DL Coursework 4

WU ZHI 17040772

PART 1: MNIST as a sequence

Task 1: (Next) Pixel prediction

Train Models

1. LSTM

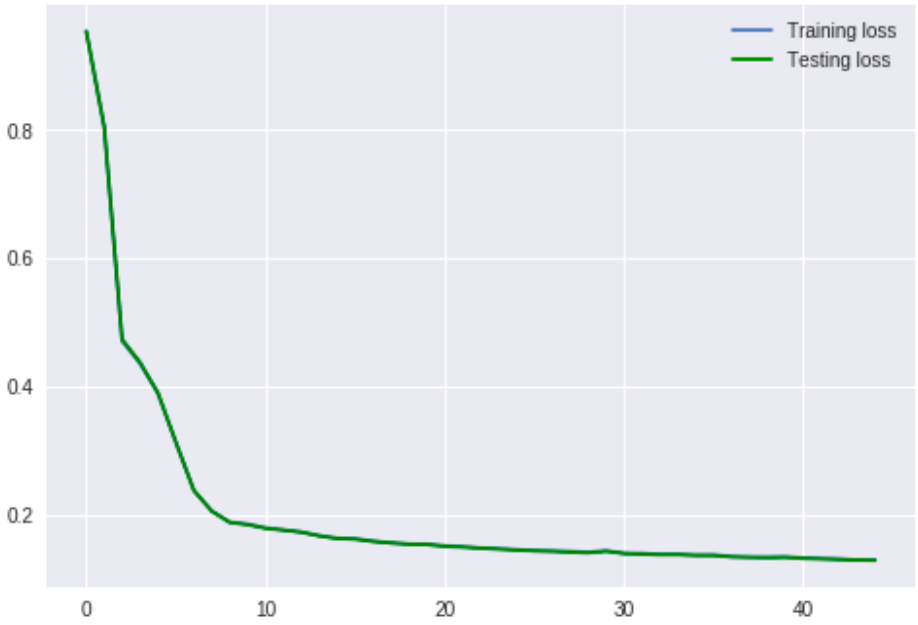


Figure 1 Training and testing loss of LSTM model

LSTM	(Beginning - 0 epochs)	(Mid-training - 1 epoch)	(End of training - 5 epochs)
Test loss	0.66882485	0.17764388	0.13497578
Test accuracy	0.787116	0.9326493	0.94286746

Table 1 Test loss and accuracy of LSTM

GRU	(Beginning - 0 epochs)	(Mid-training - 1 epoch)	(End of training - 5 epochs)
Test loss	1.1210192	0.23100725	0.13256024
Test accuracy	0.175847	0.9320999	0.9470945

Table 2 Test loss and accuracy of GRU

Pixel Prediction

Predict missing parts and compare with GT

	1 pixel	10 pixels	28 pixels	300 pixels
Epoch 0	0.93085945	0.7946974	1.0240359	1.1336417
Epoch 1	0.59918153	0.6517235	0.6749792	0.6922478
Epoch 5	0.5849918	0.60465163	0.64749354	0.70476395

Table 3 Test loss of LSTM

Discussion: From the table above, it can be found that when the epoch is same, the loss will increase if pixel number increases. When the number of the predict image is same, more epoch can reduce loss. However, difference between loss of epoch 1 and epoch 5 is very small, and it means the model converge slow after the first epoch.

