Homework 6

SHREYA CHETAN PAWASKAR
pawaskas@uci.edu
12041645

Question 1:

a)

- Let $\pi_{k\ell} = p(z_{t+1} = \ell \mid z_t = k)$.
- Using the Markov structure of the model:

  $p(z_1, z_2, z_3) \quad = p(z_1)\, p(z_2 \mid z_1)\, p(z_3 \mid z_2)$
- Substitute the values as 0.8 and 0.9,

  $p(z_1 = z_2 = z_3 \quad = 0.5\pi_{11}\pi_{11} + 0.5\pi_{22}\pi_{22} == 0.725$
- **Answer = 0.725**

---

b)

- Let $phi_{ka} = p(x_t = a \mid z_t = k)$.
- Given the variables z_1, x_1 and x_2 are conditionally independent, let's begin by assessing the impact of the observation

  $p(z_1 \mid x_1 = 1) \propto p(z_1)\, p(x_1 = 1 \mid z_1) \propto [\phi_{11}, \phi_{21}] \propto [0.25, 0.7]$
- We apply the Markov state dynamics.

  $p(z_2 \mid x_1 = 1) = \sum_{z_1} p(z_2 \mid z_1)\, p(z_1 \mid x_1 = 1) \propto [\pi_{11}\phi_{11} + \pi_{21}\phi_{21}, \pi_{12}\phi_{11} + \pi_{22}\phi_{21}] \propto [0.27, 0.68]$

  $p(z_2 \mid x_1 = 1) = \mathrm{Cat}(z_2 \mid [0.284, 0.716])$
- Then we apply the observation model:

  $$p(x_2 = 1 \mid x_1 = 1) = \sum_{z_2} p(x_2 = 1 \mid z_2)\, p(z_{21} = 1) = 0.284\phi_{11} + 0.716\phi_{21} \approx 0.57$$
- When x_1 is unobserved then the probability that $x_2 = 1$ is lower:

  $p(z_2) \qquad = \sum_{z_1} p(z_2 \mid z_1)\, p(z_1) = \mathrm{Cat}(z_2 \mid [0.45, 0.55])$

  $p(x_2 = 1) \quad = \sum_{z_2} p(x_2 = 1 \mid z_2)\, p(z_2) = 0.45\phi_{11} + 0.55\phi_{21} \approx 0.50$
- **Answer = 0.5**

---

c)

- From the structure of the HMM model and using Bayes' rule, we have:

$$p(z_1 \mid x_1, x_2) = \sum_{z_2} p(z_1, z_2 \mid x_1, x_2) \propto \sum_{z_2} p(z_1, z_2, x_1, x_2)$$
$$\propto p(z_1)\, p(x_{11}) \sum_{z_2} p(z_2 \mid z_1)\, p(x_2 \mid z_2)$$

- Evaluating the second term gives

$$p(x_2 = 4 \mid z_1) \quad \propto [\pi_{11}\phi_{14} + \pi_{12}\phi_{24}, \pi_{21}\phi_{14} + \pi_{22}\phi_{24}] \propto [0.220, 0.115]$$

- Next we have:

$$p(z_1 \mid x_1 = 2, x_2 = 4) \quad \propto p(x_1 = 2 \mid z_1)\, p(x_2 = 4 \mid z_1) \propto [0.055, 0.0115]$$

- Finally, normalizing these terms gives us:
  $p(z_1 = 1 \mid x_1 = 2, x_2 = 4) \approx 0.83$.

Since neither roll equals 1 , it is likely that the fair die is in use.

---

d)

- The most likely state sequence is $z_t = 2$ for all t, as $p(z_1)$ is uniform. Also $\pi_{22} \geq \pi_{k\ell}$ for all k, ℓ.
- $\phi_{21} \geq \phi_{ka}$ for all k, a, so the corresponding most likely observation sequence is $x_t = 1$ for all t
- The most likely configuration of the HMM is that the biased die is used at all times, and all rolls = 1.

