Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий фізико-технічний інститут

Застосування прихованих марковських моделей в задачах оцінювання, навчання та декодування

предмет «Марковські моделі та їхнє застосування»

Роботу виконав:
Студент групи ФІ-91,
Цибульник Антон Владиславович

Роботу перевірила:
Ніщенко Ірина Іванівна

2022

Зміст

Опис прихованих марковських моделей

Перш ніж переходити до розв'язків задач оцінювання, навчання та декодування, введемо декілька необхідних означень. Нехай $\{X_t\}_{t\geqslant 0}$ – ланцюг Маркова, тобто послідовність випадкових величин, які приймають значення $\{x_t\}_{t\geqslant 0}$ зі скінченної множини станів $E = \{e_1, e_2, \ldots, e_N\}$. Крім того, для вказаної послідовності справедливі властивості:

(1) Заданий початковий розподіл : $\mu_{x_0} = P(X_0 = x_0) \;\; \forall x_0 \in E$;

(2) Виконується властивість незалежності «майбутнього» від «минулого» при відомому «теперішньому»:

$\forall n \geqslant 0 \quad \forall x_0, x_1, \ldots, x_{n+1} \in E$:

$$P(X_{n+1} = x_{n+1} \,|\, X_n = x_n, \ \ldots \ , X_0 = x_0) = P(X_{n+1} = x_{n+1} \,|\, X_n = x_n)$$

Компоненти з властивості (1), які формують вектор μ, називають початковим розподілом, а сукупність імовірностей виду $P(X_{n+1} = x_{n+1} \,|\, X_n = x_n)$ для будь-якого n та довільних станів $x_n, x_{n+1} \in E$ – матрицею перехідних імовірностей A.

Зауважимо, що надалі розглядатимуться лише однорідні ланцюги Маркова, у яких перехідні ймовірності матриці A є однаковими незалежно від номера «кроку» самого ланцюга. Наприклад, $\forall n \geqslant 0$ та при фіксованих станах $x_0, x_1 \in E$:

$$P(X_1 = x_1 \,|\, X_0 = x_0) = P(X_{101} = x_1 \,|\, X_{100} = x_0) = \ldots = P(X_{n+1} = x_1 \,|\, X_n = x_0)$$

Тепер введемо послідовність випадкових величин $\{Y_t\}_{t\geqslant 0}$ зі значеннями $\{y_t\}_{t\geqslant 0}$ на скінченній множині станів $F = \{f_1, f_2, \ldots, f_M\}$. Тоді пара $\{(X_t, Y_t)\}_{t\geqslant 0}$, задана на декартовому добутку $E \times F$, є прихованою марковською моделлю за виконання таких умов:

(1) $\{X_t\}_{t\geqslant 0}$ – ланцюг Маркова;

(2) Випадкові величини $Y_0, Y_1, \ldots, Y_n$ є умовно незалежними при заданому наборі величин $X_0, X_1, \ldots, X_n$:

$\forall n \geqslant 1 \quad \forall y_0, y_1, \ldots, y_n \in F \quad \forall x_0, x_1, \ldots, x_n \in E$:

$$P(Y_0 = y_0, \ldots, Y_n = y_n \,|\, X_0 = x_0, \ldots, X_n = x_n) =$$

$$= \prod_{t=0}^{n} P(Y_t = y_t \,|\, X_0 = x_0, \ldots, X_n = x_n)$$

(3) Умовний розподіл випадкової величини Y_t при заданих $X_0, X_1, \ldots, X_n$ залежить лише від X_t :

$\forall n \geqslant 1 \quad \forall t = \overline{0, n} \quad \forall y_t \in F \quad \forall x_0, x_1, \ldots, x_n \in E$:

$$P(Y_t = y_t \,|\, X_0 = x_0, \ldots, X_n = x_n) = P(Y_t = y_t \,|\, X_t = x_t)$$

У парі $\{(X_t, Y_t)\}_{t\geqslant 0}$ послідовність $\{X_t\}_{t\geqslant 0}$ називають «прихованою», а послідовність $\{Y_t\}_{t\geqslant 0}$ – «спостережуваною». Приховану марковську модель λ з початковим розподілом μ, матрицею перехідних імовірностей A та матрицею умовних імовірностей переходу від «прихованих» станів до «спостережуваних» величин B позначатимемо так: $\lambda = (\mu, A, B)$.

Задача I (оцінювання)

Почергово розглянемо три основні задачі, які можна розв'язати за допомогою прихованих марковських моделей. У першому розділі оглянемо задачу оцінювання: за відомою прихованою марковською моделлю $\lambda = (\mu, A, B)$ й наданою протягом часу $t = \overline{0, T}$ послідовністю спостережень $y = (y_0, \ldots, y_T)_{y_t \in F}$ прагнемо визначити імовірність $P_\lambda(Y = y)$, де $Y = (Y_0, \ldots, Y_T)$.

Перетин з повною групою подій, де вектор $X = (X_0, \ldots, X_T)$ пробігає всеможливі значення $x = (x_0, \ldots, x_T)_{x_t \in E}$, дає змогу безпосередньо обчислити шукану ймовірність:

$$P_\lambda(Y = y) = P_\lambda(Y = y \bigcap \Omega) = P_\lambda(Y = y \bigcap \Big[\bigsqcup_x (X = x)\Big]),$$

а отже

$$P_\lambda(Y = y) = \sum_x P_\lambda(Y = y, X = x)$$

З огляду на властивості лінцюга Маркова, вказаний скінченновимірний розподіл виражатиметься через параметри заданої моделі λ таким чином:

$$P_\lambda(Y = y) = \sum_x \mu_{x_0} \prod_{t=0}^{T-1} A_{x_t x_{t+1}} \prod_{t=0}^{T} B_{x_t y_t}$$

Проте, за допомогою алгоритмів прямого та зворотного ходу шукану ймовірність можна обчислити іншим, більш ефективним способом.

Алгоритм прямого ходу

Алгоритм прямого ходу полягає у рекурсивному обчисленні сумісних імовірностей часткової послідовності спостережень до моменту часу t, перебуваючи в стані x_t в момент часу t :

$$\alpha_t(x_t) = P_\lambda(Y_0 = y_0, \ldots, Y_t = y_t, X_t = x_t) \tag{1}$$

Відтак крок за кроком, поклавши $\forall x_0 \in E : \alpha_0(x_0) = \mu_{x_0} B_{x_0 y_0}$, знайдемо значення усіх наступних коефіцієнтів за допомогою такого рекурентного співвідношення:

$$\alpha_{t+1}(x_{t+1}) = \sum_{x_t \in E} \alpha_t(x_t)\, A_{x_t x_{t+1}} B_{x_{t+1} y_{t+1}}$$

Тоді загальна ймовірність записуватиметься так:

$$P_\lambda(Y = y) = \sum_{x_T \in E} P_\lambda(Y_0 = y_0, \ldots, Y_T = y_T, X_T = x_T) = \sum_{x_T \in E} \alpha_T(x_T)$$

Алгоритм зворотного ходу

В алгоритмі зворотного ходу ймовірність $P_\lambda(Y = y)$ шукатиметься через рекурсивне обчислення умовних імовірностей послідовності спостережень від моменту часу $t+1$ до T при умові, що прихованим є стан x_t в момент часу t :

$$\beta_t(x_t) = P_\lambda(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T \,|\, X_t = x_t) \tag{2}$$

Поклавши $\forall x_T \in E : \beta_T(x_T) = 1$, віднайдемо значення усіх попередніх коефіцієнтів за допомогою такого рекурентного співвідношення:

$$\beta_t(x_t) = \sum_{x_{t+1} \in E} \beta_{t+1}(x_{t+1})\, A_{x_t x_{t+1}} B_{x_{t+1} y_{t+1}}$$

Відтак загальна імовірність матиме такий вид:

$$P_\lambda(Y = y) = \sum_{x_0 \in E} \beta_0(x_0)\, \mu_{x_0} B_{x_0 y_0}$$

Приклад: кмітливі учні

Розглянемо приклад розв'язування задачі оцінювання за допомогою алгоритмів прямого та зворотного ходу:

> Нехай шкільний вчитель математики протягом п'яти робочих днів задає учням домашнє завдання трьох рівнів складності: легке, складне та підвищеної складності. Водночас тип заданого завдання залежить від піднесеного, нейтрального чи поганого настрою вчителя. Як досвідчений професіонал, вчитель явно не демонструє свій настрій учням.
>
> Припустимо, що учні бажають відвідати футбольний матч, який відбудеться наступного тижня у середу. Навантаження складним домашнім завданням у ці дні було б вкрай дошкульним. В той же час, повністю розвантажити тиждень від складних завдань не вийде – викладач, скоріше за все, все ж подбає про баланс різних типів задач протягом тижня.
>
> Зібравши в уважних та кмітливих старшокласників інформацію про ймовірність зміни настрою вчителя, а також спостережуваний рівень складності заданого додому завдання в залежності від настрою викладача, учні починають дослідження: яка ймовірність спостереження наступного тижня послідовності заданих домашніх завдань бажаного рівня складності?

Формалізуємо задачу учнів таким чином: пара $\{(X_t, Y_t)\}_{t=\overline{0,5}}$ – прихована марковська модель, де $\{X_t\}_{t=\overline{1,5}}$ є прихованою послідовністю настроїв вчителя, заданих на множині станів $E = \{☺, 😐, ☹\}$, а $\{Y_t\}_{t=\overline{1,5}}$ є спостережуваною послідовністю типів завдань, заданих на множині станів $F = \{━, 🔀, 🖧\}$. Складемо матриці A, B та вектор μ згідно введених позначень:

	☺	😐	☹
☺	0.2	0.3	0.5
😐	0.2	0.2	0.6
☹	0	0.2	0.8

матриця A

	━	🔀	🖧
☺	0.7	0.2	0.1
😐	0.3	0.4	0.3
☹	0	0.1	0.9

матриця B

☺	0.05
😐	0.2
☹	0.75

вектор μ

Сформулюємо задачу оцінювання: за заданою моделлю $\lambda = (\mu, A, B)$ визначити ймовірність спостереження такої послідовності:

$$P_\lambda(Y_1 = 🖧, Y_2 = ━, Y_3 = ━, Y_4 = 🔀, Y_5 = 🔀)$$

Програматично реалізацію алгоритмів прямого та зворотного ходу наведено у Лістингах нижче (спостережені типи завдань перекодовано номерами від 1 до 3):

Лістинг 1: Алгоритм прямого ходу

```
def alpha_calculation(y,m,A,B):
    time = len(y)
    alpha = [[0.0 for i in range(len(B))] for t in range(time)]

    for t in range(time):
        for i in range(len(B)):
            if t == 0:
                alpha[t][i] = m[i]*B[i][y[t]-1]
            else:
                aA = 0
                for j in range(len(B)):
                    aA += alpha[t-1][j]*A[j][i]
                alpha[t][i] = aA*B[i][y[t]-1]

    P_alpha = 0
    for i in range(len(alpha[time-1])):
        P_alpha += alpha[time-1][i]

    return alpha, P_alpha
```

В результаті роботи алгоритму прямого ходу отримано таке значення шуканої імовірності:

$$P_\lambda(Y_1 = 🖧, Y_2 = ━, Y_3 = ━, Y_4 = 🔀, Y_5 = 🔀) = 0.0003985588$$

Лістинг 2: Алгоритм зворотного ходу

```
def beta_calculation(y,m,A,B):
    time = len(y)
    beta = [[0.0 for i in range(len(B))] for t in range(time)]

    for t in range(time-1, -1, -1):
        for i in range(len(B)):
            if t == time-1:
                beta[t][i] = 1
            else:
                sum = 0
                for j in range(len(B)):
                    sum += beta[t+1][j]*A[i][j]*B[j][y[t+1]-1]
                beta[t][i] = sum

    P_beta = 0
    for i in range(len(beta[0])):
        P_beta += beta[0][i]*B[i][y[0]-1]*m[i]

    return beta, P_beta
```

Імовірність спостереження заданої послідовності за алгоритмом зворотного ходу дорівнює:

$$P_\lambda(Y_1 = \text{🖧}, Y_2 = \text{➖}, Y_3 = \text{➖}, Y_4 = \text{🔀}, Y_5 = \text{🔀}) = 0.0003985588$$

Як бачимо, імовірності за методом прямого та зворотного ходу тотожні. Наведемо проміжні значення коефіцієнтів $\alpha_t(x_t)$ та $\beta_t(x_t)$:

$\alpha_t(x_t)$	☺	😐	☹
t=1	0.005	0.06	0.675
t=2	0.0091	0.0446	0
t=3	0.0075	0.0035	0
t=4	0.0004	0.0012	0.0006
t=5	0.0001	0.0002	0.0001

результати алгоритму прямого ходу

$\beta_t(x_t)$	☺	😐	☹
t=1	0.0018	0.0016	0.0004
t=2	0.0082	0.0073	0.0019
t=3	0.038	0.0324	0.0272
t=4	0.21	0.18	0.16
t=5	1	1	1

результати алгоритму зворотного ходу

Аналізуючи отримані результати, спершу зауважимо, що сума ймовірностей спостереження усеможливих комбінацій послідовностей заданих протягом тижня типів задач виходить рівною одиниці:

$$\sum_{y_1,y_2,y_3,y_4,y_5 \in F} P_\lambda(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, Y_4 = y_4, Y_5 = y_5) = 1$$

Тому авжеж, учнів засмутить мализна отриманої протягом їхнього дослідження імовірності. Проте при спробі підрахувати сумарну ймовірність спостереження легких задач у вівторок та середу при довільних типах задач в інші дні, результат виявиться значно кращим, але все ще не надто обнадійливим:

$$\sum_{y_1,y_4,y_5\in F} P_\lambda(Y_1 = y_1, Y_2 = \blacksquare, Y_3 = \blacksquare, Y_4 = y_4, Y_5 = y_5) = 0.02035$$