Visualize completing the image

1. Successful Example

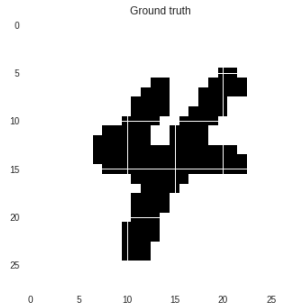


Figure 2 Successful Example ground truth

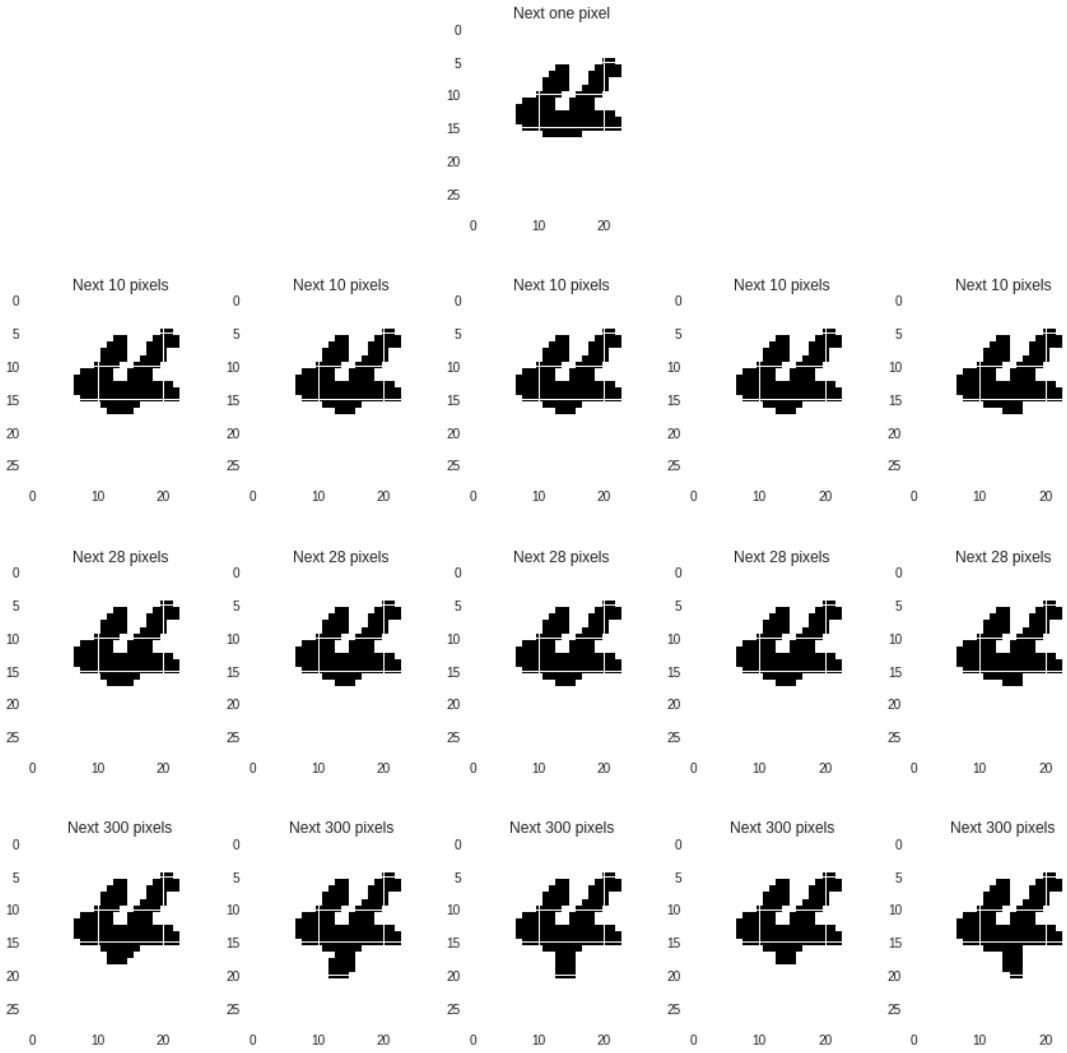


Figure 3 Successful example

2. Failure Example

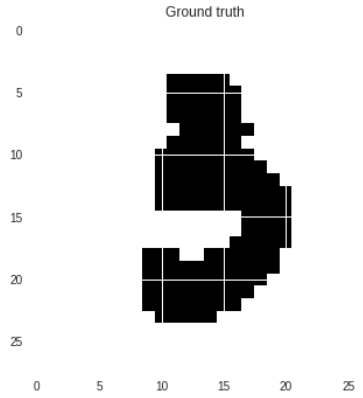


Figure 4 Failure Example ground truth

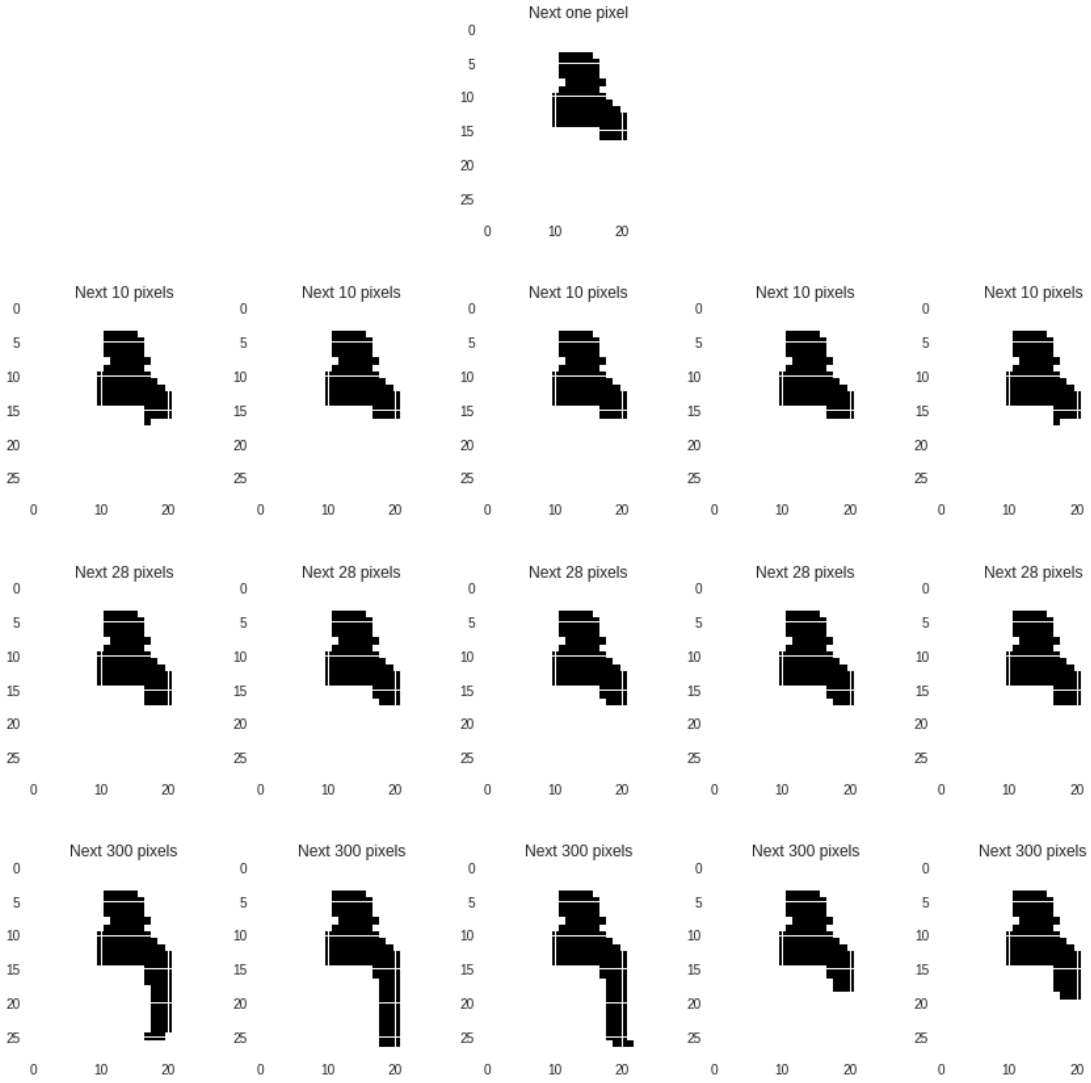


Figure 5 Failure Example

3. High Variance

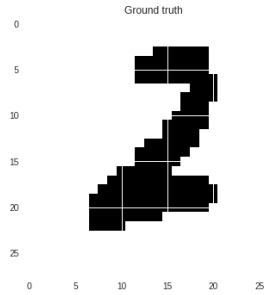


Figure 6 High Variance Ground truth

Figure 7 High Variance example

Task 2: Using pixel-to-pixel: In-painting task

1) Provide the formula used to compute the probability over the missing pixel and respectively for the missing patch.

