10…
epoch 61: evaluating on 100 examples…
epoch 61: est. objective value: 0.0
epoch 62: generating 1000 training examples…
epoch 62: training using 100 minibatches of size 10…
epoch 62: evaluating on 100 examples…
epoch 62: est. objective value: 0.0
epoch 63: generating 1000 training examples…
epoch 63: training using 100 minibatches of size 10…
epoch 63: evaluating on 100 examples…
epoch 63: est. objective value: 0.0
epoch 64: generating 1000 training examples…
epoch 64: training using 100 minibatches of size 10…
epoch 64: evaluating on 100 examples…
epoch 64: est. objective value: 0.0
epoch 65: generating 1000 training examples…
epoch 65: training using 100 minibatches of size 10…
epoch 65: evaluating on 100 examples…
epoch 65: est. objective value: 0.0
epoch 66: generating 1000 training examples…
epoch 66: training using 100 minibatches of size 10…
epoch 66: evaluating on 100 examples…
epoch 66: est. objective value: 0.0
epoch 67: generating 1000 training examples…
epoch 67: training using 100 minibatches of size 10…
epoch 67: evaluating on 100 examples…
epoch 67: est. objective value: 0.0
epoch 68: generating 1000 training examples…
epoch 68: training using 100 minibatches of size 10…
epoch 68: evaluating on 100 examples…
epoch 68: est. objective value: 0.0
epoch 69: generating 1000 training examples…
epoch 69: training using 100 minibatches of size 10…
epoch 69: evaluating on 100 examples…
epoch 69: est. objective value: 0.0
epoch 70: generating 1000 training examples…
epoch 70: training using 100 minibatches of size 10…
epoch 70: evaluating on 100 examples…
epoch 70: est. objective value: 0.0
epoch 71: generating 1000 training examples…
epoch 71: training using 100 minibatches of size 10…
epoch 71: evaluating on 100 examples…
epoch 71: est. objective value: 0.0
epoch 72: generating 1000 training examples…
epoch 72: training using 100 minibatches of size 10…
epoch 72: evaluating on 100 examples…
epoch 72: est. objective value: 0.0
epoch 73: generating 1000 training examples…
epoch 73: training using 100 minibatches of size 10…
epoch 73: evaluating on 100 examples…
epoch 73: est. objective value: 0.0
epoch 74: generating 1000 training examples…
epoch 74: training using 100 minibatches of size 10…
epoch 74: evaluating on 100 examples…
epoch 74: est. objective value: 0.0
epoch 75: generating 1000 training examples…
epoch 75: training using 100 minibatches of size 10…
epoch 75: evaluating on 100 examples…
epoch 75: est. objective value: 0.0
epoch 76: generating 1000 training examples…
epoch 76: training using 100 minibatches of size 10…
epoch 76: evaluating on 100 examples…
epoch 76: est. objective value: 0.0
epoch 77: generating 1000 training examples…
epoch 77: training using 100 minibatches of size 10…
epoch 77: evaluating on 100 examples…
epoch 77: est. objective value: 0.0
epoch 78: generating 1000 training examples…
epoch 78: training using 100 minibatches of size 10…
epoch 78: evaluating on 100 examples…
epoch 78: est. objective value: 0.0
epoch 79: generating 1000 training examples…
epoch 79: training using 100 minibatches of size 10…
epoch 79: evaluating on 100 examples…
epoch 79: est. objective value: 0.0
epoch 80: generating 1000 training examples…
epoch 80: training using 100 minibatches of size 10…
epoch 80: evaluating on 100 examples…
epoch 80: est. objective value: 0.0
epoch 81: generating 1000 training examples…
epoch 81: training using 100 minibatches of size 10…
epoch 81: evaluating on 100 examples…
epoch 81: est. objective value: 0.0
epoch 82: generating 1000 training examples…
epoch 82: training using 100 minibatches of size 10…
epoch 82: evaluating on 100 examples…
epoch 82: est. objective value: 0.0
epoch 83: generating 1000 training examples…