Наостанок, надзвичайно невтішним для школярів виявиться той факт, що серед імовірностей спостереження усеможливих послідовностей заданих домашніх завдань, найбільшою виявиться імовірність саме такого ланцюжка:

$$P_\lambda(Y_1 = \blacksquare, Y_2 = \blacksquare, Y_3 = \blacksquare, Y_4 = \blacksquare, Y_5 = \blacksquare) = 0.25559$$

Задача II (навчання)

У другому розділі розглянемо розв'язок задачі навчання: при заданій протягом часу $t = \overline{0,T}$ послідовності спостережень $y = (y_0, \ldots, y_T)_{y_t\in F}$ слід визначити таку модель $\lambda = (\mu, A, B)$, яка максимізує ймовірність спостереження $P_\lambda(Y = y)$ вказаної послідовності y. Інакше кажучи, прагнемо знайти параметри моделі μ, A та B, які найкраще пояснюють отримані спостереження:

$$\lambda^* = arg\max_\lambda P_\lambda(Y = y)$$

Означимо такі допоміжні величини:

$$\begin{gathered}\gamma_t(x_t, x_{t+1}) = P_\lambda(X_t = x_t, X_{t+1} = x_{t+1} \,|\, Y = y) \\ \gamma_t(x_t) = P_\lambda(X_t = x_t \,|\, Y = y) = \sum_{x_{t+1}\in E} \gamma_t(x_t, x_{t+1})\end{gathered} \tag{3}$$

Величина $\gamma_t(x_t, x_{t+1})$ є ймовірністю того, що в момент t відбувся перехід в часі зі стану x_t у стан x_{t+1} при заданій послідовності спостережень y. А $\gamma_t(x_t)$ є сумою ймовірностей переходу від стану x_t в усі можливі стани x_{t+1} при відомому y. Зазначимо, що означені допоміжні величини можна виразити через введені у формулах (1) й (2) ймовірності $\alpha_t(x_t)$ та $\beta_t(x_t)$:

$$\begin{aligned}\gamma_t(x_t, x_{t+1}) = P_\lambda(X_t = x_t, X_{t+1} = x_{t+1} \,|\, Y = y) &\overset{\text{def}}{=} \frac{P_\lambda(X_t = x_t, X_{t+1} = x_{t+1}, Y = y)}{P_\lambda(Y = y)} = \\ &= \frac{\alpha_t(x_t)\, A_{x_t x_{t+1}} B_{x_{t+1} y_{t+1}}\, \beta_{t+1}(x_{t+1})}{P_\lambda(Y = y)} \\ \gamma_t(x_t) = P_\lambda(X_t = x_t \,|\, Y = y) &\overset{\text{def}}{=} \frac{P_\lambda(X_t = x_t, Y = y)}{P_\lambda(Y = y)} = \frac{\alpha_t(x_t)\beta_t(x_t)}{P_\lambda(Y = y)}\end{aligned}$$

Ймовірність $P_\lambda(Y = y)$ легко оцінити за допомогою алгоритмів прямого чи зворотного ходу, розглянутих у попередньому розділі.

Алгоритм Баума-Велша

Тож за допомогою усіх введених величин, починаючи з деякої апріорної початкової моделі $\lambda^0 = (\mu^0, A^0, B^0)$, можна означити алгоритм Баума-Велша для переоцінення заданої моделі таким чином:

1. Обчислюємо по формулам (1), (2) та (3)

$$\begin{aligned} &\forall t = \overline{0, T},\ \forall x_t \in E : && \alpha_t(x_t),\ \beta_t(x_t),\ \gamma_t(x_t) \\ &\forall t = \overline{0, T-1},\ \forall x_t, x_{t+1} \in E : && \gamma_t(x_t, x_{t+1}) \end{aligned}$$

2. Переоцінюємо параметри моделі

$$\begin{aligned} &\forall x_0 \in E : && \mu_{x_0} = \gamma_0(x_0) \\ &\forall i, j \in E : && A_{ij} = \sum_{t=0}^{T-1} \gamma_t(i,j) \Big/ \sum_{t=0}^{T-1} \gamma_t(i) \\ &\forall i \in E,\ \forall j \in F : && B_{ij} = \sum_{t=\overline{0,T},\, y_t = j} \gamma_t(i) \Big/ \sum_{t=0}^{T} \gamma_t(i) \end{aligned}$$

3. Обчислюємо $P_\lambda(Y = y)$ за переоціненою моделлю. Якщо ймовірність зросла менше, ніж на деяке наперед задане число ε, то алгоритм зупиняємо.

Приклад: кмітливі учні

За допомогою алгоритму Баума-Велша розв'яжемо задачу навчання на розглянутому у першому розділі прикладі:

> Нехай наміри учнів незмінні: вони мають бажання відвідати футбольний матч, який відбудеться наступного тижня у середу. Проте, згідно результатів першого дослідження, учитель математики навряд чи сприятиме задуму учнів розвантажити тиждень від складного домашнього завдання.
>
> Справді, математика – складний предмет, проте не єдиний у школі. Викладачі з інших дисциплін так само задають легке, складне чи підвищеної складності домашнє завдання в залежності від піднесеного, нейтрального чи поганого настрою. Тож навіть якщо домашнє завдання з математики виявиться складним, за наявності легких завдань з решти предметів учням вдасться вправно перерозподілити час й виконати все в бажані терміни. То хто з викладачів найбільш імовірно задасть ланцюжок домашніх завдань бажаного рівня складності?

Нехай візьмемо за початкову модель $\lambda^0 = (\mu^0, A^0, B^0)$ дані стосовно вчителя математики:

	☺	😐	☹
☺	0.2	0.3	0.5
😐	0.2	0.2	0.6
☹	0	0.2	0.8

матриця A^0

☺	0.7	0.2	0.1
😐	0.3	0.4	0.3
☹	0	0.1	0.9

матриця B^0

☺	0.05
😐	0.2
☹	0.75

вектор μ^0

Користуючись реалізацією алгоритмів прямого та зворотного ходу (Лістинги 1 та 2), а також програмою обчислення коефіцієнтів $\gamma_t(x_t)$ й $\gamma_t(x_t, x_{t+1})$, наведеною нижче, проведемо процес переоцінки параметрів моделі λ^0 з метою максимізації ймовірності спостереження такої послідовності:

$$P_\lambda(Y_1 = \text{  }, Y_2 = \text{}, Y_3 = \text{}, Y_4 = \text{  }, Y_5 = \text{})$$

Отже, програмний код матиме вид (знову ж таки, спостережені типи завдань перекодовано номерами від 1 до 3):

Лістинг 3: Обчислення коефіцієнтів $\gamma_t(i)$ та $\gamma_t(i,j)$

```
def gamma(y,A,B,alpha,beta,P):
    time = len(y)
    gamma_i = [[0.0 for i in range(len(B))] for t in range(time)]

    for t in range(time):
        for i in range(len(B)):
            gamma_i[t][i] = alpha[t][i]*beta[t][i]/P

    gamma_ij = [[[0.0 for j in range(len(B))] for i in range(len(B))] for t in
       range(time-1)]

    for t in range(time-1):
        for i in range(len(B)):
            for j in range(len(B)):
                gamma_ij[t][i][j] =
                   alpha[t][i]*A[i][j]*B[j][y[t+1]-1]*beta[t+1][j]/P

    return gamma_i, gamma_ij
```

Виконуватимемо ітераційний процес доти, доки ймовірність зростатиме більше, ніж на задане число $\varepsilon = 0.001$. Сама функція переоцінки параметрів виглядатиме таким чином:

Лістинг 4: Переоцінення параметрів моделі

```
    def reestimation(y,m,A,B,gamma_i,gamma_ij):
        time = len(y)

        for i in range(len(B)):
            m[i] = gamma_i[0][i]

        for i in range(len(B)):
            for j in range(len(B)):
                sum_i, sum_ij = 0, 0
                for t in range(time-1):
                    sum_ij += gamma_ij[t][i][j]
                    sum_i += gamma_i[t][i]
                A[i][j] = sum_ij/sum_i

        for i in range(len(B)):
            for j in range(len(B[0])):
                sum_up, sum_down = 0, 0
                for t in range(time):
                    if y[t] == j+1: sum_up += gamma_i[t][i]
                    sum_down += gamma_i[t][i]
                B[i][j] = sum_up/sum_down

        return m, A, B
```

Проміжні результати протягом $n = 13$ ітерацій наведені у таблиці нижче. Відзначимо, що на перших ітераціях прирости ймовірностей зростають повільно (навіть спершу спадають), проте потім знову набирають значення. Власне, саме тому в алгоритмі Баума-Велша для зупинки навчання радять окрім встановлення порогової величини ε задавати й певне число ітерацій, необхідних для виконання.

Ітерація	Значення ймовірності	Приріст ймовірності
$k = 0$	0.000 399	
$k = 1$	0.013 765	0.013 366
$k = 2$	0.030 787	0.017 022
$k = 3$	0.046 890	0.016 103
$k = 4$	0.061 967	0.015 077
$k = 5$	0.076 644	0.014 677
$k = 6$	0.095 408	0.018 765
$k = 7$	0.132 333	0.036 925
$k = 8$	0.186 310	0.053 977

Ітерація	Значення ймовірності	Приріст ймовірності
$k = 9$	0.228 093	0.041 783
$k = 10$	0.243 223	0.015 130
$k = 11$	0.247 040	0.003 817
$k = 12$	0.248 180	0.001 140

Отже, на момент останньої ітерації значення шуканої ймовірності дорівнює:

$$P_\lambda(Y_1 = \text{🖧}, Y_2 = \text{➖}, Y_3 = \text{➖}, Y_4 = \text{🔀}, Y_5 = \text{🔀}) = 0.24818$$

При цьому переоцінена модель λ^* отримала такий вигляд:

	😊	😐	☹
😊	1	0	0
😐	0.53	0.47	0
☹	0	1	0

матриця A^*

	➖	🔀	🖧
😊	0.05	0.95	0
😐	0.99	0.01	0
☹	0	0	1

матриця B^*

😊	0
😐	0
☹	1

вектор μ^*

Тож у співпраці зі старшокласниками учні можуть отримати «математичні портрети» усіх викладачів своєї школи, а відтак на завершальний етап дослідження школярам залишається лише співставити «портрет», отриманий в результаті алгоритму навчання, із усіма наявними.

Зауважимо, що алгоритм Баума-Велша як окремий випадок EM-алгоритму є чутливим до вибору початкових параметрів, тобто моделі λ^0. Тому для чистоти експерименту учням, мабуть, не варто прив'язуватися до учителя математики, а натомість надати на вхід певний усереднений «портрет» викладача їхньої школи.

Наприклад, наведемо близькі до рівномірних параметри початкової моделі λ^0 (що не обов'язково відповідає випадку конкретної школи) такого виду:

	😊	😐	☹
😊	0.4	0.3	0.3
😐	0.3	0.4	0.3
☹	0.3	0.3	0.4

матриця A^0

	➖	🔀	🖧
😊	0.4	0.3	0.3
😐	0.3	0.4	0.3
☹	0.3	0.3	0.4

матриця B^0

😊	0.3
😐	0.4
☹	0.3

вектор μ^0

Алгоритм навчання при заданому $\varepsilon = 0.001$ за таку ж кількість ітерацій $n = 13$ максимізував шукану ймовірність до значення

$$P_\lambda(Y_1 = \text{🖧}, Y_2 = \text{➖}, Y_3 = \text{➖}, Y_4 = \text{🔀}, Y_5 = \text{🔀}) = 0.24689$$

При цьому модель λ^* має вигляд:

	😊	😐	☹
😊	0.46	0.54	0
😐	0	1	0
☹	1	0	0

матриця A^*

	➖	🔀	🖧
😊	0.99	0.01	0
😐	0.06	0.94	0
☹	0	0	1

матриця B^*

😊	0
😐	0
☹	1

вектор μ^*

Приклад: задача кластеризації

Розглянемо інший приклад застосування алгоритму Баума-Велша. Нехай надано великий уривок тексту (кількість символів $T \approx 50\ 000$), заданий на деякому алфавіті. Оголосимо, що кожен символ у цьому тексті слід віднести до однієї з кількох різних категорій.

Формалізуємо задачу таким чином: пара $\{(X_t, Y_t)\}_{t=\overline{0,T}}$ – прихована марковська модель. Послідовність прихованих станів $\{X_t\}_{t=\overline{0,T}}$ задана на множині категорій $E = \{e_1, e_2, \ldots, e_N\}$, а послідовність спостережуваних символів $\{Y_t\}_{t=\overline{0,T}}$ задана на алфавіті $F = \{f_1, f_2, \ldots, f_M\}$ деякої мови.

Суть задачі кластеризації літер зводитиметься до аналізу переоціненої в результаті алгоритму Баума-Велша певної початкової моделі $\lambda^0 = (\mu^0, A^0, B^0)$, сформованої за припущеннями про наданий текст. Проте, перш ніж рухатися безпосередньо до кластеризації, слід врахувати деякі зміни в роботі самого алгоритму. Розглянемо ці зміни у двох наступних підрозділах: «Шкалювання алгоритмів прямого та зворотного ходу» й «Критерій зупинки алгоритму».

Шкалювання алгоритмів прямого та зворотного ходу

Перш за все, через значну довжину вектора спостережуваних станів (тобто в силу того, що текст складається з великої кількості символів), значення коефіцієнтів $\alpha_t(x_t)$ і $\beta_t(x_t)$ з кожною наступною ітерацією будуть стрімко прямувати до нуля, що можна помітити у таблицях проміжних результатів алгоритмів прямого та зворотного ходу (сторінка 6).

Відтак не кожна програма буде в змозі оперувати з величинами такої мализни. Одним з рішень цієї проблеми є процедура шкалювання (ще кажуть нормування) відповідних ймовірностей.