Question 2:

Code:

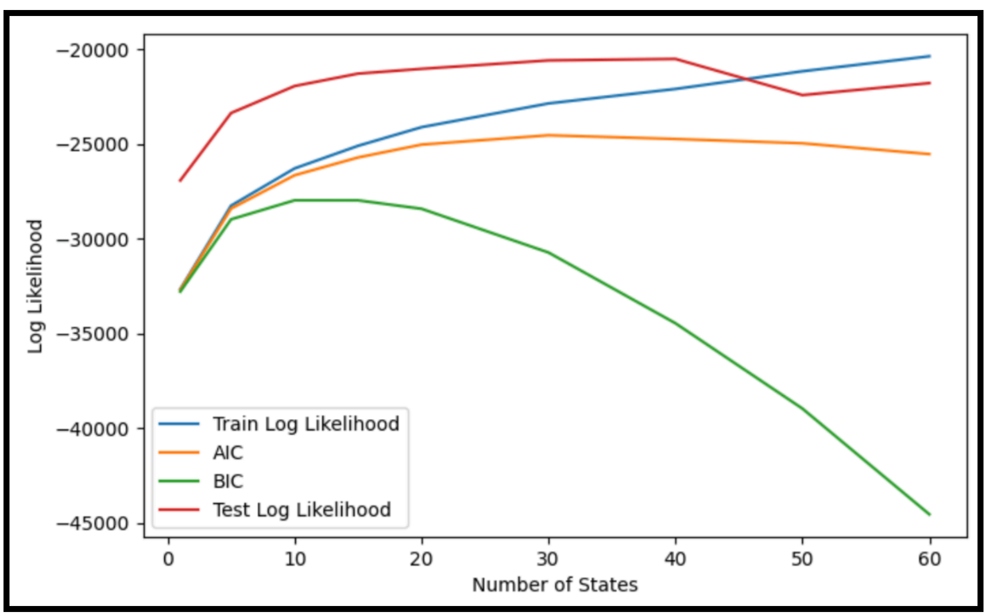
```
import numpy as np
import random
import matplotlib.pyplot as plt
from text_utils import *
from hmmlearn import hmm

load_from_saves = False
model_to_save = False

# Load the data
train_text = read_text_file("aliceTrain.txt")
test_text = read_text_file("aliceTest.txt")
train_data = text_to_num(train_text)
test_data = text_to_num(test_text)

num_valid_chars = 27
max_num_em_steps = 500
em_change_tolerance = 1e-6

m_values = [1, 5, 10, 15, 20, 30, 40, 50, 60]
models = []
training_log_likelihoods = []

if model_to_save:
   for M in m_values:
       model = hmm.CategoricalHMM(n_components=M, n_iter=max_num_em_steps,
tol=em_change_tolerance)
       model.fit(np.reshape(train_data, (-1, 1)))
       models.append(model)
       training_log_likelihoods = list(model.monitor_.history)
       model_save = {
           "n_features": model.n_features,
           "emissionprob_": model.emissionprob_,
           "transmat_": model.transmat_,
           "startprob_": model.startprob_,
           "tll": np.asarray(list(model.monitor_.history))
       }
       np.save(f"model_m_{M}.npy", model_save)
else:
```

```
    for M in m_values:
        data = np.load(f"model_m_{M}.npy", allow_pickle=True).item()
        model = hmm.CategoricalHMM(n_components=M, verbose=True)
        model.n_features = data["n_features"]
        model.emissionprob_ = data["emissionprob_"]
        model.transmat_ = data["transmat_"]
        model.startprob_ = data["startprob_"]
        t11 = data["tll"]
        models.append(model)
        training_log_likelihoods.append(t11.tolist())

AIC_train_values = np.zeros(len(m_values))
BIC_train_values = np.zeros(len(m_values))
test_log_likelihoods = np.zeros(len(m_values))
max_train_log_likelihoods = np.zeros(len(m_values))

for index_of_M in range(len(m_values)):
    M = m_values[index_of_M]
    num_param = (M * (M - 1)) + (M * (num_valid_chars - 1)) + (M - 1)
    AIC_train_values[index_of_M] =
np.max(np.asarray(training_log_likelihoods[index_of_M])) - num_param
    BIC_train_values[index_of_M] =
np.max(np.asarray(training_log_likelihoods[index_of_M])) - 0.5 * num_param *
np.log(train_data.shape[0])
    test_log_likelihoods[index_of_M] = models[index_of_M].score(np.reshape(test_data,
(-1, 1)))
    max_train_log_likelihoods[index_of_M] =
np.max(np.asarray(training_log_likelihoods[index_of_M]))

# Plot log-likelihood, AIC, BIC, and test log-likelihood
plt.plot(m_values, max_train_log_likelihoods, label="Train Log Likelihood")
plt.plot(m_values, AIC_train_values, label="AIC")
plt.plot(m_values, BIC_train_values, label="BIC")