One-pixel missing case

Missing pixel:

Assuming x_{miss} is the missing pixel, and $P(x_{miss})$ is the probability calculated by trained model (Based on previous pixels). $x_1, x_2, x_3 \dots x_n$ is the future pixels.

$$P(x_{miss}|x_1, x_2, x_3 \dots x_n) = \frac{P(x_1, x_2, x_3 \dots x_n|x_{miss}).\, p(x_{miss})}{p(x_1, x_2, x_3 \dots x_n)}$$

And Likelihood is:

$$\begin{aligned} P(x_1, x_2, x_3 \dots x_n|x_{miss}) &= P(x_1|x_{miss}).\, P(x_2, x_3 \dots x_n|x_1, x_{missing}) \\ &= P(x_1|x_{miss}).\, P(x_2|x_1, x_{miss}).\, P(x_3 \dots x_n|x_1, x_{missing}) \\ &= P(x_1|x_{miss}).\, P(x_2|x_1, x_{miss}) \cdots P(x_n|x_{missing}, x_1, x_2 \cdots x_{n-1}) \end{aligned}$$

So, when likelihood is maximum likelihood estimation, the probability of missing pixel is the most probable.

Missing patch:

For a 2×2 missing patch in 28×28, assuming the missing pixels are x_m, x_{m+1}, x_{m+2} and x_{m+3}.

For x_m, we can use the previous method (the method in missing pixel) to calculate the most probable (0 or 1).

$$P(x_m|x_1, x_2, x_3 \dots x_n) = \frac{P(x_1, x_2, x_3 \dots x_n|x_m).\, p(x_m)}{p(x_1, x_2, x_3 \dots x_n)}$$

However the future points here are exclusive of x_{m+1}, x_{m+2} and x_{m+3}, beacause we do not know the actual pixel values of them.

The likelihood here is:

$$\begin{aligned} P(x_1, x_2, x_3 \dots x_n|x_m) &= P(x_1|x_m).\, P(x_2, x_3 \dots x_n|x_1, x_m) \\ &= P(x_1|x_m).\, P(x_2|x_1, x_m).\, P(x_3 \dots x_n|x_1, x_m) \\ &= P(x_1|x_m).\, P(x_2|x_1, x_m) \cdots P(x_n|x_m, x_1, x_2 \cdots x_{n-1}) \end{aligned}$$

The pixel value of x_m should be 1 or 0, so we need to calculate the most probable probability $P(x_m = 1|x_1, x_2, x_3 \dots x_n)$ and $P(x_m = 0|x_1, x_2, x_3 \dots x_n)$, and we choose the higher probability between $P(x_m = 1|x_1, x_2, x_3 \dots x_n)$ and $P(x_m = 0|x_1, x_2, x_3 \dots x_n)$ to predict x_m value.

After calcuating x_m, we can use trained model based previous pixel value and x_m to calculate $P(x_{m+1})$, and the pixel prediction of x_{m+1} can be calculated by the same way as x_m, but the future pixels for x_{m+1} are exclusive of x_{m+2} and x_{m+3}.

When we use same method to predict x_{m+2}, the future pixels for x_{m+1} are exclusive of x_{m+3}. Affer we predict x_m, x_{m+1} and x_{m+2}, we can use method of missing pixel to predict it.

2) Visualize the most probable in-painting, according to your model. How does this compare to the ground truth? (Compare cross-entropy between your most probable sample and the ground truth). Explain the difference. It is enough to include just one example per task/dataset.

