$$P(x_{N+1}|X_N) = \sum_{z_{N+1}} P(x_{N+1}, z_{N+1}|X_N) = \sum_{z_{N+1}} P(x_{N+1}|z_{N+1})P(z_{N+1}|X_N)$$

$$P(x_{N+1}|X_N) = \sum_{z_{N+1}} P(x_{N+1}|z_{N+1}) \sum_{z_N} P(z_{N+1}, z_N|X_N) = \sum_{z_{N+1}} P(x_{N+1}|z_{N+1}) \sum_{z_N} P(z_{N+1}|z_N)P(z_N|X_N)$$

$$P(x_{N+1}|X_N) = \sum_{z_{N+1}} P(x_{N+1}|z_{N+1}) \sum_{z_N} P(z_{N+1}|z_N) \frac{P(z_N, X_N)}{P(X_N)}$$

$$P(x_{N+1}|X_N) = \sum_{z_{N+1}} P(x_{N+1}|z_{N+1}) \sum_{z_N} P(z_{N+1}|z_N) \frac{\alpha(z_N)}{P(X_N)}$$

由上可知，要预测观测值必须知道过去的隐藏状态值，从而确定转换概率$P(z_{N+1}|z_N)$和发射概率$P(x_{N+1}|z_{N+1})$，然后可以确定下一观测值的概率，如果不确定过去的隐藏状态值，可以先通过维特比算法确定最有可能的隐藏状态序列。

```python
# 已知当前序列预测未来（下一个）观测值的概率
def predict(self, X, x_next, Z_seq=np.array([]), istrain=True):
    if self.trained == False or istrain == False:  # 需要根据该序列重新训练
        self.train(X)

    X_length = len(X)
    if Z_seq.any():
        Z = np.zeros((X_length, self.n_state))
        for i in range(X_length):
            Z[i][int(Z_seq[i])] = 1
    else:
        Z = np.ones((X_length, self.n_state))
    # 向前向后传递因子
    alpha, _ = self.forward(X, Z)  # P(x,z)
    prob_x_next = self.emit_prob(np.array([x_next]))*np.dot(alpha[X_length - 1],self.transmat_prob)
    return prob_x_next
```