epoch 83: training using 100 minibatches of size 10…
epoch 83: evaluating on 100 examples…
epoch 83: est. objective value: 0.0
epoch 84: generating 1000 training examples…
epoch 84: training using 100 minibatches of size 10…
epoch 84: evaluating on 100 examples…
epoch 84: est. objective value: 0.0
epoch 85: generating 1000 training examples…
epoch 85: training using 100 minibatches of size 10…
epoch 85: evaluating on 100 examples…
epoch 85: est. objective value: 0.0
epoch 86: generating 1000 training examples…
epoch 86: training using 100 minibatches of size 10…
epoch 86: evaluating on 100 examples…
epoch 86: est. objective value: 0.0
epoch 87: generating 1000 training examples…
epoch 87: training using 100 minibatches of size 10…
epoch 87: evaluating on 100 examples…
epoch 87: est. objective value: 0.0
epoch 88: generating 1000 training examples…
epoch 88: training using 100 minibatches of size 10…
epoch 88: evaluating on 100 examples…
epoch 88: est. objective value: 0.0
epoch 89: generating 1000 training examples…
epoch 89: training using 100 minibatches of size 10…
epoch 89: evaluating on 100 examples…
epoch 89: est. objective value: 0.0
epoch 90: generating 1000 training examples…
epoch 90: training using 100 minibatches of size 10…
epoch 90: evaluating on 100 examples…
epoch 90: est. objective value: 0.0
epoch 91: generating 1000 training examples…
epoch 91: training using 100 minibatches of size 10…
epoch 91: evaluating on 100 examples…
epoch 91: est. objective value: 0.0
epoch 92: generating 1000 training examples…
epoch 92: training using 100 minibatches of size 10…
epoch 92: evaluating on 100 examples…
epoch 92: est. objective value: 0.0
epoch 93: generating 1000 training examples…
epoch 93: training using 100 minibatches of size 10…
epoch 93: evaluating on 100 examples…
epoch 93: est. objective value: 0.0
epoch 94: generating 1000 training examples…
epoch 94: training using 100 minibatches of size 10…
epoch 94: evaluating on 100 examples…
epoch 94: est. objective value: 0.0
epoch 95: generating 1000 training examples…
epoch 95: training using 100 minibatches of size 10…
epoch 95: evaluating on 100 examples…
epoch 95: est. objective value: 0.0
epoch 96: generating 1000 training examples…
epoch 96: training using 100 minibatches of size 10…
epoch 96: evaluating on 100 examples…
epoch 96: est. objective value: 0.0
epoch 97: generating 1000 training examples…
epoch 97: training using 100 minibatches of size 10…
epoch 97: evaluating on 100 examples…
epoch 97: est. objective value: 0.0
epoch 98: generating 1000 training examples…
epoch 98: training using 100 minibatches of size 10…
epoch 98: evaluating on 100 examples…
epoch 98: est. objective value: 0.0
epoch 99: generating 1000 training examples…
epoch 99: training using 100 minibatches of size 10…
epoch 99: evaluating on 100 examples…
epoch 99: est. objective value: 0.0
epoch 100: generating 1000 training examples…
epoch 100: training using 100 minibatches of size 10…
epoch 100: evaluating on 100 examples…
epoch 100: est. objective value: 0.0
epoch 101: generating 1000 training examples…
epoch 101: training using 100 minibatches of size 10…
epoch 101: evaluating on 100 examples…
epoch 101: est. objective value: 0.0
epoch 102: generating 1000 training examples…
epoch 102: training using 100 minibatches of size 10…
epoch 102: evaluating on 100 examples…
epoch 102: est. objective value: 0.0
epoch 103: generating 1000 training examples…
epoch 103: training using 100 minibatches of size 10…
epoch 103: evaluating on 100 examples…
epoch 103: est. objective value: 0.0
epoch 104: generating 1000 training examples…
epoch 104: training using 100 minibatches of size 10…
epoch 104: evaluating on 100 examples…
epoch 104: est. objective value: 0.0