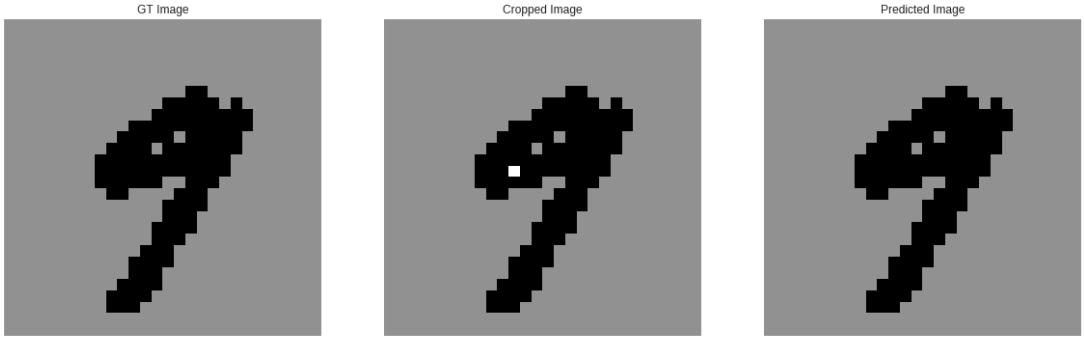


Figure 8 Predicted one missing pixel

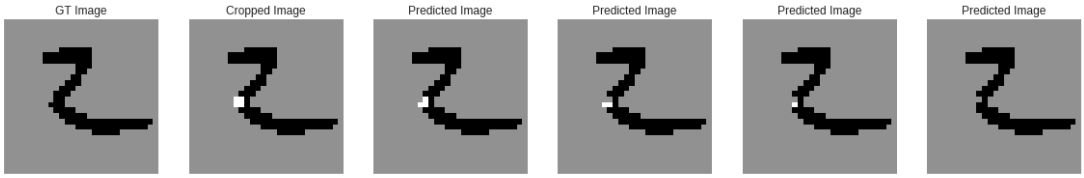


Figure 9 Predicted 2×2missing pixels

How does this compare to the ground truth? (Compare cross-entropy between your most probable sample and the ground truth).

The cross-entropy of one pixel is 0.323343, and one patch is 0.786534. The cross-entropy of one pixel here is smaller than the loss in our training model epoch 5 result, and future pixels here can decrease loss. One patch contain 4 pixel, so it is likely have higher loss than one pixel.

PART 2: Learning multiple tasks with LSTM-s

Question

1) **Without running any experiments (5 pts)**, try to think about the following scenarios and answer these questions:

- Consider we generate the data with $Dirichlet(\alpha)$, where $\alpha = (10., 1., 1.)$. What do you think the LSTM model will learn, if anything? Remember we are effectively changing the prediction task, every time we are resampling the probability vector $\mathbf{p}$. *Hint: Think about the distribution over* $\mathbf{p}$ *that this prior induces.*

The prior distribution of $\mathbf{p}$ is $p(\mathbf{p}) = \frac{1}{B(\alpha)} \prod_{i=1}^{3} p_i^{\alpha i-1}$, and function B is the Beta function (multivariate). $\alpha = (10, 1, 1)$ will result in p_1 having the majority of the weight subject to $p_1 + p_2 + p_3 = 1$, so p_1 is very likely to take a value close to one, which means symbol one will be sampled for most of the time. LSTM will perform better on such stable patterns, all it has left to learn is the when to reset.

- What if we consider a more uniformative prior, like $\alpha = (1.1, 1.1, 1.1)$?

If $\alpha = (1.1, 1.1, 1.1)$, this prior will result in p_1, p_2 and p_3 being uniformly distributed. Thus the sequence they generated are more fulctuate and harder for LSTM to predict. Maybe in this case LSTM cannot interpret the hidden structure well, given that the probabilities been reset regularly, since it has equal likely chances of seeing extreme values (like $(1, 0, 0)$) and more averaged values (like $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$) of p's.

- How does this (learning ability and generalization) depend on the lenght of the tasks T and the unrolling length on the LSTM? It might be helpful to consider the two extremes: i) $T = 1$ (we reset the task at every time step). What should the model learn in this case?, ii) $T = \infty$ (we sample the task once and keep it forever). What should the model learn in this case? (Answer this for both previous priors)

For T =1,the it is very hard for extreme prior to makes any difference to the leaning and generalization ability, because the resampled p is very close to (1,0,0) in each step. When T = ∞, it does not need to learn the time of reset and just predict next symbol, so the leaning and generalization ability is not so important. If the prior is in uniform distribution, for T = 1, the LSTM will perform very badly in prediction because it cannot learn the potential structure well. Although it can study the precise potential distribution, the randomness in the uniform prior is very high and it cannot learn a high accuracy. If T = ∞, LSTM can learn well in terms of the enough unrolling length because the probabilities are constant. However, the generalization could be not good.

- Does this increase or descrease the complexity of the prediction problem? What about the ability to generalize to unseen $\mathbf{p}$?