plt.plot(m_values, test_log_likelihoods, label="Test Log Likelihood")
plt.xlabel("Number of States")
plt.ylabel("Log Likelihood")
plt.legend()
plt.show()

# Identify best models based on AIC and BIC
best_model_AIC = np.argmax(AIC_train_values)
```

```
best_model_BIC = np.argmax(BIC_train_values)

print(f"\nBest Model according to AIC: M = {m_values[best_model_AIC]}")
print(f"Best Model according to BIC: M = {m_values[best_model_BIC]}\n")

test_indices = [0, best_model_AIC, best_model_BIC, len(m_values) - 1]
test_indices.sort()
sample_size = 500

for i in test_indices:
   sample, _ = models[idx].sample(sample_size)
   sample_text = num_to_text(sample.flatten())
   print(f"Sample from HMM with M = {m_values[i]}:\n{sample_text}\n")

erase_rate = 0.2
erase_ind = np.random.uniform(size=(len(test_text),)) < erase_rate
num_erased = np.sum(erase_ind)
noisy_text = np.copy(test_text)
noisy_text[erase_ind] = "*"
noisy_data = text_to_num(noisy_text)

denoised_text_list = []
num_corrected = np.zeros(len(test_indices))

for _char, i in enumerate(test_indices):
   M = m_values[i]
   model = models[i]
   erase_prob = np.eye(num_valid_chars, num_valid_chars + 1)
   erase_prob[:, :-1] *= (1.0 - erase_rate)
   erase_prob[:, -1] = erase_rate
   obs_mat = np.matmul(model.emissionprob_, erase_prob)
   noise_fix_model = hmm.CategoricalHMM(n_components=M, verbose=True)
   noise_fix_model.n_features = num_valid_chars + 1
   noise_fix_model.emissionprob_ = obs_mat
   noise_fix_model.transmat_ = model.transmat_
   noise_fix_model.startprob_ = model.startprob_
   noisy_prob = noise_fix_model.predict_proba(np.reshape(noisy_data, (-1, 1)))

   denoised_data = np.copy(noisy_data)
      # Find the most probable missing letters
   for i in range(noisy_data.shape[0]):
```

```
        if noisy_text[i] == "*":
            letter_probability = np.matmul(model.emissionprob_.T, noisy_prob[i])
            letter = np.argmax(letter_probability)
            denoised_data[i] = letter

    num_corrected[_char] = num_erased - np.sum(denoised_data != test_data)
    denoised_text
    denoised_text = num_to_text(denoised_data.flatten())
    denoised_text_list.append(denoised_text)

sample_size = 500

print("Denoised Outputs and Correction Metrics")
print("Original Noisy Text:")
print("".join(noisy_text[:sample_size].tolist()))
print()

for i in range(len(test_indices)):
    percent_correct = (float(num_corrected[i]) / float(num_erased)) * 100
    print(f"M = {m_values[test_indices[i]]} | Letters Corrected:
{int(num_corrected[i])} | Percentage Corrected: {percent_correct:.2f}%")
    print(denoised_text_list[i][:sample_size])
```