Отже, одразу після обчислення на черговому кроці t коефіцієнтів за формулами (1) та (2) додають крок переоцінки:

$$\widehat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum\limits_{j \in E} \alpha_t(j)}, \;\; \widehat{\beta}_t(i) = \frac{\beta_t(i)}{\sum\limits_{j \in E} \beta_t(j)}$$

А тоді коефіцієнти (3) слід обчислювати наступним чином:

$$\gamma_t(i) = \frac{\widehat{\alpha}_t(i)\,\widehat{\beta}_t(i)}{\sum\limits_{j \in E} \widehat{\alpha}_t(j)\,\widehat{\beta}_t(j)}, \;\; \gamma_t(i,j) = \frac{\widehat{\alpha}_t(i)\,A_{ij}\,B_{jy_{t+1}}\,\widehat{\beta}_{t+1}(j)}{\sum\limits_{k \in E}\sum\limits_{w \in E} \widehat{\alpha}_t(k)\,A_{kw}\,B_{wy_{t+1}}\,\widehat{\beta}_{t+1}(w)}$$

Процедура нормування має декілька різних модифікацій. Наприклад[1], в одному з варіантів на кроці t коефіцієнти спершу алгоритму прямого, а за ними й коефіцієнти алгоритму зворотного ходу шкалюють однаковими нормуючими множниками такого виду:

$$c_t = \frac{1}{\sum\limits_{j \in E} \alpha_t(j)} \tag{4}$$

Водночас, інші дослідження[2] стверджують, що такі зміни не внесуть значної переваги. Зокрема, у ході власних спостережень також не вдалося помітити значущої відмінності отриманих результатів чи різниці швидкості збіжності алгоритму в залежності від способу нормування.

Критерій зупинки алгоритму

Стосовно критерію зупинки алгоримту навчання, то, враховуючи модифікації коефіцієнтів $\alpha_t(x_t)$ та $\beta_t(x_t)$, замість відстеження від ітерації до ітерації безпосередніх приростів імовірностей спостереження бажаної послідовності, слід обчислювати зміну відповідних логарифмів імовірностей, побудованих через коефіцієнти нормування (4) таким чином:

$$\ln P(Y = y) = -\sum_{t=0}^{T} \ln c_t \tag{5}$$

Отже, процес переоцінки параметрів моделі можна припиняти тоді, коли приріст логарифмів імовірностей припиняє перевищувати деяке наперед задане число ε.

В межах задачі поділу літер деякого алфавіту на кілька різних категорій, визначимо оптимальне порогове значення ε через дослідження збіжності результатів самої кластеризації. Інакше кажучи, необхідно знайти величину приросту $\Delta \ln P(Y = y)$, за якої модель досягає такого стану, що результати задачі кластеризації перестають істотно змінюватися.

[1] Mikael Nilsson, «First Order Hidden Markov Model: Theory and Implementation Issues»

[2] Mark Hasegawa-Johnson, «Lecture 14: Log Viterbi and Scaled Forward-Backward»

Побудуємо аналіз збіжності результатів задачі кластеризації таким чином: замість введення міри схожості результатів двох послідовних ітерацій, введемо функцію подібності для поточного результату та деякого еталонного виходу, отриманого заздалегідь протягом роботи дуже великої кількості ітерацій ($n = 200, 400, 600$). Доречність вибору такого способу відстеження збіжності буде наочно продемонстрована у наступних підрозділах.

Тож нехай результати поточного групування літер на деякій ітерації n позначимо як $\mathbf{G^n} = (\mathbf{G}_i^n)_{i \in E}$, а еталонні групи – як $\mathbf{G}^* = (\mathbf{G}_i^*)_{i \in E}$. Тоді функцію подібності δ між результатами $\mathbf{G^n}$ та $\mathbf{G}^*$ покладемо так:

$$\delta(\mathbf{G^n}, \mathbf{G}^*) = \frac{1}{|E|} \cdot \sum_{i \in E} \frac{\sum\limits_{j \in \mathbf{G}_i^n} \mathbb{1}(j \in \mathbf{G}_i^*)}{\max\{|\mathbf{G}_i^n|, |\mathbf{G}_i^*|\}} \tag{6}$$

Фактично, міра (6) є середнім арифметичним між відношеннями попарного порівняння відповідних поточних та еталонних груп, тобто відсотковими співвідношеннями виду:

$$\frac{\text{кількість збігів поточного класу з еталонним}}{\text{загальна кількість літер або у поточному класі, або в еталонному}}$$

Розглянемо специфіку роботи введеної міри на такому прикладі: нехай в результаті $n = 400$ ітерацій алгоритму навчання деякої початкової моделі отримано такий розподіл літер англійського алфавіту на три кластери:

група $\mathbf{G}_{\text{I}}^*$	«a», «e», «i», «o», «q», «u»
група $\mathbf{G}_{\text{II}}^*$	«c», «d», «f», «k», «l», «m», «n», «r», «s», «v», «x», «z»
група $\mathbf{G}_{\text{III}}^*$	«b», «g», «h», «j», «p», «t», «w», «y»

Вважатимемо ці результати еталонними. Натомість, під час іншого запуску алгоритму Баума-Велша для такої ж початкової моделі на $n = 50$ ітерації отримано розподіл такого виду:

група $\mathbf{G}_{\text{I}}^{50}$	«a», «e», «i», «o», «q», «u», «x», «z»
група $\mathbf{G}_{\text{II}}^{50}$	«c», «d», «f», «k», «l», «m», «n»
група $\mathbf{G}_{\text{III}}^{50}$	«b», «g», «h», «j», «p», «t», «w», «y», «r», «s», «v»

Обчислимо відношення подібності між парою груп $\mathbf{G}_{\text{I}}^{50}$ та $\mathbf{G}_{\text{I}}^*$: кількість збігів між цими класами рівна 6, в той час як загальна кількість символів у групі $\mathbf{G}_{\text{I}}^{50}$ рівна 8. Отже, схожіть цих груп оцінюватиметься у 75%. Аналогічним чином обчислимо подібність між категоріями $\mathbf{G}_{\text{II}}^{50}$ та $\mathbf{G}_{\text{II}}^*$: кількість збігів рівна 7, а кількість символів в еталонному класі $\mathbf{G}_{\text{II}}^*$ – 12. Відповідно, відсоток схожості дорівнює 58%. Для останної пари кластерів відсоток подібності рівний 73%. Отже, міра схожості δ для груп $\mathbf{G}^{50}$ та $\mathbf{G}^*$ записуватиметься так:

$$\delta(\mathbf{G}^{50}, \mathbf{G}^*) = \frac{1}{3}(75\% + 58\% + 73\%) = 69\%$$

Отже, зважаючи на збіжні властивості алгоритму Баума-Велша, рано чи пізно міра схожості (6) зросте аж до стовідсоткового значення. Тож фіксуючи протягом роботи алгоритму прирости $\Delta \ln P(Y = y)$ та відповідні показники міри подібності, є можливість відстежити оптимальне значення порогової величини ε, за якого алгоритм навчання можна припиняти. В ключі швидкодії алгоритму, такі спостереження будуть цінними Конкретні приклади та висновки стосовно значення ε будуть розглянуті для кожної задачі кластеризації окремо.

Підготовчий етап

Озброївшись введеними модифікаціями й уточненнями алгоритму Баума-Велша, кластеризуймо літери українського алфавіту, розглянувши текст перших трьох глав роману «Місто» україномовного автора Валер'яна Підмогильного.

Проте, перш за все виконаємо підготовчий етап, залишивши в тексті лише букви одного регістру, знаки пробілу та апострофа. Наприклад, оглянемо такий уривок:

> *На вулиці йому спала раптова думка. Що коли зайти в яку велику установу? Може, там якраз випадково, потрібен молодий кмітливий рахівник чи реєстратор? Просто зайти й спитати. Це ж не гріх. Скажуть немає, то й піде. А раптом пощастить? Ця думка схвилювала його. В душі йому жила міцна надія на свою долю, бо кожному властиво вважати себе за цілком виключне явище під сонцем і місяцем. Він звернув до ґанку під великою вивіскою «Державне видавництво України» і швидким кроком зійшов на другий поверх.*

Цей же уривок тексту після «чистки»:

> *на вулиці йому спала раптова думка що коли зайти в яку велику установу може там якраз випадково потрібен молодий кмітливий рахівник чи реєстратор просто зайти й спитати це ж не гріх скажуть немає то й піде а раптом пощастить ця думка схвилювала його в душі йому жила міцна надія на свою долю бо кожному властиво вважати себе за цілком виключне явище під сонцем і місяцем він звернув до ґанку під великою вивіскою державне видавництво україни і швидким кроком зійшов на другий поверх*

Такого роду редагування можна виконати або за допомогою вбудованих функцій різних текстових редакторів, або самостійно написавши невеликий скрипт аналогічного призначення на обраній мові програмування.

Отже, на кінець підготовчого етапу отримуємо текст ($T = 45\ 070$ символів), заданий на алфавіті F довжиною $M = 35$ елементи (33 літери, пробіл та апостроф):

$$F = \{\text{«а»}, \text{«б»}, \text{«в»}, \ldots, \text{«ь»}, \text{«ю»}, \text{«я»}, \text{« »}, \text{«'»}\} \tag{7}$$

Кластеризація літер на дві категорії

Розглянемо приховану марковську модель на декартовому добутку $E \times F$, де $E = \{\text{I}, \text{II}\}$ – множина двох категорій, а F є алфавітом, отриманим після підготовчого етапу. Нехай про особливості самого алфавіту нам нічого не відомо, а тому початкову модель $\lambda^0 = (\mu^0, A^0, B^0)$ покладемо близькою до рівномірної:

	I	II
I	0.4	0.6
II	0.6	0.4

матриця A^0

	f_1	...	f_M
I	$\sim \frac{1}{M}$	...	$\sim \frac{1}{M}$
II	$\sim \frac{1}{M}$	...	$\sim \frac{1}{M}$

матриця B^0

I	0.4
II	0.6

вектор μ^0

Один із варіантів програмної реалізації заповнення елементів початкової матриці B^0 рівномірними, але водночас неоднаковими, значеннями полягає у фомуванні комірок в два кроки: спершу заповнимо елементи матриці згідно такого правила:

$$\widetilde{B}_{ij} = \tfrac{1}{M} + \xi_{ij}, \text{ де } \xi_{ij} \sim U(-0.01,\, 0.01),$$

де випадкова величина ξ має рівномірний розподіл на відрізку, при цьому математичне сподівання ξ дорівнює нулю, а дисперсія є малою. А тоді задля виконання умови стохастичності, наступним кроком зробимо нормування по рядкам $i \in E$:

$$B^0_{ij} = \frac{\widetilde{B}_{ij}}{\sum\limits_{j=1}^{M} \widetilde{B}_{ij}}$$

Результати

Проаналізуємо модель $\lambda^* = (A^*, B^*, \mu^*)$, отриману в результаті $n = 400$ ітерацій алгоритму Баума-Велша. Віднесемо кожен символ алфавіту F до одного з двох класів I або II, порівнявши для кожного елемента $j \in F$ ймовірності $B^*_{\text{I}j}$ та $B^*_{\text{II}j}$:

I категорія	«а», «е», «и», «о», «у», «ь», «я», «і»
II категорія	«б», «в», «г», «д», «ж», «з», «й», «к», «л», «м», «н», «п», «р», «с», «т», «ф», «х», «ц», «ч», «ш», «щ», «ґ», «ї», «є», «ю», « », «’»

(8)

За вийнятком окремих символів, літери з першої категорії позначають на письмі голосні звуки українського алфавіту, а літери з другої категорії – приголосні. Віднесення букв «ї», «є» та «ю» до групи II можна пояснити тим, що кожен такий символ складається з двох частин, перша з яких – приголосний звук «й».

Переоцінена матриця A^*, в свою чергу, містить інформацію щодо ймовірностей переходів між класами I та II. Відтак, перехід з голосної літери у приголосну (тобто компонента $A^*_{\text{I,II}}$) має найбільшу ймовірність, а перехід $A^*_{\text{I,I}}$ з голосної у голосну – найменшу. Такі результати цілком узгоджуються з особливостями чергування голосних та приголосних в українській мові:

	I	II
I	0.01	0.99
II	0.64	0.36

матриця A^*

Підійдемо до спроб охарактеризувати збіжні властивості алгоритму кластеризації літер на дві категорії. Користуючись мірою подібності (6), відстежимо від ітерації до ітерації схожість результатів з еталонним виходом (8):

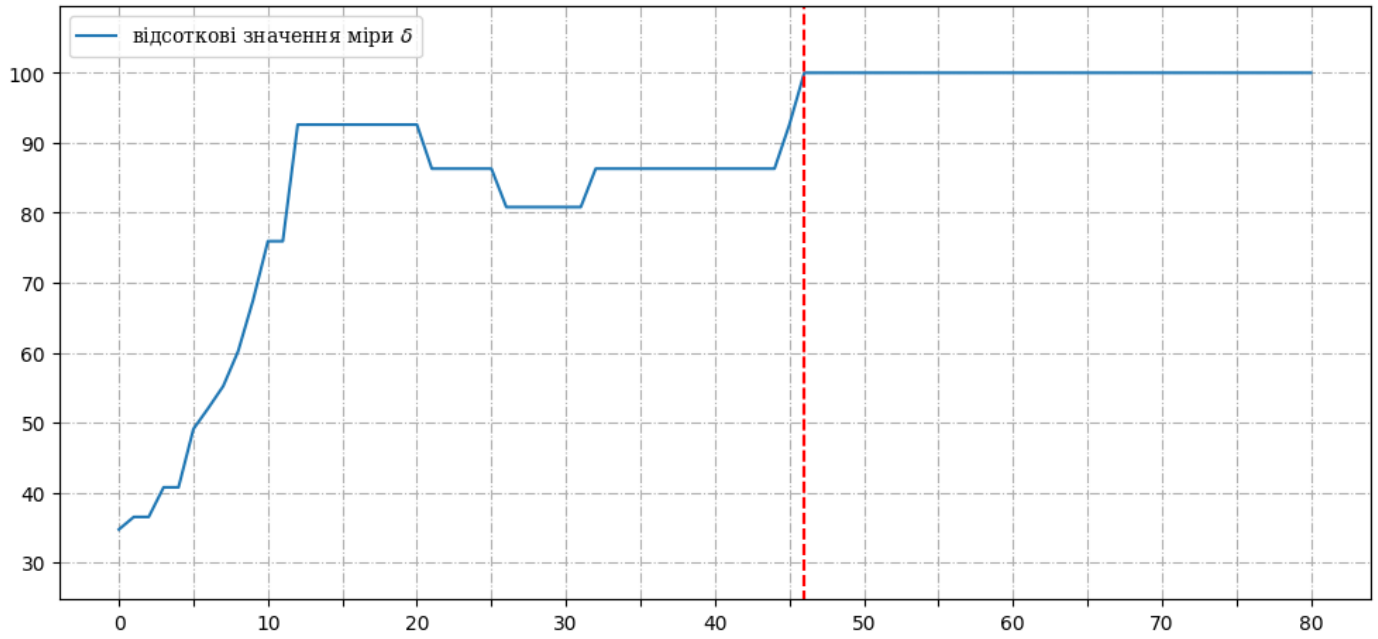

Рис. 1а: Значення δ (вісь ординат) від ітерації до ітерації (вісь абсцис)

Червоною вертикальною прямою позначена перша ітерація, коли результат збігається з еталоном. Таким чином, вже на $n = 46$ ітерації поділ літер на голосні-приголосні набуває остаточного, фінального вигляду.