epoch 105: generating 1000 training examples…
epoch 105: training using 100 minibatches of size 10…
epoch 105: evaluating on 100 examples…
epoch 105: est. objective value: 0.0
epoch 106: generating 1000 training examples…
epoch 106: training using 100 minibatches of size 10…
epoch 106: evaluating on 100 examples…
epoch 106: est. objective value: 0.0
epoch 107: generating 1000 training examples…
epoch 107: training using 100 minibatches of size 10…
epoch 107: evaluating on 100 examples…
epoch 107: est. objective value: 0.0
epoch 108: generating 1000 training examples…
epoch 108: training using 100 minibatches of size 10…
epoch 108: evaluating on 100 examples…
epoch 108: est. objective value: 0.0
epoch 109: generating 1000 training examples…
epoch 109: training using 100 minibatches of size 10…
epoch 109: evaluating on 100 examples…
epoch 109: est. objective value: 0.0
epoch 110: generating 1000 training examples…
epoch 110: training using 100 minibatches of size 10…
epoch 110: evaluating on 100 examples…
epoch 110: est. objective value: 0.0
epoch 111: generating 1000 training examples…
epoch 111: training using 100 minibatches of size 10…
epoch 111: evaluating on 100 examples…
epoch 111: est. objective value: 0.0
epoch 112: generating 1000 training examples…
epoch 112: training using 100 minibatches of size 10…
epoch 112: evaluating on 100 examples…
epoch 112: est. objective value: 0.0
epoch 113: generating 1000 training examples…
epoch 113: training using 100 minibatches of size 10…
epoch 113: evaluating on 100 examples…
epoch 113: est. objective value: 0.0
epoch 114: generating 1000 training examples…
epoch 114: training using 100 minibatches of size 10…
epoch 114: evaluating on 100 examples…
epoch 114: est. objective value: 0.0
epoch 115: generating 1000 training examples…
epoch 115: training using 100 minibatches of size 10…
epoch 115: evaluating on 100 examples…
epoch 115: est. objective value: 0.0
epoch 116: generating 1000 training examples…
epoch 116: training using 100 minibatches of size 10…
epoch 116: evaluating on 100 examples…
epoch 116: est. objective value: 0.0
epoch 117: generating 1000 training examples…
epoch 117: training using 100 minibatches of size 10…
epoch 117: evaluating on 100 examples…
epoch 117: est. objective value: 0.0
epoch 118: generating 1000 training examples…
epoch 118: training using 100 minibatches of size 10…
epoch 118: evaluating on 100 examples…
epoch 118: est. objective value: 0.0
epoch 119: generating 1000 training examples…
epoch 119: training using 100 minibatches of size 10…
epoch 119: evaluating on 100 examples…
epoch 119: est. objective value: 0.0
epoch 120: generating 1000 training examples…
epoch 120: training using 100 minibatches of size 10…
epoch 120: evaluating on 100 examples…
epoch 120: est. objective value: 0.0
epoch 121: generating 1000 training examples…
epoch 121: training using 100 minibatches of size 10…
epoch 121: evaluating on 100 examples…
epoch 121: est. objective value: 0.0
epoch 122: generating 1000 training examples…
epoch 122: training using 100 minibatches of size 10…
epoch 122: evaluating on 100 examples…
epoch 122: est. objective value: 0.0
epoch 123: generating 1000 training examples…
epoch 123: training using 100 minibatches of size 10…
epoch 123: evaluating on 100 examples…
epoch 123: est. objective value: 0.0
epoch 124: generating 1000 training examples…
epoch 124: training using 100 minibatches of size 10…
epoch 124: evaluating on 100 examples…
epoch 124: est. objective value: 0.0
epoch 125: generating 1000 training examples…
epoch 125: training using 100 minibatches of size 10…
epoch 125: evaluating on 100 examples…
epoch 125: est. objective value: 0.0
epoch 126: generating 1000 training examples…