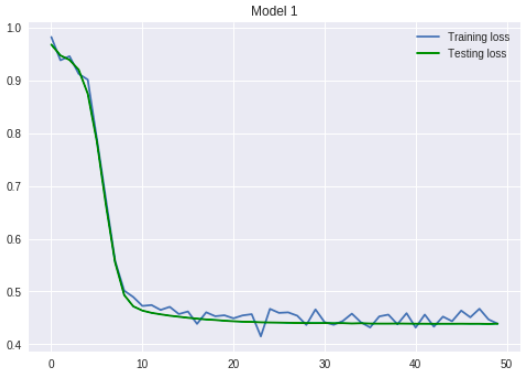
The complexity can be considered as the difficulty of predicting sequence unseen in training. If the T is large, the complexity usually will increase, because the model is harder to learn the potential distributions. The model may just use a few combinations to learn the probabilities. However, the combinations used could be sampled from a large range. Thus, the accuracy of predicting on unseen p will decrease and the complexity will increase.

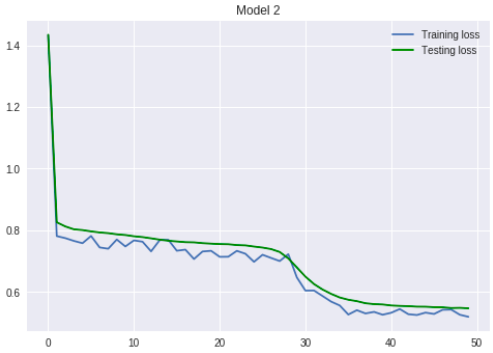
4) **Comparison to the Bayesian update (15 pts)**

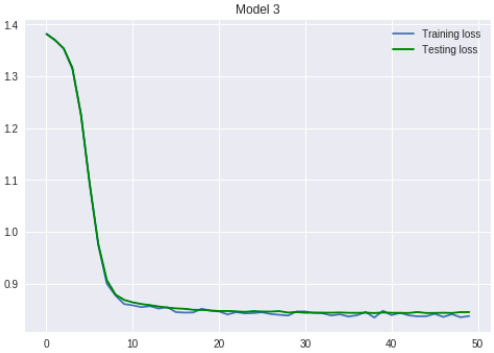
Going back to the generative process in the task description. For a given prior, for each the mini-tasks (selecting/sampling a $\mathbf{p}$), one could compute the Bayesian posterior at each time step. We start with a prior and every time we observe a symbol with update our posterior over the parameters $\mathbf{p}$ given the data. We do this every time step, till we reach the RESET symbol which marks the end of the task. Then we start again, from the prior.

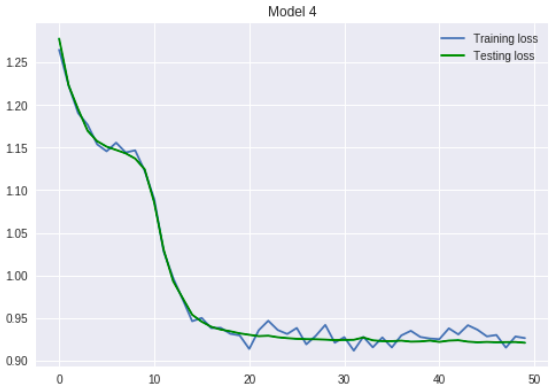
i) Derive the posterior update for each time step. (Hint: since the two distribution are conjugates or each other, the posterior has a closed form). **[3 pts]**

2) **Time to check your intuitions (10 pts)**









From the training loss and test loss for 4 models above, it can be found that training loss is not smooth, whereas testing loss is smooth. Both training loss and testing loss are stable in the end of training time for all 4 models.

