ml-hw01-q1

June 17, 2024

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
```

[ ]:
```python
u1 = np.array([-1, 1]).T
u2 = np.array([-2.5, 2.5]).T
u3 = np.array([-4.5, 4.5]).T
sigma1 = np.eye(2)
sigma2 = np.eye(2)
sigma3 = np.eye(2)
```

[ ]:
```python
def create_data(num_train= 700, num_test= 300):
    synthetic_samples = []
    synthetic_test_samples = []
    actual_test_labels = []
    for i in range(num_test + num_train):
        # choose lable at randomeness of 1/3
        label = np.random.choice([1, 2, 3], p=[1/3, 1/3, 1/3])
        if label == 1:
            sample = np.random.multivariate_normal(u1, sigma1)
        elif label == 2:
            sample = np.random.multivariate_normal(u2, sigma2)
        else:
            sample = np.random.multivariate_normal(u3, sigma3)
        # sample according to the distribution u_i, sigma_i given
        # add to training/test set accordingly
        if i < num_train:
            synthetic_samples.append((sample, label))
        else:
            actual_test_labels.append(label)
            synthetic_test_samples.append(sample)
    # converting sample arrays to np arrays
    sample_data = np.array([i for i, _ in synthetic_samples])
    sample_labels = np.array([l for _, l in synthetic_samples])
    synthetic_test_samples = np.array(synthetic_test_samples)
    return sample_data, sample_labels, synthetic_test_samples,␣
↪actual_test_labels
```

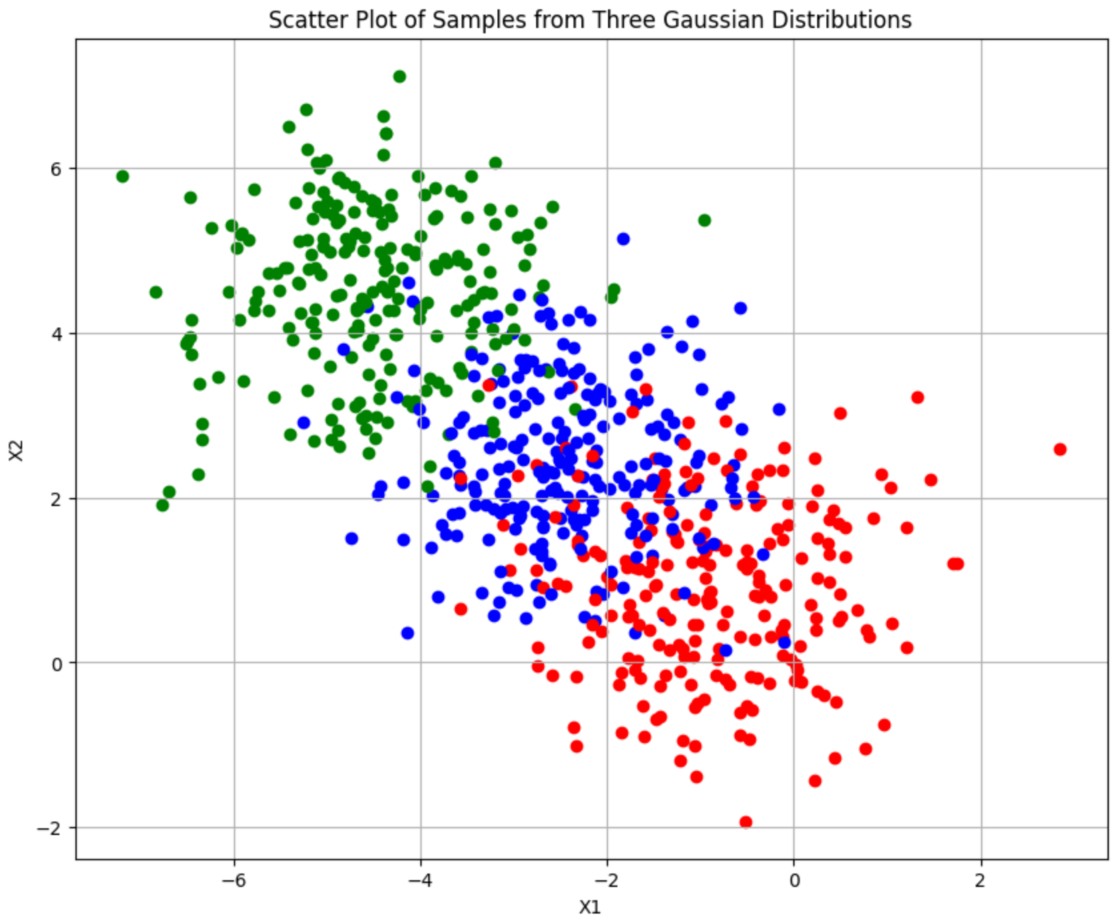
[ ]:
```python
def knn_model_run(i=1, sample_data=None, sample_labels=None, synthetic_test_samples=None, actual_test_labels=None):
    knn_classifier = KNeighborsClassifier(n_neighbors=i, p=2)  # l2 norm
    # training model
    knn_classifier.fit(sample_data, sample_labels)