Водночас, з графіку видно, що протягом значного часу (наприклад, від ітерації $n = 32$ до $n = 44$) розподіл літер лишається незмінним, але все ще недосконалим ($\approx 86\%$ схожості). Саме тому побудова міри подібності (6) ґрунтувалася на схожості поточних результатів з еталонними на противагу порівнянню сусідніх результатів кластеризації.

Фіксуючи з кожною ітерацією значення приростів $\varepsilon = \Delta \ln P(Y = y)$, отримуємо такі спостереження:

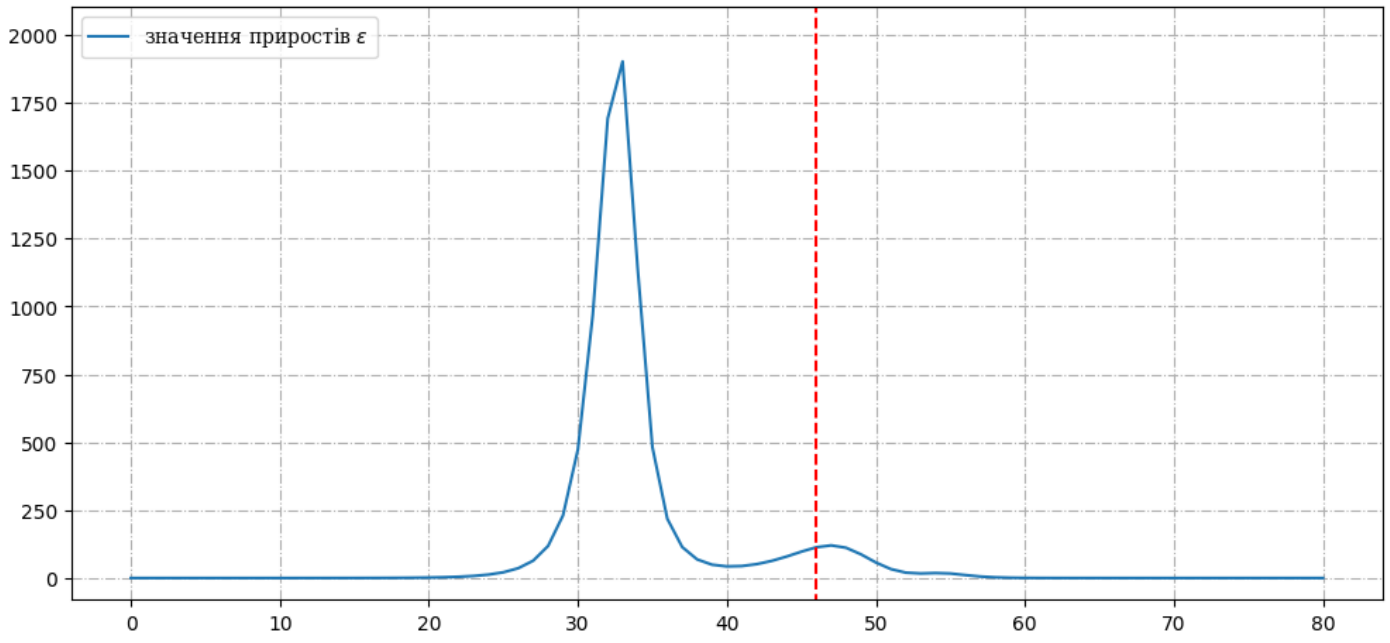

Рис. 1б: Значення ε (вісь ординат) від ітерації до ітерації (вісь абсцис)

Бачимо, що зростання починає відбуватися не одразу, алгоритму необхідний деякий час, аби «розігнатися». Саме про цей ефект йшла мова при розв'язанні прикладу кмітливих учнів (сторінка 10). Крім того, цікавим є факт, що перший збіг з еталоном припадає не на ітерації найбільшого зростання.

Оминувши інформацію про надвеликі значення приростів ε, розглянемо графік більш детально:

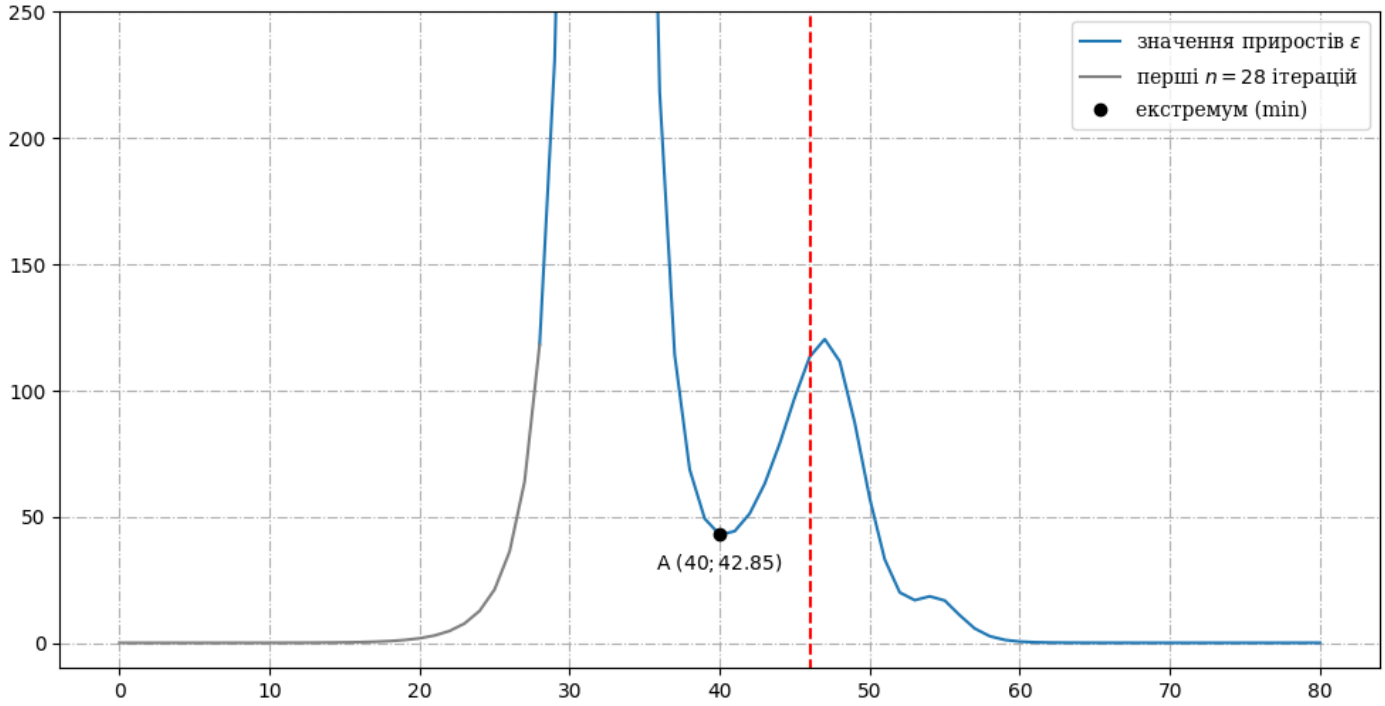

Рис. 2: Значення ε на проміжку $(0, 250)$

Проаналізувавши наведений графік, виявимо оптимальний критерій зупинки алгоритму Баума-Велша для випадку кластеризації на дві категорії. Тож нехай перші $n = 28$ ітерацій є обов'язковими до виконання (позначимо ці «ітерації для розгону» сірим кольором).

Тоді, фактично, оптимальною пороговою величиною ε^* є значення найменшого серед усіх екстремумів (а саме: мінімумів), наявних до ітерації першого збігу з еталоном. У даному випадку єдиним таким мінімумом є точка A з координатами $(40, 42.85)$, тобто на ітерації $n = 40$ зі значенням приросту $\varepsilon = 42.85$ (ця точка позначена чорною крапкою).

Отже, виконавши декілька десятків обов'язкових ітерацій, алгоритм навчання може зупинятися тоді, коли вперше спорігатиме приріст $\varepsilon^* < 42.85$.

Кластеризація літер на три категорії

Продемонструємо результати поділу літер на більш ніж два класи: введемо приховану марковську модель на множині трьох категорій $E = \{\text{I, II, III}\}$ та на отриманому раніше алфавіті (7):

$$F = \{\text{«а», «б», «в», ..., «ь», «ю», «я», « », «'»}\}$$

Початкову модель $\lambda^0 = (\mu^0, A^0, B^0)$ знову покладемо близькою до рівномірної:

	I	II	III
I	0.4	0.3	0.3
II	0.3	0.4	0.3
III	0.3	0.3	0.4

матриця A^0

	f_1	...	f_{35}
I	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$
II	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$
III	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$

матриця B^0

I	0.3
II	0.4
III	0.3

вектор μ^0

Проаналізуваши матрицю B^*, отриману після $n = 400$ ітерацій алгоритму Баума-Велша, отримуємо результати:

I категорія	«б», «в», «г», «д», «ж», «к», «л», «м», «н», «п», «р», «т», «ф», «ц», «ч», «ш», «щ», «ґ»
II категорія	«а», «е», «и», «о», «у», «ь», «я», «і», « »
III категорія	«з», «й», «с», «х», «ї», «є», «ю», «'»

(9)

Бачимо, що категорію I складають літери на позначення приголосних звуків, а категорію II – на позначення голосних. Категорія III містить приголосний «й», парні по дзвінкості букви «з» та «с», глухий «х» та букви «ї», «є», «ю», які самі по собі складаються з двох звуків, один з яких – «й».

Аналогічним до попереднього розділу чином зобразимо спостереження стосовно збіжності алгоритму кластеризації на три категорії:

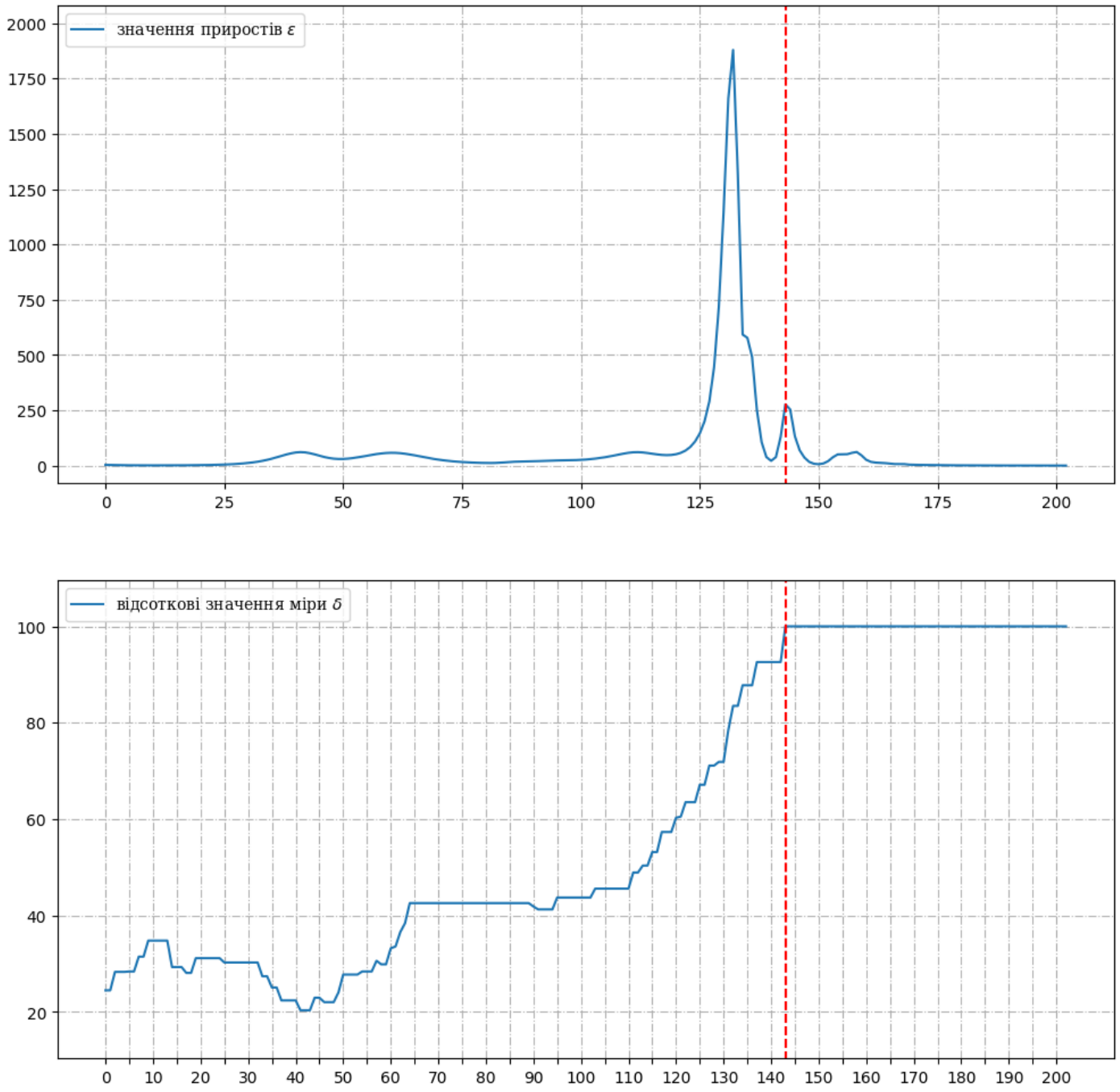

Рис. 3: Значення приростів ε та міри δ від ітерації до ітерації

Отже, вже на ітерації $n = 143$ результати задачі кластеризації збігаються з еталоном (9), причому перший збіг знову потрапляє не в множину ітерацій найбільного «піку» зростання різниць $\Delta \ln P(Y = y)$. Розглянемо детальніше графік приростів ε (Рис. 4), аби віднайти оптимальне порогове значення зупинки алгоритму навчання.