3) **Analysis results (10 pts)**

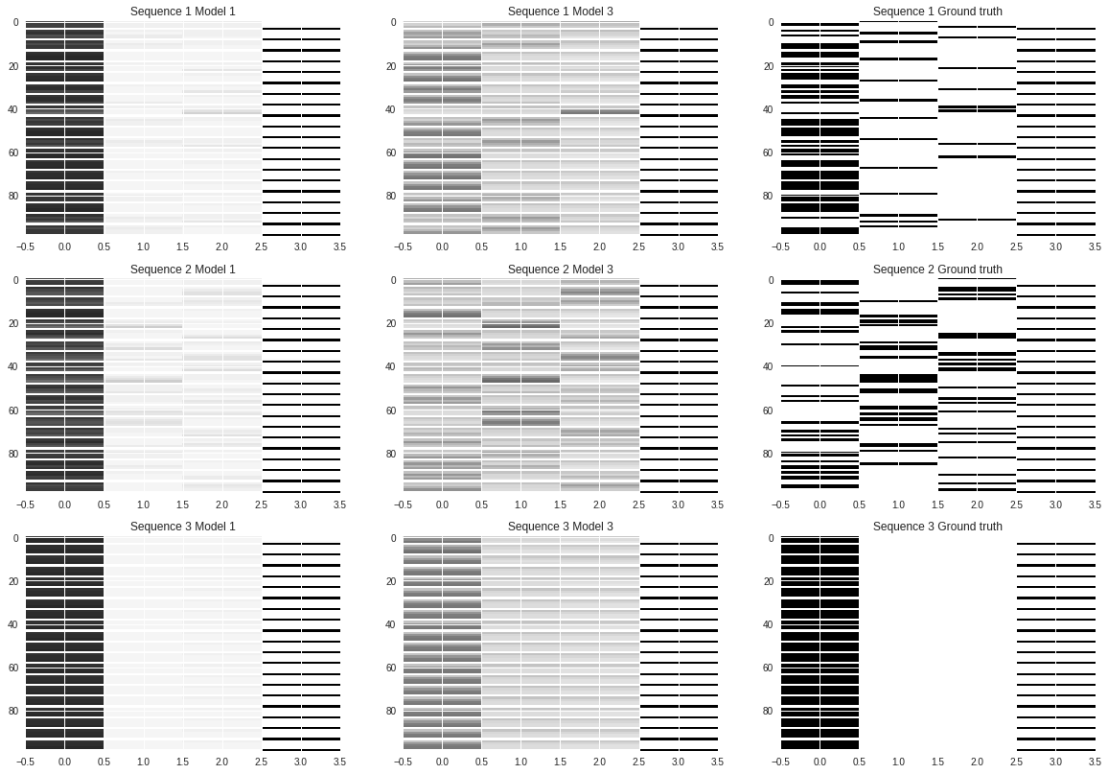


Figure 10 Model 1 and 3

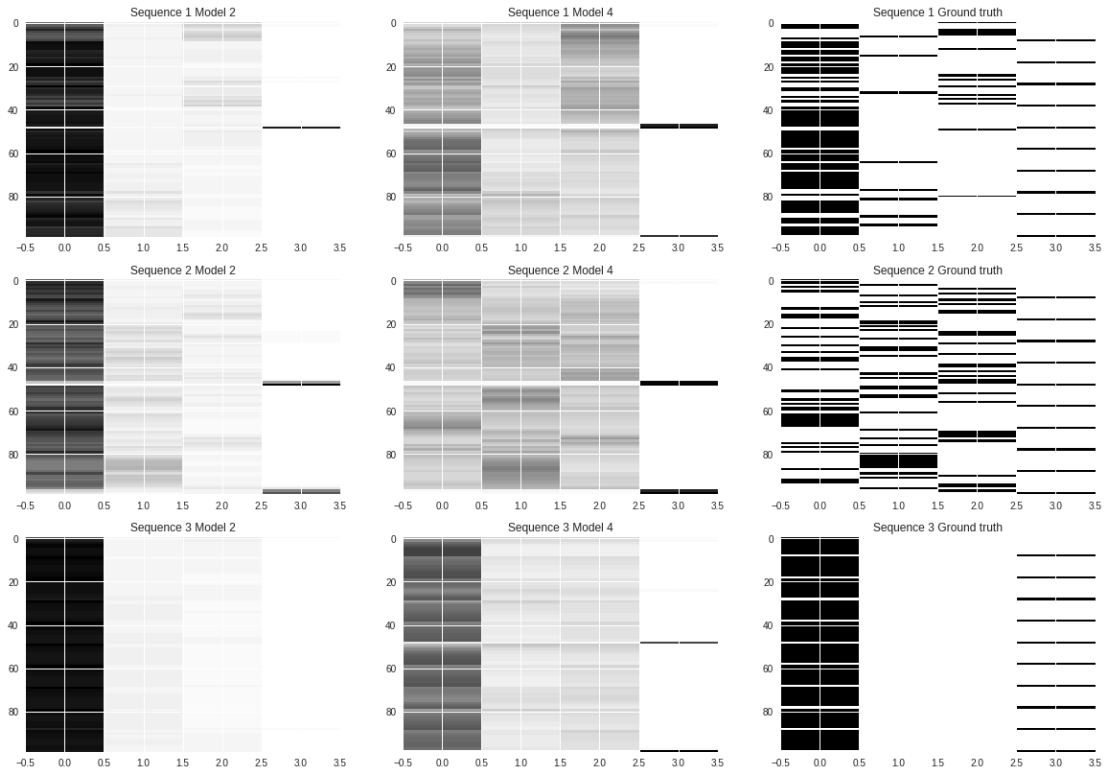


Figure 11 Model 2 and 4

4) **Comparison to the Bayesian update (15 pts)**

i) Derive the posterior update for each time step. (Hint: since the two distribution are conjugates or each other, the posterior has a closed form). **[3 pts]**

ANS：

The prior is alos Dirichlet distribution because the prior and likelihood are conjugate.

The prior is:

$$p_1, p_2, p_3 \sim \frac{\Gamma(\sum_{i=1}^{3} \alpha_i)}{\sum_{i=1}^{3} \Gamma(\alpha_i)} \prod_{i=1}^{3} p_i^{\alpha_i - 1}$$

The likelihood is:

$$x_1, x_2, x_3 \sim \frac{\Gamma(n+1)}{\sum_{i=1}^{3} p_i^{x_i}} \prod_{i=1}^{3} p_i^{x_i}$$

Thus, for one observation $X_1 = (x_1^{(1)}, x_1^{(2)}, x_1^{(3)})$, the posterior is:

$$P(p_1, p_2, p_3 | X_1) \propto \prod_{i=1}^{3} p_i^{\alpha_i - 1} \prod_{i=1}^{3} p_i^{x_i} \propto \prod_{i=1}^{3} p_i^{\alpha_i - 1 + x_1^{(i)}}$$

Then we can see:

$$p_1, p_2, p_3 | X_1 \sim Dirichlet(\alpha_1 - 1 + x_1^{(1)}, \alpha_2 - 1 + x_1^{(2)}, \alpha_3 - 1 + x_1^{(3)})$$

Thus, for T observations $X_1, X_2, X_3, \ldots X_T$, the posterior is:

