Algorithm Implementation

EQ2341 Pattern Recognition and Machine Learning, Assignment 3

Jingxuan Mao jmao@kth.se

Yuqi Zheng yuqizh@kth.se

May 6, 2022

1 The Forward Algorithm

1.1 Implementation of the Forward Algorithm

@GaussD/prob

```
    def likelihood(self, obsrv):

        pX = np.zeros(len(obsrv))
        for j in range(len(obsrv)):
            pX[j] = stats.norm.pdf(obsrv[j], self.means[0],
                self.stdevs[0])

        return pX

    def prob(obsrv, distr):

        pX = [d.likelihood(obsrv) for d in distr]
        pX = np.array(pX)

        pX_scaled = np.zeros(pX.shape)
        factors = np.zeros(pX.shape[1])
        for i in range(pX_scaled.shape[0]):
            for j in range(pX_scaled.shape[1]):
                pX_scaled[i, j] = pX[i, j] / np.amax(pX[:, j])
                factors[j] = np.amax(pX[:, j])

        return pX, pX_scaled, factors
```

@MarkovChain/forward

```
    def forward(self, pX):

        n_states = pX.shape[0]
        n_obsrvs = pX.shape[1]
        alpha_temp = np.zeros((n_states, n_obsrvs))
        alpha_hat = np.zeros((n_states, n_obsrvs))
        c = np.zeros(n_obsrvs)

        # Initialization
        alpha_temp[:, 0] = self.q * pX[:, 0]
        c[0] = np.sum(alpha_temp[:, 0])
        alpha_hat[:, 0] = alpha_temp[:, 0] / c[0]

        # Forward Step
```

```
    for t in range(1, n_obsrvs):
        alpha_temp[:, t] = pX[:, t] * np.dot(alpha_hat[:, t-1],
            self.A[:, :n_states])
        c[t] = np.sum(alpha_temp[:, t])
        alpha_hat[:, t] = alpha_temp[:, t] / c[t]

    # Termination
    if self.is_finite:
        c = np.append(c, np.dot(alpha_hat[:, -1], self.A[:, -1]))

    return alpha_hat, c
```

1.2 Verification of the @MarkovChain/forward Function

$$q = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad A = \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.9 & 0.1 \end{pmatrix}$$

$$g_1 = \mathcal{N}(0, 1)$$

$$g_2 = \mathcal{N}(3, 2)$$

Unscaled:

$$pX = \begin{pmatrix} 0.3910 & 0.0136 & 0.1714 \\ 0.0555 & 0.1955 & 0.1390 \end{pmatrix}$$

$$\hat{\alpha} = \begin{pmatrix} 1 & 0.3847 & 0.4189 \\ 0 & 0.6153 & 0.5811 \end{pmatrix}$$

$$c = \begin{pmatrix} 0.3910 & 0.0318 & 0.1417 & 0.0581 \end{pmatrix}$$

Scaled:

$$pX_{scaled} = \begin{pmatrix} 1 & 0.0695 & 1 \\ 0.1418 & 1 & 0.8111 \end{pmatrix}$$

$$\hat{\alpha}_{scaled} = \begin{pmatrix} 1 & 0.3847 & 0.4189 \\ 0 & 0.6153 & 0.5811 \end{pmatrix}$$

$$c_{scaled} = \begin{pmatrix} 1 & 0.1625 & 0.8266 & 0.0581 \end{pmatrix}$$

2 Probability of a Feature Sequence

2.1 Implementation of Log-Probability Calculation

@HMM/logprob

```
def logprob(obsrv, distr, mc):

    pX, pX_scaled, factors = GaussD.prob(obsrv=obsrv, distr=distr)
    alpha_hat, c = mc.forward(pX)
    alpha_hat_scaled, c_scaled = mc.forward(pX_scaled)

    c_ori = np.append(c_scaled[0:len(factors)] * factors,
        c_scaled[-1])
    prob_log = sum(np.log(c))
    prob_log_scaled = sum(np.log(c_ori))

    return alpha_hat, alpha_hat_scaled, c, c_scaled, prob_log,
        prob_log_scaled
```

2.2 Validation of the @HMM/logprob Function

$$problog = -9.1877$$