Нехай перші $n = 35$ ітерацій алгоритму покладемо обов'язковими до виконання (позначимо їх сірим кольором). Тоді значення ε^* буде визначатися як найменше число ε серед точок мінімумів, наявних до ітерації першого збігу з фінальним результатом, тобто серед точок A, B, C, D або E, позначених чорними крапками.

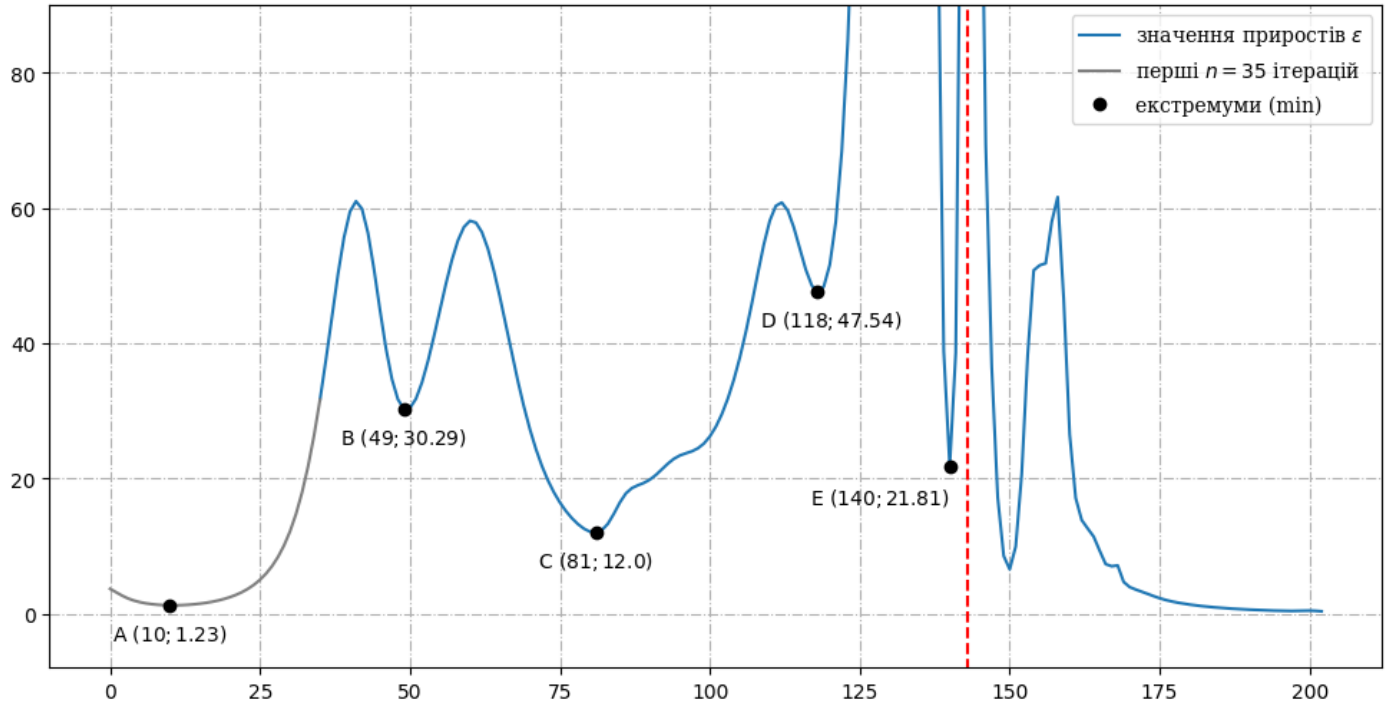

Рис. 4: Значення ε на проміжку $(0, 90)$

Однак, точка A входить у сіру область перших $n = 35$ ітерацій, а відтак, цю точку можна викреслити з множини претендентів на оптимальну порогову величину. Отже, серед множини точок, які лишилися, точка C має наменшу координату, а тому робимо висновок, що, виконавши декілька десятків обов'язкових ітерацій, алгоритм навчання можна зупиняти при першому спостереженні приросту $\varepsilon^* < 12$.

Кластеризація літер на чотири категорії

Нарешті, поділимо літери на чотири групи: розглянемо приховану марковську модель на множині чотирьох категорій $E = \{\text{I, II, III, IV}\}$ та алфавіті (7):

$$F = \{\text{«а»}, \text{«б»}, \text{«в»}, \ldots, \text{«ь»}, \text{«ю»}, \text{«я»}, \text{« »}, \text{«'»}\}$$

Початкову модель $\lambda^0 = (\mu^0, A^0, B^0)$ покладемо близькою до рівномірної:

	I	II	III	IV
I	0.3	0.2	0.2	0.3
II	0.2	0.3	0.3	0.2
III	0.3	0.2	0.2	0.3
IV	0.2	0.3	0.3	0.2

матриця A^0

	f_1	...	f_{35}
I	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$
II	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$
III	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$
IV	$\sim \frac{1}{35}$	...	$\sim \frac{1}{35}$

матриця B^0

I	0.2
II	0.3
III	0.2
IV	0.3

вектор μ^0

В результаті $n = 400$ ітерацій маємо такий розподіл:

$$
\begin{array}{ll}
\text{I категорія} & \text{«л», «н», «р», «ц», «’»} \\
\text{II категорія} & \text{«з», «й», «с», «ї», «є», «ю», « »} \\
\text{III категорія} & \text{«б», «в», «г», «д», «ж», «к», «м», «п», «т», «ф», «х», «ч», «ш», «щ», «ґ»} \\
\text{IV категорія} & \text{«а», «е», «и», «о», «у», «ь», «я», «і»}
\end{array} \tag{10}
$$

Таким чином, категорія IV має літери, які в задачі групування на два класи алгоритм відніс до голосних. Категорія III містить приголосні, II група – приголосний «й», парні по дзвінкості «з» та «с», а також букви «ї», «є», «ю». А клас I заповнений глухим приголосним «ц» та сонорними «л», «н» та «р».

Наостанок, продемонструємо збіжні властивості алгоритму:

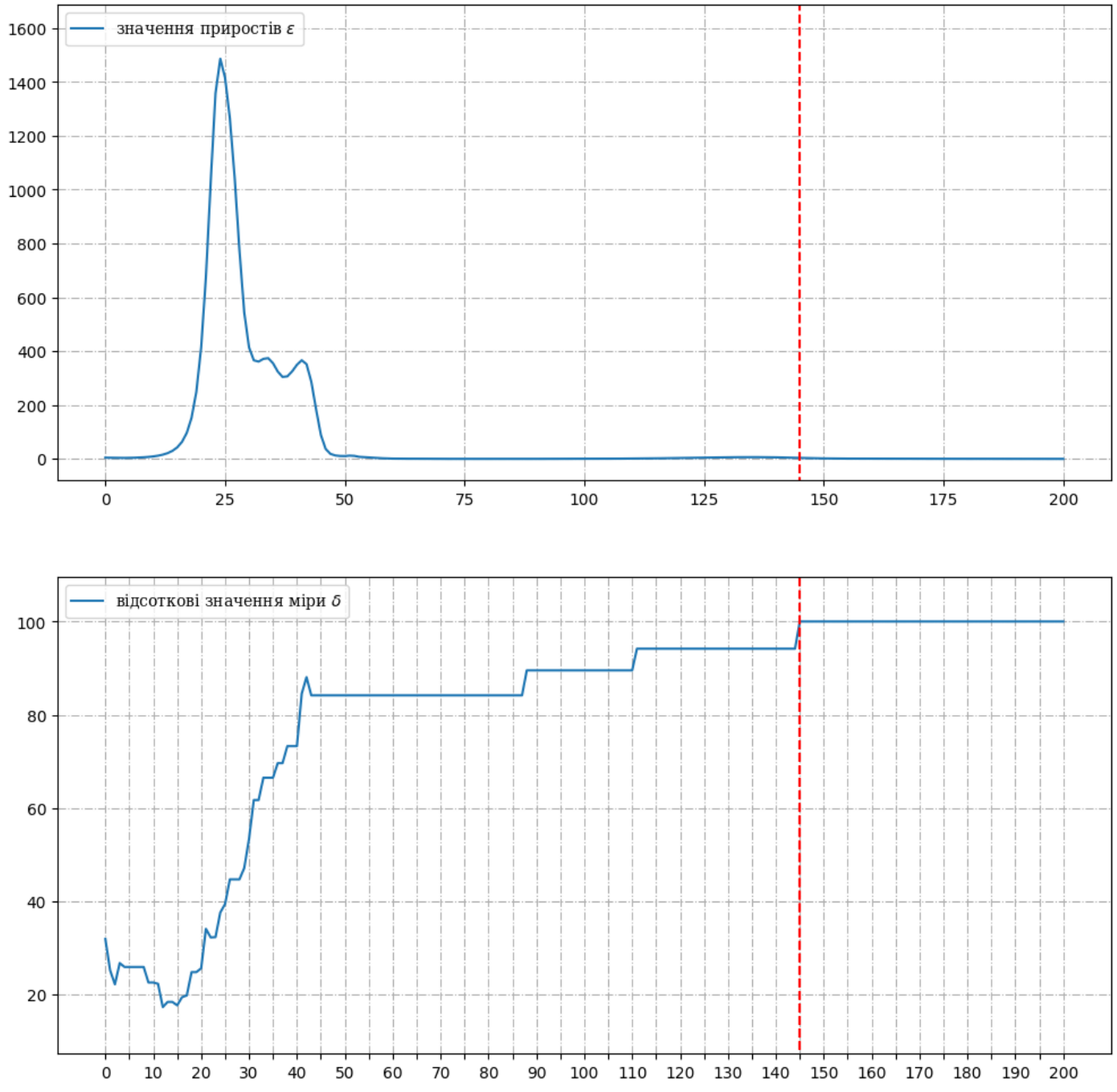

Рис. 5: Значення приростів ε та міри δ від ітерації до ітерації

Порівнявши швидкості збіжності на Рис. 3 та Рис. 5, можна зауважити, що хоч алгоритм поділу на три та чотири кластери збігся за майже аналогічну кількість ітерацій ($n = 143$ проти $n = 145$), характер збіжності алгоритмів зовсім різний, що наочно проглядається при порівнянні графіків приростів ε та значень міри δ.