epoch 126: training using 100 minibatches of size 10…
epoch 126: evaluating on 100 examples…
epoch 126: est. objective value: 0.0
epoch 127: generating 1000 training examples…
epoch 127: training using 100 minibatches of size 10…
epoch 127: evaluating on 100 examples…
epoch 127: est. objective value: 0.0
epoch 128: generating 1000 training examples…
epoch 128: training using 100 minibatches of size 10…
epoch 128: evaluating on 100 examples…
epoch 128: est. objective value: 0.0
epoch 129: generating 1000 training examples…
epoch 129: training using 100 minibatches of size 10…
epoch 129: evaluating on 100 examples…
epoch 129: est. objective value: 0.0
epoch 130: generating 1000 training examples…
epoch 130: training using 100 minibatches of size 10…
epoch 130: evaluating on 100 examples…
epoch 130: est. objective value: 0.0
epoch 131: generating 1000 training examples…
epoch 131: training using 100 minibatches of size 10…
epoch 131: evaluating on 100 examples…
epoch 131: est. objective value: 0.0
epoch 132: generating 1000 training examples…
epoch 132: training using 100 minibatches of size 10…
epoch 132: evaluating on 100 examples…
epoch 132: est. objective value: 0.0
epoch 133: generating 1000 training examples…
epoch 133: training using 100 minibatches of size 10…
epoch 133: evaluating on 100 examples…
epoch 133: est. objective value: 0.0
epoch 134: generating 1000 training examples…
epoch 134: training using 100 minibatches of size 10…
epoch 134: evaluating on 100 examples…
epoch 134: est. objective value: 0.0
epoch 135: generating 1000 training examples…
epoch 135: training using 100 minibatches of size 10…
epoch 135: evaluating on 100 examples…
epoch 135: est. objective value: 0.0
epoch 136: generating 1000 training examples…
epoch 136: training using 100 minibatches of size 10…
epoch 136: evaluating on 100 examples…
epoch 136: est. objective value: 0.0
epoch 137: generating 1000 training examples…
epoch 137: training using 100 minibatches of size 10…
epoch 137: evaluating on 100 examples…
epoch 137: est. objective value: 0.0
epoch 138: generating 1000 training examples…
epoch 138: training using 100 minibatches of size 10…
epoch 138: evaluating on 100 examples…
epoch 138: est. objective value: 0.0
epoch 139: generating 1000 training examples…
epoch 139: training using 100 minibatches of size 10…
epoch 139: evaluating on 100 examples…
epoch 139: est. objective value: 0.0
epoch 140: generating 1000 training examples…
epoch 140: training using 100 minibatches of size 10…
epoch 140: evaluating on 100 examples…
epoch 140: est. objective value: 0.0
epoch 141: generating 1000 training examples…
epoch 141: training using 100 minibatches of size 10…
epoch 141: evaluating on 100 examples…
epoch 141: est. objective value: 0.0
epoch 142: generating 1000 training examples…
epoch 142: training using 100 minibatches of size 10…
epoch 142: evaluating on 100 examples…
epoch 142: est. objective value: 0.0
epoch 143: generating 1000 training examples…
epoch 143: training using 100 minibatches of size 10…
epoch 143: evaluating on 100 examples…
epoch 143: est. objective value: 0.0
epoch 144: generating 1000 training examples…
epoch 144: training using 100 minibatches of size 10…
epoch 144: evaluating on 100 examples…
epoch 144: est. objective value: 0.0
epoch 145: generating 1000 training examples…
epoch 145: training using 100 minibatches of size 10…
epoch 145: evaluating on 100 examples…
epoch 145: est. objective value: 0.0
epoch 146: generating 1000 training examples…
epoch 146: training using 100 minibatches of size 10…
epoch 146: evaluating on 100 examples…
epoch 146: est. objective value: 0.0
epoch 147: generating 1000 training examples…