    # Predict labels for the test data
    predicted_labels = knn_classifier.predict(synthetic_test_samples)

    # the score is the #correct classified labels, so error rate is 1 - #correct classified lables
    error_rate_train = 1 - knn_classifier.score(sample_data, sample_labels)

    # we compare the labels of the predicted labels to their actual labels, and take the average since we're summing 1's if they don't match
    # which gives us the error rate in a range of [0,1]
    error_rate_test = (sum(i != y for i, y in zip(predicted_labels, actual_test_labels))) / len(predicted_labels)

    return error_rate_train, error_rate_test
```

[ ]:
```python
def Q1_p2_3(sample_data, sample_labels, synthetic_test_samples, actual_test_labels):
    colors = ['r', 'g', 'b']
    labels = ['Gaussian 1', 'Gaussian 2', 'Gaussian 3']

    plt.figure(figsize=(10, 8))
    label_color_map = ['r', 'b', 'g']
    # Plot the data points with their corresponding labels
    for data, label in zip(sample_data, sample_labels):
        plt.scatter(data[0], data[1], color=label_color_map[label - 1])

    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title('Scatter Plot of Samples from Three Gaussian Distributions')
    plt.grid(True)
    plt.show()

    # test samples plot
    plt.figure(figsize=(10, 8))
    for data, label in zip(synthetic_test_samples, actual_test_labels):
        plt.scatter(data[0], data[1], color=label_color_map[label - 1])
    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title('Scatter Plot of Test Samples from Three Gaussian Distributions')
    plt.grid(True)
    plt.show()
```

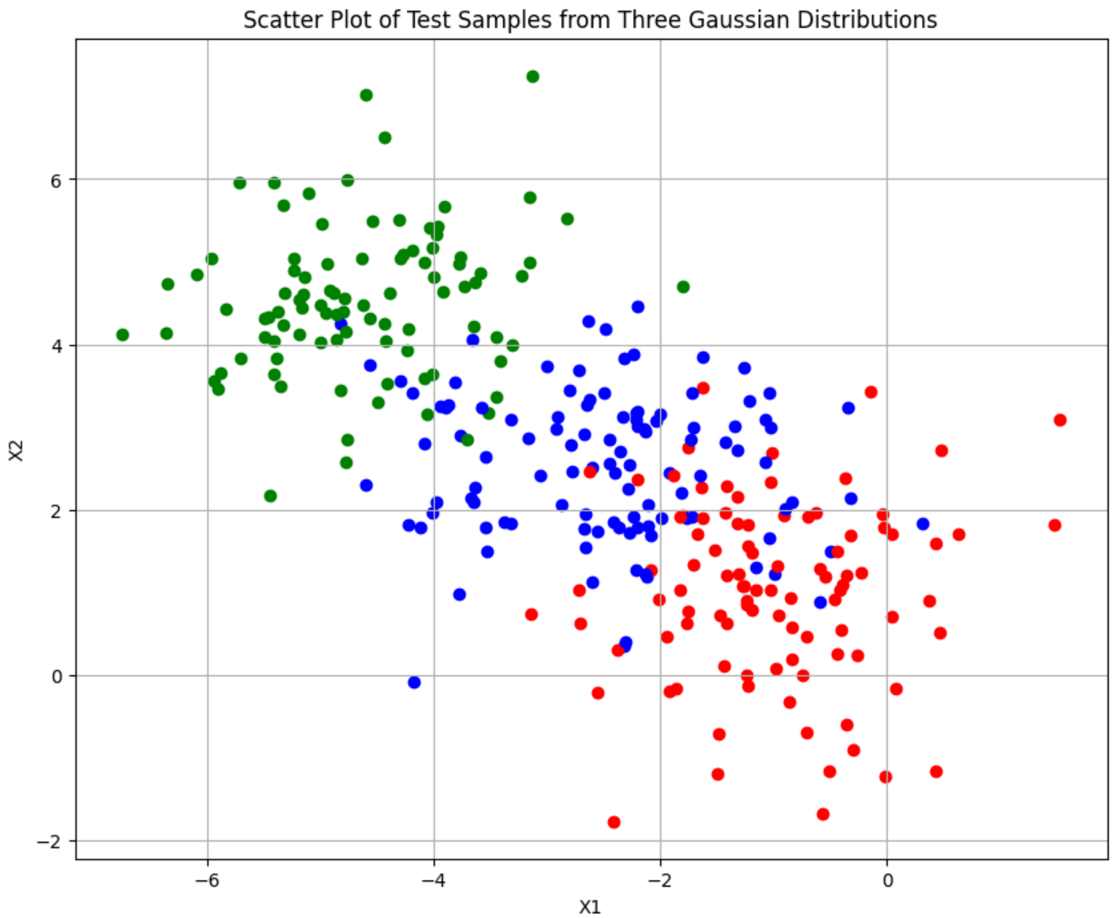
[ ]:
```python
def Q1_p4_5(sample_data, sample_labels, synthetic_test_samples,␣
↪actual_test_labels):
    for i in range(1, 21):
        # Q1.4,5
        # call function to get error rates for knn model with k=i
        error_rate_train, error_rate_test = knn_model_run(i, sample_data,␣
↪sample_labels, synthetic_test_samples, actual_test_labels)
        print(f"\nerror rate on knn model with {i} neighbours on train set:␣
↪{error_rate_train}")
        print(f"error rate on knn model with {i} neighbours on test set:␣
↪{error_rate_test}")
```

[ ]:
```python
def Q1_p6():
    m_test = 100
    train_errors = []
    test_errors = []
    m_train_vals = [i for i in range(10, 45, 5)]
    for m_train_i in m_train_vals:
        # create data with different training sample sizes and train knn model␣
↪with k = 10
        sample_data, sample_labels, synthetic_test_samples, actual_test_labels␣
↪= create_data(m_train_i, m_test)
        error_rate_train, error_rate_test = knn_model_run(10, sample_data,␣
↪sample_labels, synthetic_test_samples, actual_test_labels)
        # procedure to save error rates for each different model
        train_errors.append(error_rate_train)
        test_errors.append(error_rate_test)