З графіку приростів ε (Рис. 6) робимо висновок, що серед наявних мінімумів у допустимих точках B, C, D, E та F, найменшим є значення приросту у точці F :

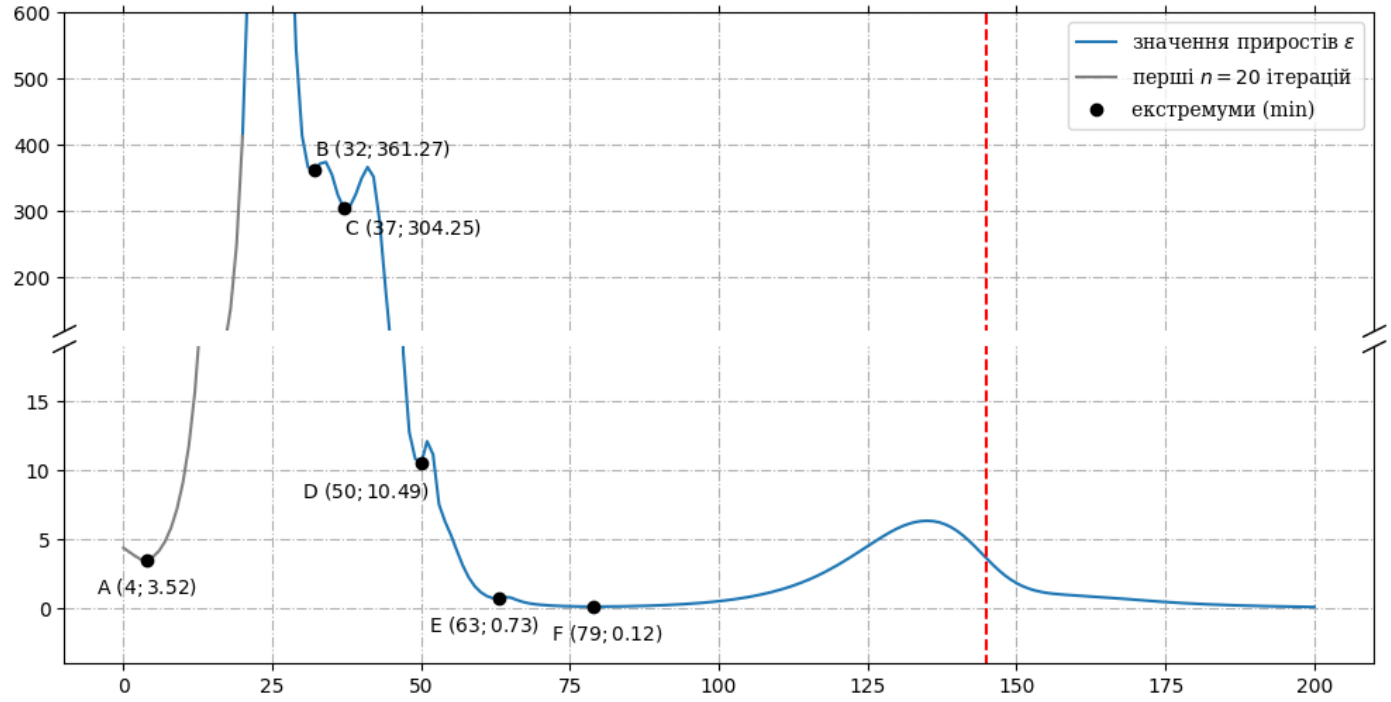

Рис. 6: Значення ε на проміжках $(0, 15)$ та $(200, 600)$

Отже, виконавши перші декілька десятків ітерацій, алгоритм Баума-Вешла можна зупиняти при першому спостереженні приросту $\varepsilon^* < 0.12$.

Результати та прикінцеві зауваження

Використовуючи алгоритм навчання Баума-Велша, було отримано такі результати задачі кластеризації на дві, три та чотири кластери:

Категорії	Модель λ^0	Результати кластеризації	Ітерація збіжності	Критерій зупинки алгоритму
I, II	стр. 16	розподіл 8	$n = 46$	$\varepsilon^* < 42.85$
I, II, III	стр. 19	розподіл 9	$n = 143$	$\varepsilon^* < 12$
I, II, III, IV	стр. 21	розподіл 10	$n = 145$	$\varepsilon^* < 0.12$

Табл. 1: Результати задачі кластеризації

Зауважимо, що наведені результати справедливі лише для вказаних моделей λ^0, зокрема для фіксованих векторів μ^0, матриць A^0 та B^0. Більш того, оскільки рівномірні матриці B^0 формуються випадковим чином (згідно правил, наведених на стр. 16), в інших дослідників не вийде відтворити графіки збіжності алгоритмів (Рис. 1а, 1б, 3 та 5) з точністю до аналогічної кількості екстремумів чи відповідних значень ε^*.

Проте, виконавши значну кількість рестартів для B^0, спостереження стосовно загальних тенденцій, притаманних розглянутим алгоритмам кластеризації, можуть бути сформульовані таким чином:

Категорії	Перші n обов'язкових ітерацій	Ітерація збіжності	Критерій зупинки алгоритму
I, II	$n \sim 50$	$n \sim 100$	$\varepsilon^* < 0.4$
I, II, III		$n \sim 200$	
I, II, III, IV		$n \sim 200$	

Табл. 2: Узагальнені спостереження

Токаж варто зазначити, чому на жодній з візуалізацій цього роздлілу не відслідковується збіжність алгоритму впритул до еталонних $n = 400$ ітерацій: після $n = 80$ ітерацій у випадку двох категорій чи після $n = 200$ ітерацій у випадках трьох та чотирьох категорій, прирости ε або стрімко спадають, або якщо і мають певні точки екструмумів, то ці значення нехтовно малі.

Наостанок, наведемо у таблиці нижче порівняльну характеристику часу виконання алгоритмів кластеризації за, наприклад, $n = 100$ ітерацій:

Кількість ітерацій	Кількість категорій	Час виконання алгоритму
$n = 100$	I, II	3,28 хв
	I, II, III	5,51 хв
	I, II, III, IV	8,36 хв

Табл. 3: Порівняльна таблиця часу виконання алгоритму

Приклад: дешифрування тексту

Іншим прикладом використання алгоритму Баума-Велша в задачах кластеризації може слугувати декодування зашифрованого (наприклад, шифром Цезаря) тексту, кількість символів в якому $T \approx 100\,000$. Суть завдання полягатиме в аналізі результатів алгоритму навчання деякої початкової моделі, сформованої згідно припущень про закодований текст, та виявленні для кожної літери шифру відповідної літери оригінального тексту.

Отже, з формальної точки зору задача звучатиме так: пара $\{(X_t, Y_t)\}_{t=\overline{0,T}}$ – прихована марковська модель. Послідовність букв оригінального тексту $\{X_t\}_{t=\overline{0,T}}$ задана на множині літер $E = \{e_1, e_2, \ldots, e_M\}$, а послідовність спостережуваних символів шифру $\{Y_t\}_{t=\overline{0,T}}$ задана на алфавіті $F = \{f_1, f_2, \ldots, f_M\}$ деякої мови. Тоді задача кластеризації полягатиме у встановленні кожному символу з F відповідного символу з множини E.

Оскільки вектор спостережуваних символів має значну довжину, то, знову ж таки, для алгоритму Баума-Велша справедливі модифікації, наведені у підрозділах «Шкалювання алгоритмів прямого та зворотного ходу» та «Критерій зупинки алгоритму».

Для задачі дешифрування взято IV-X глави роману «Місто» автора Валер'яна Підмогильного. Тож в якості передпрограмного етапу підготуємо такі дві версії обраного тексту:

«оригінальний» текст	очищена версія початкового тексту: залишені лише літери українського алфавіту (відповідно, без знаків пробілу та построфа)
«зашифрований» текст	зашифрована версія «оригінального» тексту

Наприклад, розгялянемо уривок:

> *Проблема грошей набувала дедалі загрозливіших форм. Він був напередодні банкротства всього свого вбрання від кашкета до галош, що, послуживши йому півроку, починало виявляти ознаки страшного, хоч і природного занепаду, якого годі було вже приховати ретельним чищенням. Процес одягання, такий приємний йому колись, тепер у сущу муку обернувся, бо вранці наявніш, ніж будь-коли, показувалась руїна його білизни, крайнє зужиття черевиків та лихий блиск ліктів на піджаку, віщун майбутньої дірки.*

«Оригінальна» версія цього уривку матиме вид:

проблемагрошейнабуваладедалізагрозливішихформвінбувнапередоднібанкр отствавсьогосвоговбраннявідкашкетадогалошщопослужившийомупівpокупочин аловиявлятиознакистрашногохочіприродногозанепадуякогогодібуловжеприхова тиретельнимчищеннямпроцесодяганнятакийприємниййомуколисьтеперусущуму куобернувсябовранцінаявнішніжбудьколипоказувасьруїнайогобілизникрайнєзу життячеревиківталихийблискліктівнапіджакувіщунмайбутньоїдірки

В той час як «зашифрована» шифром Цезаря:

тусдоипгжусїимpгдцегогзизговкгжусколевїлшчсупеврдцергтиуизсзрвдгрну схфхегефясжсфесжседугррбевзнгїнихгзсжгосїьстсфоцйлеїлмспцтвеуснцтсґлрго селбеобхлскргнлфхугїрсжсшсґвтулусзрсжскгритгзцбнсжсжсзвдцосейитулшсегх луихиоярлпґльиррбптусщифсзбжгррбхгнлмтуліпрлммспцнсолфяхитиуцфцьцп цнцсдиурцефбдсеугрщвргбервїрвйдцзянсолтснгкцегогфяуцюргмсжсдволкрлнуг мрікцйлххбґиуиелнвехголшлмдолфновнхвергтвзйгнцевьцрпгмдцхрясюзвунл

Для подальшого формування початкової моделі λ^0 необхідні обидва варіанти тестів: як оригінальний, так і зашифрований. Шифровка тексту у прикладі здійснювалася шифром Цезаря, тобто зсувом кожної букви алфавіту на три позиції вперед:

а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ґ	ї	ь	є	ю	я	і
г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ґ	ї	ь	є	ю	я	і	а	б	в

Табл. 4: Шифрувальна таблиця

Тож на кінець підготовчого етапу отримуємо оригінальний та зашифрований тексти ($T = 112\,216$ символів), задані на алфавіті F довжиною $M = 33$ елементи:

$$F = \{\text{«а»}, \text{«б»}, \text{«в»}, \ldots, \text{«ь»}, \text{«є»}, \text{«ю»}, \text{«я»}, \text{«і»}\} \tag{11}$$

Останнім кроком перед запуском самого алгоритму навчання задамо початкову модель λ^0 таким чином:

A^0 : На основі аналізу оригінального тексту визначимо компоненти $\widetilde{A}_{ij}$ як кількість спостережень впорядкованої пари букв (i, j) у заданому тексті. Наступним кроком для виконання умови стохастичності матриці, а також задля невід'ємності її елементів, додамо до кожного $\widetilde{A}_{ij}$ число 5, а потім віднормуємо матрицю, поділивши кожен елемент на суму елементів відповідної стрічки:

$$\forall i \in E: \quad A^0_{ij} = \frac{\widetilde{A}_{ij}}{\sum\limits_{j \in F} \widetilde{A}_{ij}}$$

B^0 : Аби якнайкраще відповідати характеристикам наданого тексту (чутливість алгоритму до початкових параметрів прослідковувалася й раніше), побудуємо B^0 на основі результатів кластеризації літер на голосні-приголосні, використовуючи методологію попередніх розділів. Наприклад, групи голосних та приголосних, сформованих в результаті аналізу оригінального тексту, позначимо як C (consonants) та V (vowels) відповідно. А результати аналізу зашифрованого тексту – як $C^\#$ й $V^\#$. Тоді B^0 формуватиметься так:

$$\forall i \in E, \forall j \in F: \quad B^0_{ij} = \begin{cases} \sim \frac{1}{|C^\#|}, & i \in C, j \in C^\# \\ \sim 0, & i \in C, j \in V^\# \\ \sim \frac{1}{|V^\#|}, & i \in V, j \in V^\# \\ \sim 0, & i \in V, j \in C^\# \end{cases}$$

μ^0 : Покладемо вектор початкового розподілу як інваріантний розподіл ланцюга Маркова, тобто як розв'язок такого рівняння:

$$\mu^0 A = \mu^0,$$

де A – сформована у першому пункті матриця перехідних імовірностей.

Зважаючи на те, що початкова модель λ^0 формується, крім іншого, й на основі незашифрованого тексту, розв'язок задачі декодування є радше демонстративним, аніж дослідницьким.

Нарешті, запустимо процес виконання алгоритму навчання для заданої моделі $\lambda^0 = (\mu^0, A^0, B^0)$, враховуючи, що процес переоцінення слід виконувати лише для вектора μ та матриці B. Крім того, в якості ланцюжка спостережень слід взяти лише перші 1000 символів зашифрованого тексту.

Результати

Отримавши в результаті роботи $n = 200$ ітерацій переоцінену матрицю B^*, віднесемо кожен символ i з множини F символу j з множини E таким чином:

$$\forall i \in F : j = arg \max_{j \in E} B^*_{ij}$$

Утворена шифрувальна таблиця матиме вид:

а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ґ	ї	ь	є	ю	я	і
г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	к	ш	щ	ґ	ї	ш	к	ю	я	ї	а	б	в

Табл. 5: Шифрувальна таблиця

Порівнявши з Табл. 4 отримані відповідності, можна помітити, що чотири літери розшифровано неправильно (помилкові літери виділені синім кольором). Відтак відсоток правильно декодованих літер складає 88%.