epoch 147: training using 100 minibatches of size 10…
epoch 147: evaluating on 100 examples…
epoch 147: est. objective value: 0.0
epoch 148: generating 1000 training examples…
epoch 148: training using 100 minibatches of size 10…
epoch 148: evaluating on 100 examples…
epoch 148: est. objective value: 0.0
epoch 149: generating 1000 training examples…
epoch 149: training using 100 minibatches of size 10…
epoch 149: evaluating on 100 examples…
epoch 149: est. objective value: 0.0
epoch 150: generating 1000 training examples…
epoch 150: training using 100 minibatches of size 10…
epoch 150: evaluating on 100 examples…
epoch 150: est. objective value: 0.0
epoch 151: generating 1000 training examples…
epoch 151: training using 100 minibatches of size 10…
epoch 151: evaluating on 100 examples…
epoch 151: est. objective value: 0.0
epoch 152: generating 1000 training examples…
epoch 152: training using 100 minibatches of size 10…
epoch 152: evaluating on 100 examples…
epoch 152: est. objective value: 0.0
epoch 153: generating 1000 training examples…
epoch 153: training using 100 minibatches of size 10…
epoch 153: evaluating on 100 examples…
epoch 153: est. objective value: 0.0
epoch 154: generating 1000 training examples…
epoch 154: training using 100 minibatches of size 10…
epoch 154: evaluating on 100 examples…
epoch 154: est. objective value: 0.0
epoch 155: generating 1000 training examples…
epoch 155: training using 100 minibatches of size 10…
epoch 155: evaluating on 100 examples…
epoch 155: est. objective value: 0.0
epoch 156: generating 1000 training examples…
epoch 156: training using 100 minibatches of size 10…
epoch 156: evaluating on 100 examples…
epoch 156: est. objective value: 0.0
epoch 157: generating 1000 training examples…
epoch 157: training using 100 minibatches of size 10…
epoch 157: evaluating on 100 examples…
epoch 157: est. objective value: 0.0
epoch 158: generating 1000 training examples…
epoch 158: training using 100 minibatches of size 10…
epoch 158: evaluating on 100 examples…
epoch 158: est. objective value: 0.0
epoch 159: generating 1000 training examples…
epoch 159: training using 100 minibatches of size 10…
epoch 159: evaluating on 100 examples…
epoch 159: est. objective value: 0.0
epoch 160: generating 1000 training examples…
epoch 160: training using 100 minibatches of size 10…
epoch 160: evaluating on 100 examples…
epoch 160: est. objective value: 0.0
epoch 161: generating 1000 training examples…
epoch 161: training using 100 minibatches of size 10…
epoch 161: evaluating on 100 examples…
epoch 161: est. objective value: 0.0
epoch 162: generating 1000 training examples…
epoch 162: training using 100 minibatches of size 10…
epoch 162: evaluating on 100 examples…
epoch 162: est. objective value: 0.0
epoch 163: generating 1000 training examples…
epoch 163: training using 100 minibatches of size 10…
epoch 163: evaluating on 100 examples…
epoch 163: est. objective value: 0.0
epoch 164: generating 1000 training examples…
epoch 164: training using 100 minibatches of size 10…
epoch 164: evaluating on 100 examples…
epoch 164: est. objective value: 0.0
epoch 165: generating 1000 training examples…
epoch 165: training using 100 minibatches of size 10…
epoch 165: evaluating on 100 examples…
epoch 165: est. objective value: 0.0
epoch 166: generating 1000 training examples…
epoch 166: training using 100 minibatches of size 10…
epoch 166: evaluating on 100 examples…
epoch 166: est. objective value: 0.0
epoch 167: generating 1000 training examples…
epoch 167: training using 100 minibatches of size 10…
epoch 167: evaluating on 100 examples…
epoch 167: est. objective value: 0.0
epoch 168: generating 1000 training examples…
epoch 168: training using 100 minibatches of size 10…