A) Plot:



B)

- The number of free parameters in a multinomial hidden Markov model (HMM) consists of the initial state distribution, the state transition matrix, and the emission distribution matrix.
- Including all of them, the number of free parameters is $d = (M-1) + M(M-1) + M(W-1)$.
- As plotted in the above graph, the Bayesian Information Criterion (BIC) imposes a larger penalty on the number of parameters than the Akaike Information Criterion (AIC).
- Consequently, the BIC favors simpler models with fewer parameters compared to the AIC, which is more lenient towards complex models with a larger number of parameters.

C)

- AIC is largest for M=30 and BIC is largest for M=10.
- Test log-likelihood is largest for =30s predicted by AIC
- But it's very low for M=10
- In this case, AIC performed better.
- Theoretical results suggest that AIC is often better for moderate datasets, but it can be inconsistent, and thus inferior to BIC for very large datasets.

---

d):

HMM with M = 1

```
seuhhs_aeo__sls_ended_oshiuka_na_ms__eea_wel_a_ernl_is__eani_wbe_lmte__oeauhi_r
uinstg_h_oh_ieo_ywttie_abytos_hao_oui_ietdeierdysoehitashi_r_ntp_edes_hcresestm
awweesia_oss_onco_htgnhheaefverr__d_ahewia_tahnsa_ait___h_elt_n_ennehhmsrdi_e_k
nnwnaug_l_uglienarlii_st_o_m___gttryotat_enisget_ednnak_py__er_lees__igiposiiu_
ylvpcio_er_e_dotovweauxmaeirehelrregleo_prasedts_mwtd_haaueleegr___ie_so_rd_rut
mi__idlut_eteteiciiandiba_te__daes_oe_m_whrfwmonlste_out_irl_icesu_eca__oeu_se_
_et__h_ee_saddormnno_j_ywe
```

HMM with M = 20

```
yonedfeave_dailg_ce_at_anrefire_hut_ande_howde_we_cor_sousonlg_eesbe_tid_yer_mh
ass_aalem_oy_ant_doy_br_tre_linomet_ir_tawf_tal_anl_any_tovilf_bplee_aluch_wind
_a_thayestind_bis_he_attrith_bet_mve_he_host_mro_the_thy_limy_dhetterthowkesdik
e_i_msk_toot_yautef_be_bre_tau_alig_hit_the_maketath_tes_han_ang_it_dhe_let_tol
g_ad_hacy_hat_ikt_swe_hepd_an_sure_asuaalilt_bu_toumh_wort_shed_he_tory_pes_you
_le_ald_aplene_ta_thyatowd_it_gher_no_liowte_laa_nhime_dont_sounysrent_tomo_he_
shirn_an_shar_wint_pbe_ggl
```

HMM with M = 30

```
plenned_thint_i_den_tangt_i_thymwas_le_thastine_alld_bwer_the_tee_widis_dene_ni
_hare_thed_to_plot_uouoy_dor_os_a_mastingn_vosed_ot_hasicg_mars_ttdce_outh_alit
e_mavle_hutisk_maplinh_i_pat_letreadxot_i_ore_sar_coemsar_ud_the_shicg_en_it_ou
led_nersawe_surlayhats_maid_a_doene_putweriersads_miicg_the_whe_feavine_neay_id
_thkd_udhad_wug_o_satse_qhe_ereptor_a_hand_muld_sas_ot_sadint_therdnw_a_dis_an_
iss_yoy_trerie_verius_ot_isper_thklece_che_cup_hor_imyd_y_wououed_lin_i_i_loagai
d_dem_inwond_sveand_itfe_d
```

HMM with M = 60

```
tauebed_i_cany_the_yortitert_astring_at_lepse_getghked_mapt_a_rile_sormoulded_a
s_to_alice_sughts_that_om_is_ther_fou_sint_ok_the_hith_thruest_the_nou_yoct_on_
untes_co_thes_vavy_be_she_she_vamh_noulk_on_it_a_ksovene_il_moull_alike_ware_no
r_cou_yen_you_lubp_evothe_come_it_to_thint_hetned_the_soup_emeng_twer_thed_tork
e_hety_hath_shoy_mo_mou_reep_shing_on_up_med_at_dcace_on_alace_tave_trked_beeke
r_dirnie_the_mis_i_said_raad_usfoich_tciupon_i_ware_and_alice_wory_youryord_bus
ses_piut_twe_toce_faly_rem
```

- The unigram model captures only the basic frequencies of letters.
- More complicated models learn states which emit a number of common short words, but are still very far from real English text.
- A much larger training corpus would be needed to learn more realistic models.