$$P(p_1, p_2, p_3 | X_1, X_2, X_3, \ldots X_T) \propto \prod_{i=1}^{3} p_i^{\alpha_i - 1 + \sum_{j=1}^{T} x_j^{(i)}}$$

Then we can see:

$$p_1, p_2, p_3 | X_1, X_2, X_3, \ldots X_T \sim Dirichlet(\alpha_1 - 1 + \sum_{i=1}^{T} x_i^{(1)}, \alpha_2 - 1 + \sum_{i=1}^{T} x_i^{(2)}, \alpha_3 - 1 + \sum_{i=1}^{T} x_i^{(3)})$$

ii) Implement this posterior update and use it to infer the probabilities over the next symbol, for the previously generated test sequences. This will tell you, what the inferred probabilities would be, if we knew the structure of the model, the prior and that the reset symbol means the tasks has finished and we should reset our estimate to the prior. (For test sequence 1 and 2, use the prior that generated them, for test sequence 3 compute the updates starting from both priors) **[5 pts]**

ANS:

The probabilities can be writen as:

$$P(x_{T+1} = i | X_1, X_2, X_3, \ldots X_T, \alpha_1, \alpha_2, \alpha_3) = \int_p P(x_{T+1} | p) P(p | X_1, X_2, X_3, \ldots X_T, \alpha_1, \alpha_2, \alpha_3) dp$$
$$= \frac{\sum_{i=1}^{3} \sum_{j=1}^{T} x_j^{(i)} + \alpha_i}{L + \sum_{m=1}^{3} \alpha_m}$$

Where $L = \sum_{i=1}^{3} \sum_{j=1}^{T} x_j^{(i)}$