epoch 168: evaluating on 100 examples…
epoch 168: est. objective value: 0.0
epoch 169: generating 1000 training examples…
epoch 169: training using 100 minibatches of size 10…
epoch 169: evaluating on 100 examples…
epoch 169: est. objective value: 0.0
epoch 170: generating 1000 training examples…
epoch 170: training using 100 minibatches of size 10…
epoch 170: evaluating on 100 examples…
epoch 170: est. objective value: 0.0
epoch 171: generating 1000 training examples…
epoch 171: training using 100 minibatches of size 10…
epoch 171: evaluating on 100 examples…
epoch 171: est. objective value: 0.0
epoch 172: generating 1000 training examples…
epoch 172: training using 100 minibatches of size 10…
epoch 172: evaluating on 100 examples…
epoch 172: est. objective value: 0.0
epoch 173: generating 1000 training examples…
epoch 173: training using 100 minibatches of size 10…
epoch 173: evaluating on 100 examples…
epoch 173: est. objective value: 0.0
epoch 174: generating 1000 training examples…
epoch 174: training using 100 minibatches of size 10…
epoch 174: evaluating on 100 examples…
epoch 174: est. objective value: 0.0
epoch 175: generating 1000 training examples…
epoch 175: training using 100 minibatches of size 10…
epoch 175: evaluating on 100 examples…
epoch 175: est. objective value: 0.0
epoch 176: generating 1000 training examples…
epoch 176: training using 100 minibatches of size 10…
epoch 176: evaluating on 100 examples…
epoch 176: est. objective value: 0.0
epoch 177: generating 1000 training examples…
epoch 177: training using 100 minibatches of size 10…
epoch 177: evaluating on 100 examples…
epoch 177: est. objective value: 0.0
epoch 178: generating 1000 training examples…
epoch 178: training using 100 minibatches of size 10…
epoch 178: evaluating on 100 examples…
epoch 178: est. objective value: 0.0
epoch 179: generating 1000 training examples…
epoch 179: training using 100 minibatches of size 10…
epoch 179: evaluating on 100 examples…
epoch 179: est. objective value: 0.0
epoch 180: generating 1000 training examples…
epoch 180: training using 100 minibatches of size 10…
epoch 180: evaluating on 100 examples…
epoch 180: est. objective value: 0.0
epoch 181: generating 1000 training examples…
epoch 181: training using 100 minibatches of size 10…
epoch 181: evaluating on 100 examples…
epoch 181: est. objective value: 0.0
epoch 182: generating 1000 training examples…
epoch 182: training using 100 minibatches of size 10…
epoch 182: evaluating on 100 examples…
epoch 182: est. objective value: 0.0
epoch 183: generating 1000 training examples…
epoch 183: training using 100 minibatches of size 10…
epoch 183: evaluating on 100 examples…
epoch 183: est. objective value: 0.0
epoch 184: generating 1000 training examples…
epoch 184: training using 100 minibatches of size 10…
epoch 184: evaluating on 100 examples…
epoch 184: est. objective value: 0.0
epoch 185: generating 1000 training examples…
epoch 185: training using 100 minibatches of size 10…
epoch 185: evaluating on 100 examples…
epoch 185: est. objective value: 0.0
epoch 186: generating 1000 training examples…
epoch 186: training using 100 minibatches of size 10…
epoch 186: evaluating on 100 examples…
epoch 186: est. objective value: 0.0
epoch 187: generating 1000 training examples…
epoch 187: training using 100 minibatches of size 10…
epoch 187: evaluating on 100 examples…
epoch 187: est. objective value: 0.0
epoch 188: generating 1000 training examples…
epoch 188: training using 100 minibatches of size 10…
epoch 188: evaluating on 100 examples…
epoch 188: est. objective value: 0.0
epoch 189: generating 1000 training examples…
epoch 189: training using 100 minibatches of size 10…
epoch 189: evaluating on 100 examples…