---

e) Code:

```
erase_rate = 0.2
erase_ind = np.random.uniform(size=(len(test_text),)) < erase_rate
num_erased = np.sum(erase_ind)
noisy_text = np.copy(test_text)
noisy_text[erase_ind] = "*"
noisy_data = text_to_num(noisy_text)

denoised_text_list = []
num_corrected = np.zeros(len(test_indices))

for _char, i in enumerate(test_indices):
   M = m_values[i]
   model = models[i]
   erase_prob = np.eye(num_valid_chars, num_valid_chars + 1)
   erase_prob[:, :-1] *= (1.0 - erase_rate)
   erase_prob[:, -1] = erase_rate
   obs_mat = np.matmul(model.emissionprob_, erase_prob)
   noise_fix_model = hmm.CategoricalHMM(n_components=M, verbose=True)
   noise_fix_model.n_features = num_valid_chars + 1
   noise_fix_model.emissionprob_ = obs_mat
   noise_fix_model.transmat_ = model.transmat_
   noise_fix_model.startprob_ = model.startprob_
   noisy_prob = noise_fix_model.predict_proba(np.reshape(noisy_data, (-1, 1)))

   denoised_data = np.copy(noisy_data)
   # Find the most probable missing letters
```

```
    for i in range(noisy_data.shape[0]):
        if noisy_text[i] == "*":
            letter_probability = np.matmul(model.emissionprob_.T, noisy_prob[i])
            letter = np.argmax(letter_probability)
            denoised_data[i] = letter

    num_corrected[_char] = num_erased - np.sum(denoised_data != test_data)
    denoised_text
    denoised_text = num_to_text(denoised_data.flatten())
    denoised_text_list.append(denoised_text)
```

---

f)

- The given equations represent the marginal and conditional probability distributions, where the last equality indicates that the letters are conditionally independent given the Markov states.

$$p\left(x_t \mid y\right)=\sum_{z_t} p\left(x_t, z_t \mid y\right)=\sum_{z_t} p\left(x_t \mid z_t, y\right) p\left(z_t \mid y\right)=\sum_{z_t} p\left(x_t \mid z_t, y_t\right) p\left(z_t \mid y\right)$$

where the last equality means letters x_t are conditionally independent given the Markov states z_t.

- The result then follows because $p\left(x_t \mid z_t, y_t=*\right)=p\left(x_t \mid z_t\right)$
- This follows from the uninformative nature of $p\left(y_t=* \mid x_t\right)$.
  The marginal probability distribution of the letters depends solely on the transition probabilities between the Markov states and the emission probabilities of the letters given those states

---

g)

- The minimum-error decision rule chooses $\hat{x}_t=\arg \max_{x_t} p\left(x_t \mid y\right)$ to maximize the marginal posterior distribution from question C.
- It selects the option that has the highest marginal posterior probability, which is the probability of that option being true.