Прослідкуємо за приростами логарифмів ймовірностей та за зростанням відсотку правильно вгаданих літер від ітерації до ітерації:

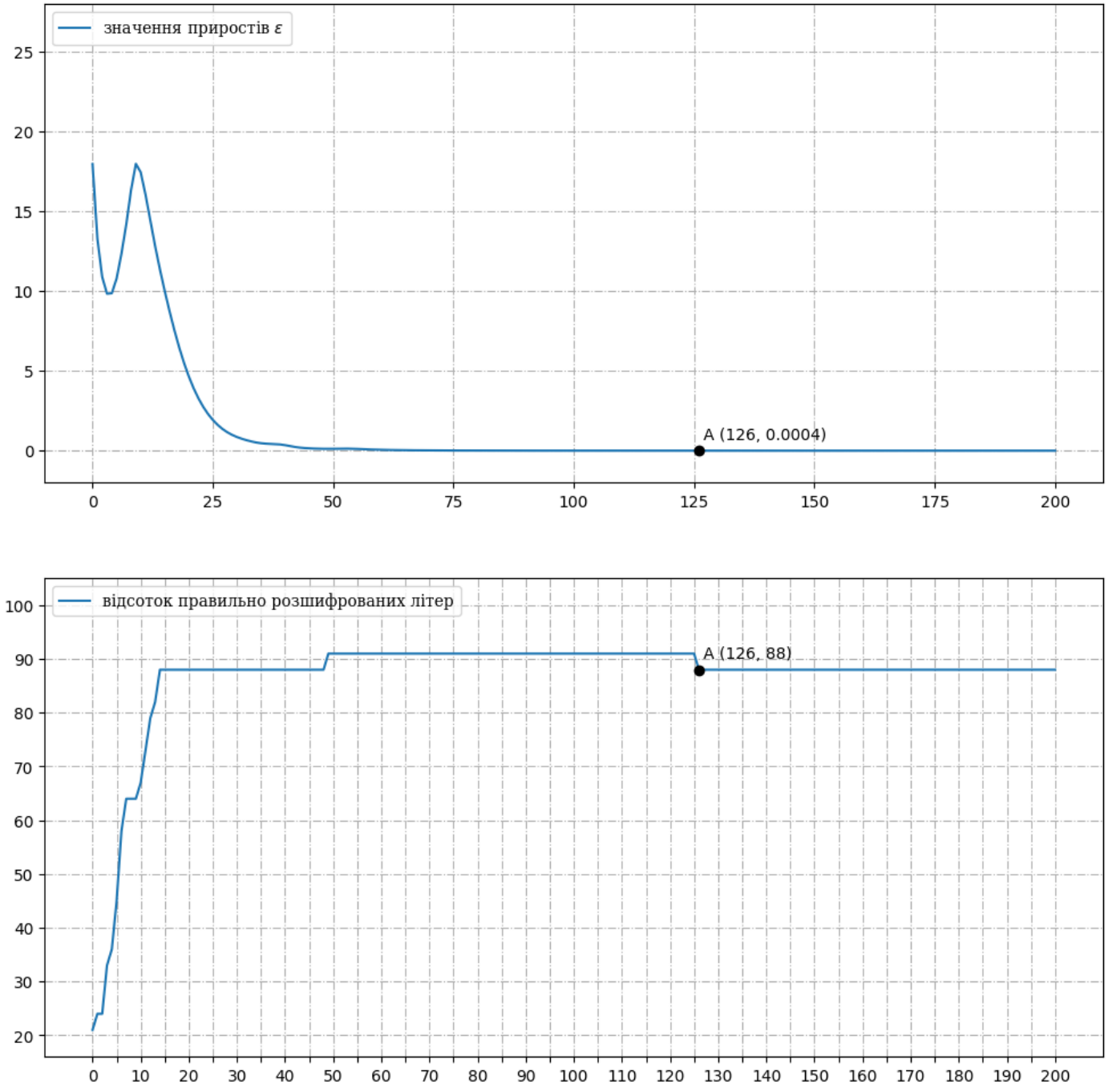

Рис. 7: Початкова модель $\lambda^0 = (A^0, B^0_{\text{голосні/приголосні}}, \mu^0_{\text{стаціонарний}})$

Назагал, модель демонструє незначні стрибки зростання та подальше стрімке спадання значень ε. Точкою A позначена координата ітерації $n = 126$, за якої відсоток правильно розшифрованих літер вперше сягає 88% й надалі не змінюється (на сотні ітерацій вперед).

Окремої уваги заслуговують й інші конфігурації початкової моделі λ^0. Наприклад, схожий характер збіжності прослідковується при аналогічному формуванні матриці B^0, проте при заданні початкового розподілу μ^0 рівномірним (Рис. 8).

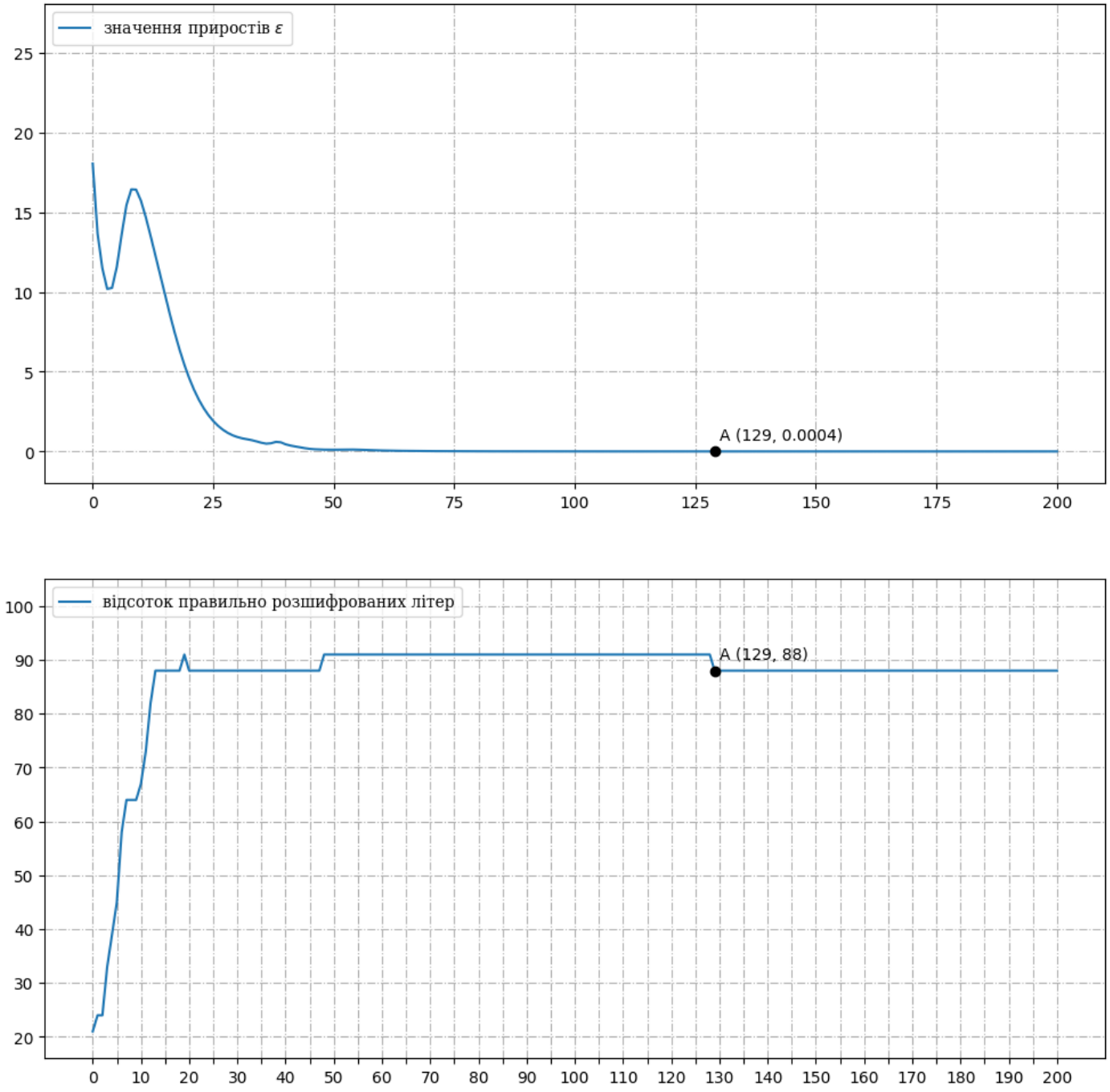

Рис. 8: Початкова модель $\lambda^0 = (A^0, B^0_{\text{голосні/приголосні}}, \mu^0_{\text{рівномірний}})$

Бачимо, що швидкість збіжності хоч і менша, проте не набагато: вже за $n = 129$ ітерацій відсоток досягає аналогічних 88%.

Натомість, при спробі задати початкові параметри як матриці B^0, так і вектора μ^0 рівномірним чином (Рис. 9), збіжність триває значно довше, включно до ітерації $n = 167$. Водночас, прирости ε зростають протягом перших ітерацій більш інтенсивно, що можна пояснити відсутністю в матриці B певної «підказки», як це було раніше.

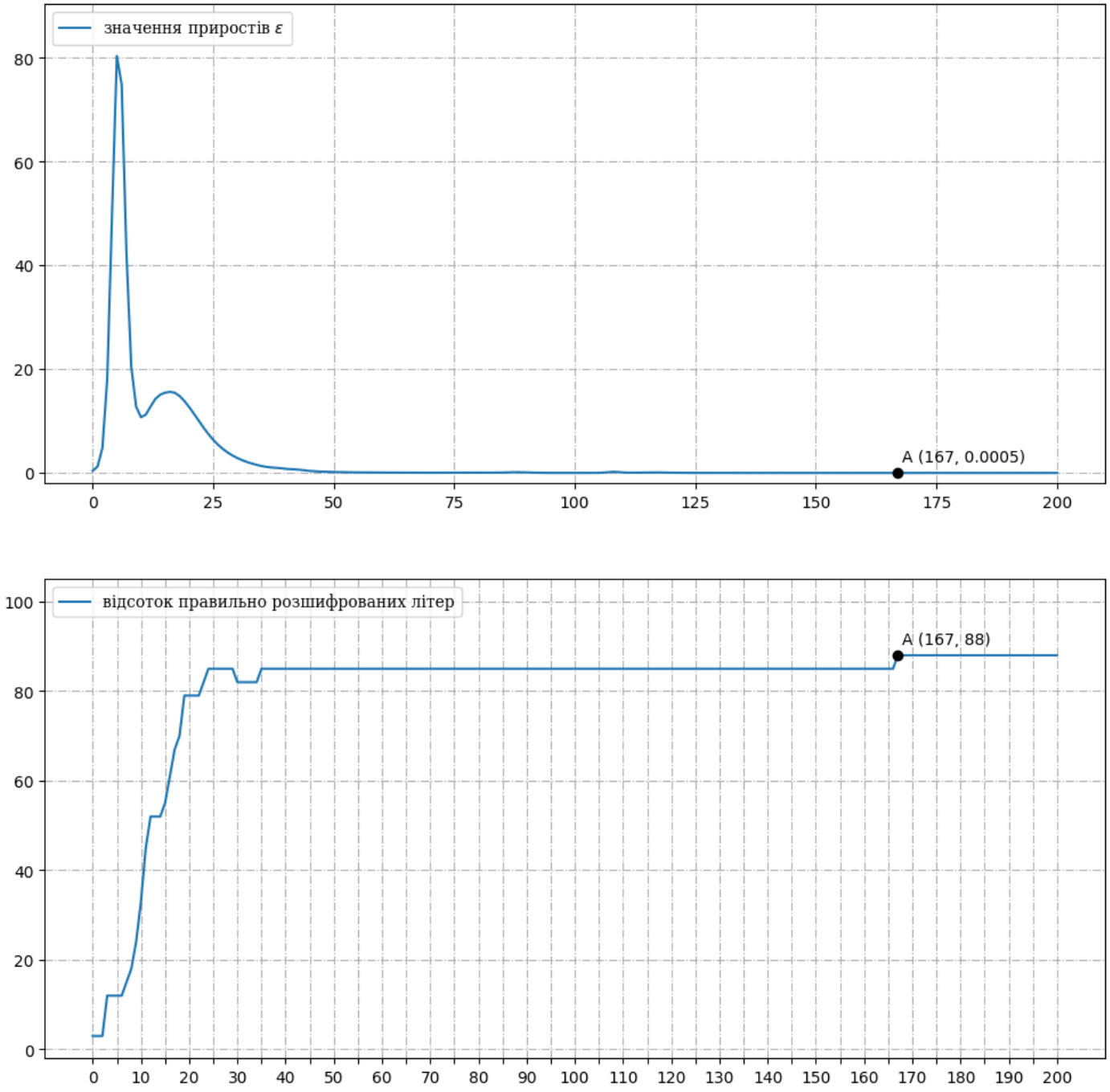

Рис. 9: Початкова модель $\lambda^0 = (A^0, B^0_{\text{рівномірна}}, \mu^0_{\text{рівномірний}})$

Отже, хоч збіжність триває і довше, рівномірні початкові параметри (за винятком матриці A^0, яка в усіх випадках має формуватися як частотний аналіз оригінального тексту) також демонструють хороші результати. Чого зовсім не скажеш про останню з можливих комбінацій, яка буде розглянута нижче.

Тож четвертим варіантом набору початкових параметрів є комбінація рівномірної матриці B^0 та стаціонарного розподілу μ^0 (Рис. 10). Прикро, проте у такому випадку модель зовсім не покращується, тобто відсоток правильно вгаданих літер сягає виключно нульового значення.