epoch 189: est. objective value: 0.0
epoch 190: generating 1000 training examples…
epoch 190: training using 100 minibatches of size 10…
epoch 190: evaluating on 100 examples…
epoch 190: est. objective value: 0.0
epoch 191: generating 1000 training examples…
epoch 191: training using 100 minibatches of size 10…
epoch 191: evaluating on 100 examples…
epoch 191: est. objective value: 0.0
epoch 192: generating 1000 training examples…
epoch 192: training using 100 minibatches of size 10…
epoch 192: evaluating on 100 examples…
epoch 192: est. objective value: 0.0
epoch 193: generating 1000 training examples…
epoch 193: training using 100 minibatches of size 10…
epoch 193: evaluating on 100 examples…
epoch 193: est. objective value: 0.0
epoch 194: generating 1000 training examples…
epoch 194: training using 100 minibatches of size 10…
epoch 194: evaluating on 100 examples…
epoch 194: est. objective value: 0.0
epoch 195: generating 1000 training examples…
epoch 195: training using 100 minibatches of size 10…
epoch 195: evaluating on 100 examples…
epoch 195: est. objective value: 0.0
epoch 196: generating 1000 training examples…
epoch 196: training using 100 minibatches of size 10…
epoch 196: evaluating on 100 examples…
epoch 196: est. objective value: 0.0
epoch 197: generating 1000 training examples…
epoch 197: training using 100 minibatches of size 10…
epoch 197: evaluating on 100 examples…
epoch 197: est. objective value: 0.0
epoch 198: generating 1000 training examples…
epoch 198: training using 100 minibatches of size 10…
epoch 198: evaluating on 100 examples…
epoch 198: est. objective value: 0.0
epoch 199: generating 1000 training examples…
epoch 199: training using 100 minibatches of size 10…
epoch 199: evaluating on 100 examples…
epoch 199: est. objective value: 0.0
epoch 200: generating 1000 training examples…
epoch 200: training using 100 minibatches of size 10…
epoch 200: evaluating on 100 examples…
epoch 200: est. objective value: 0.0
```

The following code loads a test observation and visualizes it.

[79]:
```
obs_matrix = readdlm("test-scene.txt")
obs = vec(obs_matrix)
obs = convert(Vector{Float64}, obs)
p1 = visualize(obs_matrix)
```



### 1.1.6 Q 1F [2 pts]

Conditioned on this observation, run importance sampling with or without `neural_amortized_inference` as the data-driven proposal. Compare the average log probabilities of the two methods. The importance sampling with `neural_amortized_inference` proposal should have significantly better results.

[80]:
```julia
# load trained parameters
let data = JLD2.load("neural_amortized_inference_trained.jld2", "data")
    for name in [:W1, :b1, :W2, :b2]
        Gen.init_param!(neural_amortized_inference, name, data[(:param, name)])
    end
end;
```

[81]:
```julia
function logmeanexp(scores)
    logsumexp(scores) - log(length(scores))
end;

# Make constraints based on the observed image (obs_matrix)
```

```
constraints = Gen.choicemap()
for row_id in 1:N_ROWS
    for col_id in 1:N_COLS
        constraints[:image => row_id => col_id => :brightness] =␣
  ↪obs_matrix[row_id, col_id]
    end
end

traces = Vector()
for _ in 1:10
    (trace, _) = importance_resampling(two_d_world, (), constraints, 100)
    push!(traces, trace)
end

scores = [get_score(t) for t in traces]

println(logmeanexp(scores))
```

```
-340.1912669383046
```

[82]:
```
# run importance sampling *with* the neural amortized inference, amount of␣
  ↪compute = 100, repeat 10 times
traces = Vector()
for _ in 1:10
    (trace, _) = importance_resampling(two_d_world, (), constraints,␣
  ↪neural_amortized_inference, (vec(obs_matrix),), 100)
    push!(traces, trace)
end

scores = [get_score(t) for t in traces]

println(logmeanexp(scores))
```

```
137.1581832966427
```