---

h)

- The models with M=1, 10, 30, and 60 states correct 19.50 %, 45.29 %, 48.62 % and 51.43% of the erased characters, respectively.
- Choosing one of the 27 characters at random would correct 3.7%.
- The unigram model improves on this by always picking the most common character, space, but the HMM models do much better:

```
Original, Noisy Text:
```

```
*he_king_an*_queen_of*he*r***w*re_*eat*d*on_thei*_thron*_wh*n_th*y*arr*ve*_with
_a_g*eat_crowd_a*sembl*d*a*out_th**_al*_sorts*of**i**l*_b**ds_*nd_beast*_a*_*el
l_as_the_whole_pack*of_card*_th*_knav*****_*tanding_bef*r*_them*i*_chains_*it*_
a_soldie**o*_each_*ide_*o_g*ar***im_*nd_*ear_*he*k**g_*a*_the*white*r*bbi*_with
_a_trumpe*_in_one*ha*d_and_a*scrol*_of_parchment*in_the_othe*_i*_th*_very*middl
e_of*the_*ourt*wa*_a_*a*l*_w**h_*_l*r*e*d*sh_o*_ta**s_upon_it_th*y*l*oked_s*_g*
od_that_it_*ade_ali*e_qui*

M = 1
Letters Corrected: 375
Percentage Corrected: 19.50%
_he_king_an__queen_of_he_r___w_re__eat_d_on_thei__thron__wh_n_th_y_arr_ve__with
_a_g_eat_crowd_a_sembl_d_a_out_th___al__sorts_of__i__l__b__ds__nd_beast__a___el
l_as_the_whole_pack_of_card__th__knav_______tanding_bef_r__them_i__chains__it__
a_soldie__o__each__ide__o_g_ar___im__nd__ear__he_k__g__a__the_white_r_bbi__with
_a_trumpe__in_one_ha_d_and_a_scrol__of_parchment_in_the_othe__i__th__very_middl
e_of_the__ourt_wa__a__a_l__w__h___l_r_e_d_sh_o__ta__s_upon_it_th_y_l_oked_s__g_
od_that_it__ade_ali_e_qui_

M = 20
Letters Corrected: 871
Percentage Corrected: 45.29%
the_king_and_queen_of_he_re__wore_heat_dhon_theit_thrond_when_they_arrevet_with
_a_gheat_crowd_atsembled_alout_thet_ale_sorts_of_tinile_breds_and_beaste_at_tel
l_as_the_whole_pack_of_carde_the_knave__he_atanding_bef_re_them_it_chains_tite_
a_soldie__ou_each_tide_to_ghar_thim_and_tear_thetking_tau_the_white_rebbie_with
_a_trumpet_in_one_hand_and_a_scrold_of_parchment_in_the_othet_it_the_very_middl
e_of_the_tourt_wau_a_tauld_wich_a_lorte_dish_ot_tares_upon_it_they_looked_so_gh
od_that_it_tade_alice_quin

M = 30
Letters Corrected: 935
Percentage Corrected: 48.62%
the_king_and_queen_of_he_re__were_deated_on_theid_throne_when_they_arr_ver_with
_a_gteat_crowd_a_sembled_alout_thed_all_sorts_of_tinkle_berds_and_beaste_at_del
l_as_the_whole_pack_of_carde_the_knave_t_e_ttanding_befere_them_it_chains_tite_
a_soldie__ot_each_aide_to_geare__im_and_dear_the_king_sad_the_white_rebbie_with
_a_trumper_in_one_hand_and_a_scrolg_of_parchment_in_the_other_it_the_very_middl
e_of_the_uourt_war_a_sakle_winh_a_larke_dish_ot_tares_upon_it_they_leoked_si_gt
od_that_it_sade_aline_quin

M = 60
Letters Corrected: 989
Percentage Corrected: 51.43%
the_king_and_queen_of_hearle_ware_beated_on_theid_throng_whin_they_arrever_with
_a_gbeat_crowd_a_sembledea_out_thet_aly_sortsiof_singly_beads_and_beaste_at_wel
l_as_the_whole_pack_of_cardh_the_knave_tae_otanding_befere_them_it_chains_wite_
a_soldie__ou_each_wide_yo_grare__im_and_bear_theeking_had_the_whiterrubbie_with
```

```
_a_trumped_in_one_haad_and_atscroly_of_parchment_in_the_othed_it_the_very_middl
e_of_the_dourt_war_a_ealle_warh_a_larte_d_sh_on_tates_upon_it_they_looked_so_go
od_that_it_wade_alice_quin
```