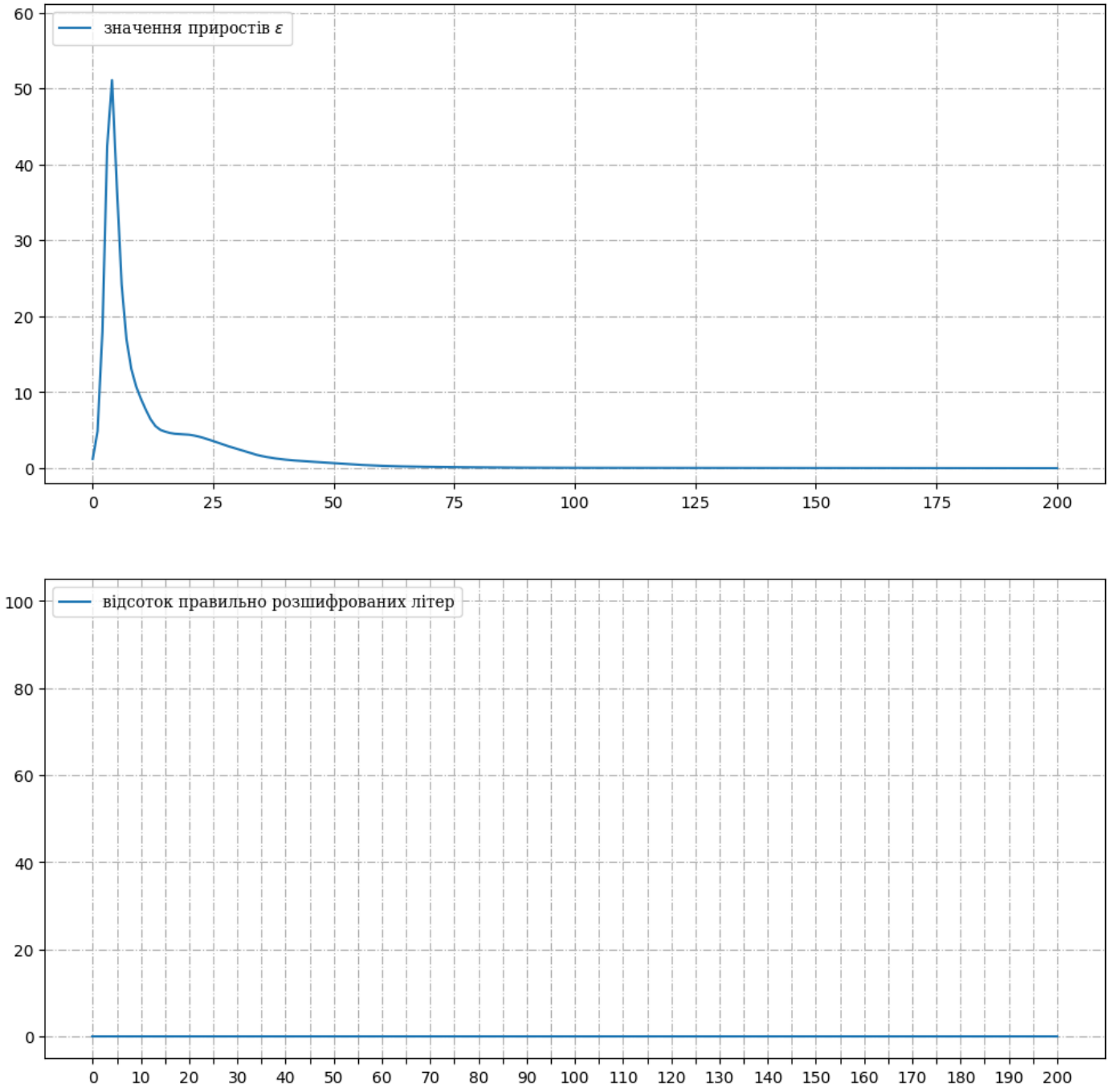

Рис. 10: Початкова модель $\lambda^0 = (A^0, B^0_{\text{рівномірна}}, \mu^0_{\text{стаціонарний}})$

Задача III (декодування)

Розглянемо розв'язок останньої із задач – задачі декодування. За заданою моделлю $\lambda = (\mu, A, B)$ і послідовністю спостережень $y = (y_0, \ldots, y_T)_{y_t \in F}$ слід віднайти послідовність станів $x = (x_0, \ldots, x_T)_{x_t \in E}$ прихованого ланцюга, яка максимізує умовну ймовірність $P_\lambda(X = x \,|\, Y = y)$.

Еквівалентним буде пошук послідовності, яка максимізує ймовірність сумісної появи ланцюжків: $P_\lambda(X = x, Y = y)$. Алгоритм, який дозволяє ефективно розв'язати задачу декодування, називається алгоритмом Вітербі.

Алгоритм Вітербі

Введемо величини $\delta_t(x_t)$ – ймовірності спостереження ланцюжка довжини t, використовуючи найкращий шлях, що закінчується станом x_t в момент часу t :

$$\delta_t(x_t) = \max_{x_0,\ldots,x_{t-1}} P_\lambda(X_0 = x_0, \ldots, X_t = x_t, Y_0 = y_0, \ldots, Y_t = y_t)$$

При заданій моделі $\lambda = (\mu, A, B)$, покладаючи $\forall x_0 \in E : \delta_0(x_0) = \mu_{x_0} B_{x_0 y_0}$, усі вказані коефіцієнти можна визначити ітераційно:

$$\delta_{t+1}(x_{t+1}) = B_{x_{t+1} y_{t+1}} \cdot \max_{x_t}\{\delta_t(x_t) A_{x_t x_{t+1}}\}$$

При цьому, щоб знайти оптимальний ланцюжок прихованих станів, необхідно відстежувати аргумент, при якому досягається максимум $\delta_t(x_t)$ для кожного t та x_t. Таким чином, алгоритм Вітербі знаходження найбільш ймовірного ланцюжка прихованих станів є таким:

1. Ініціалізація:

$$\forall x_0 \in E : \qquad \delta_0(x_0) = \mu_{x_0} B_{x_0 y_0},\ \psi_0(x_0) = 0$$

2. Рекурентно обчислити коефіцієнти $\delta_t(x_t)$ та відповідні аргументи $\psi_t(x_t)$:

$$\forall t = \overline{0, T},\ \forall x_{t+1} \in E : \qquad \delta_{t+1}(x_{t+1}) = B_{x_{t+1} y_{t+1}} \cdot \max_{x_t \in E}\{\delta_t(x_t) A_{x_t x_{t+1}}\}$$
$$\forall t = \overline{0, T},\ \forall x_{t+1} \in E : \qquad \psi_{t+1}(x_{t+1}) = arg\max_{x_t \in E}\{\delta_t(x_t) A_{x_t x_{t+1}}\}$$

3. Покласти зворотну точку відліку:

$$\delta^* = \max_{x_T \in E}\{\delta_T(x_T)\}$$
$$\psi^* = arg\max_{x_T \in E}\{\delta_T(x_T)\}$$

4. Визначити оптимальний ланцюжок станів (у зворотному порядку), починаючи з останнього $x_T^* = \psi^*$:

$$\forall t = \overline{T-1, 0} : \qquad x_t^* = \psi_{t+1}(x_{t+1}^*)$$

Приклад: кмітливі учні

Маючи в арсеналі інтсрумент для розв'язку задачі декодування, прослідкуємо за ще одним дослідженням учнів:

На жаль, результати задачі оцінювання показали учням, що, скоріше за все, наступного тижня у вечір проти четверга вони сидітимуть не на трибунах стадіону, а за робочими столами у своїх кімнатах, виконуючи домашнє завдання з математики.

Зневірившись, у школярів виникає ідея безжальної авантюри: використати прийом психологічного тиску та просто-напросто спробувати вмовити (фактично, благати) викладача задати легке домашнє завдання у бажані для учнів дні. Проте, виконавши таку операцію під гарячу руку, себто під час поганого настрою викладача, наслідки можуть бути доволі невтішними.

Відтак, отримавши в результаті задачі оцінювання на основі «математичного портрету» викладача найбільш імовірний ланцюжок заданих домашніх завдань наступного тижня, учні можуть спробувати спрогнозувати відповідну послідовність прихованих станів, тобто настроїв викладача. Тоді, завчасно звернувшись із пропозицією до учителя математики у найбільш сприйтливий для цього день, успіх буде гарантованим.

Використовуючи алгоритм Вітербі, віднайдемо ланцюжок $x = (x_1, x_2, x_3, x_4, x_5)$ прихованих натроїв викладача за отриманим в результаті задачі оцінювання найбільш імовірним ланцюжком спотережень заданого домашнього завдання протягом наступного тижня (висновки на стр. 7):

$$y = (\text{🖧}, \text{🖧}, \text{🖧}, \text{🖧}, \text{🖧})$$

Наведемо дані стосовно учителя математики, тобто модель λ :

	☺	😐	☹
☺	0.2	0.3	0.5
😐	0.2	0.2	0.6
☹	0	0.2	0.8

матриця A

	—	🔀	🖧
☺	0.7	0.2	0.1
😐	0.3	0.4	0.3
☹	0	0.1	0.9

матриця B

☺	0.05
😐	0.2
☹	0.75

вектор μ

Наразі маємо усю необхідну інформацію, аби запустити алгоритм декодування (спостережені типи завдань та приховані стани настроїв перекодовано номерами від 1 до 3):

Лістинг 5: Алгоритм Вітербі

```
def viterbi(y,m,A,B):

    delta = [[0.0 for i in range(len(B))] for t in range(time)]
    psi = [[0 for i in range(len(B))] for t in range(time)]

```

```
        for t in range(time):
            for i in range(len(B)):
                if t == 0:
                    delta[t][i] = m[i]*B[i][y[t]-1]
                else:
                    dA = []
                    for j in range(len(B)):
                        dA.append(delta[t-1][j]*A[j][i]*B[i][y[t]-1])
                    delta[t][i] = max(dA)
                    psi[t][i] = np.argmax(dA) + 1

        x = []
        delta_hat = max(delta[time-1])
        x.append(np.argmax(delta[time-1])+1)

        for t in range(time-2, -1, -1):
            x.insert(0, psi[t+1][x[0]-1])

        return x
```

В результаті отримуємо таку послідовність:

$$x = (☹, ☹, ☹, ☹, ☹)$$

Зрештою, це зовсім не дивно, зважаючи на заданий ланцюжок домашніх завдань. Можливо, учням варто задіяти класного керівника для врегулювання проблеми. Тоді задача декодування зведеться до прогнозу настрою їхнього куратора.

Приклад: дешифрування тексту

Наостанок, розв'яжемо задачу декодування вже розглянутого в одному з попередніх розділів зашифрованого тексту за допомогою алгоритму Вітербі. Сформулюємо задачу так: за відомими спостереженнями (тобто за набором літер зашифрованого тексту) віднайдемо прихований ланцюжок станів (для нас це буде розшифрований текст), використовуючи при цьому модель, навчену в результаті попереднього запуску алгоритму Баума-Велша (за методологією, наведеною на стр. 25).

Однак, через надвелику кількість літер у тексті (інакше кажучи, спостережуваних станів), алгоритм Вітербі слід модифікувати до алгоритму log-Вітербі.

Алгоритм log-Вітербі

У випадках, коли кількість спостережень є дуже великою, ітеративний пошук добутків, з яких формуються коефіцієнти $\delta_t(x_t)$, так чи інакше призведе до оперування з надмалими значеннями. Проте, використовуючи зростаючі властивості функції логарифма, усі максимуми, які беруть участь у формуванні величин $\delta_t(x_t)$, можна замінити рівносильними співвідношеннями:

$$\max \log f(x) = \log \max f(x)$$

Крім того, в алгоритмі log-Вітербі важливим є додатковий крок попередньої обробки, суть якого полягає у заміщенні всіх нульових елементів матриць та векторів наданої моделі $\lambda = (\mu, A, B)$ на найменші ненульові елементи, які сприймаються програматично.

Наприклад, для мови `Python` пороговою точністю є значення 10^{-323}. Це можна легко перевірити, порівнявши результати виконання відповідних команд:

Команда	Результат
`print(pow(10,-323))`	1e-323
`print(pow(10,-324))`	0.0

Отже, враховуючи усі зазначені зауваження, алгоритм log-Вітербі формулюватиметься таким чином:

0. Попередня обробка:

$$\forall i \in E: \quad \mu_i = \begin{cases} 10^{-323}, & \mu_i = 0 \\ \mu_i, & \mu_i \neq 0 \end{cases}$$

$$\forall i \in E,\, j \in F: \quad A_{ij} = \begin{cases} 10^{-323}, & A_{ij} = 0 \\ A_{ij}, & A_{ij} \neq 0 \end{cases}$$

$$\forall i \in E,\, j \in F: \quad B_{ij} = \begin{cases} 10^{-323}, & B_{ij} = 0 \\ B_{ij}, & B_{ij} \neq 0 \end{cases}$$

1. Ініціалізація:

$$\forall x_0 \in E: \qquad \delta_0(x_0) = \ln \mu_{x_0} + \ln B_{x_0 y_0},\ \psi_0(x_0) = 0$$

2. Рекурентно обчислити коефіцієнти $\delta_t(x_t)$ та відповідні аргументи $\psi_t(x_t)$:

$$\forall t = \overline{0, T},\ \forall x_{t+1} \in E: \quad \delta_{t+1}(x_{t+1}) = \ln B_{x_{t+1} y_{t+1}} + \max_{x_t \in E} \{\delta_t(x_t) + \ln A_{x_t x_{t+1}}\}$$

$$\forall t = \overline{0, T},\ \forall x_{t+1} \in E: \quad \psi_{t+1}(x_{t+1}) = arg \max_{x_t \in E} \{\delta_t(x_t) A_{x_t x_{t+1}}\}$$

3. Покласти зворотну точку відліку:

$$\delta^* = \max_{x_T \in E} \{\delta_T(x_T)\},\ \psi^* = arg \max_{x_T \in E} \{\delta_T(x_T)\}$$

4. Визначити оптимальний ланцюжок станів (у зворотному порядку), починаючи з останнього $x_T^* = \psi^*$:

$$\forall t = \overline{T-1, 0}: \qquad x_t^* = \psi_{t+1}(x_{t+1}^*)$$

Результати

Отримавши на вхід алгоритму log-Вітербі матрицю A^0, сформовану в результаті частотного аналізу оригінального тексту, переоцінену в ході роботи алгоритму Баума-Велша матрицю B^* (88% правильно вгаданих літер) та переоцінений вектор початкового розподілу μ^*, маємо змогу порівняти отриманий розв'язок задачі декодування. Отже, оригінал тексту має вигляд:

Проблема грошей набувала дедалі загрозливіших форм. Він був напередодні банкротства всього свого вбрання від кашкета до галош, що, послуживши йому півроку, починало виявляти ознаки страшного, хоч і природного занепаду, якого годі було вже приховати ретельним чищенням. Процес одягання, такий приємний йому колись, тепер у сущу муку обернувся, бо вранці наявніш, ніж будь-коли, показувалась руїна його білизни, крайнє зужиття черевиків та лихий блиск ліктів на піджаку, віщун майбутньої дірки.

В той час як результати роботи алгоритму є ось такими (пробіли вставлені виключно для простоти читання):

проблеза грошей набувала дедалі загрозливіших щорз він був напередодні банкровства всього свого вбрання від каємета до галоє мо постуживши йому півроку починало виявляти ознаки страєщого гоч і природного занепаду якого годі було вже приховати ретельним чименням процес одягання такий приємний йому колись тепер у суму муку обернувся бо вранці наявніш ній будь коли показувалась ружна його білизни крайнє зужиття черевиків та лихий блиск ліктів на піджаку вімун зайбутньої дірки