    # Plotting the results
    plt.figure(figsize=(10, 6))
    plt.plot(m_train_vals, train_errors, label='Train Error')
    plt.plot(m_train_vals, test_errors, label='Test Error')
    plt.xlabel('Training Set Size (m_train)')
    plt.ylabel('Error Rate')
    plt.title('Train and Test Errors as a function of Training Set Size')
    plt.grid(True)
    plt.show()
```

[ ]:
```python
# by default with give us 700 synthetic samples as required in Q1.1 and 300␣
↪test samples for Q1.3
sample_data, sample_labels, synthetic_test_samples, actual_test_labels =␣
↪create_data()

# plots samples on to graphs- one for the synthetic sample and the other for␣
↪the test sampels for Q1.2, Q1.3
Q1_p2_3(sample_data, sample_labels, synthetic_test_samples, actual_test_labels)
```



Scatter Plot of Samples from Three Gaussian Distributions
X2
X1
6
4
2
0
−2
−6
−4
−2
0
2



Q1. sections 4, 5 Function trains KNN model on k = 1,2,...,20 training error rate is 0 and test error rate is approx 0.233 (since we take from different ditributions and don't set a seed we will get various training errors) The reason we get such different results is since we are training on k=1 => we overfit on the training set since we train where each neighbor classifies as it is and overfit tends to give a worse training error on the test set. Therefore, the overfitting will make it so we missclassify on the test set and therefore we get an error rate $> 0 =$ training error rate.

The test error rate doesn't decrease as we raise the value of k. Counter Example: K=4 where test error rate $= 0.18$ but for K=5 we get that the test error rate is 0.1966, where the test error rate increased.

[ ]:
```
Q1_p4_5(sample_data, sample_labels, synthetic_test_samples, actual_test_labels)
```

```
error rate on knn model with 1 neighbours on train set: 0.0
error rate on knn model with 1 neighbours on test set: 0.23333333333333334

error rate on knn model with 2 neighbours on train set: 0.08714285714285719
error rate on knn model with 2 neighbours on test set: 0.22333333333333333
```

```
error rate on knn model with 3 neighbours on train set: 0.09285714285714286
error rate on knn model with 3 neighbours on test set: 0.20666666666666667

error rate on knn model with 4 neighbours on train set: 0.11285714285714288
error rate on knn model with 4 neighbours on test set: 0.18

error rate on knn model with 5 neighbours on train set: 0.11857142857142855
error rate on knn model with 5 neighbours on test set: 0.19666666666666666

error rate on knn model with 6 neighbours on train set: 0.13428571428571423
error rate on knn model with 6 neighbours on test set: 0.18666666666666668

error rate on knn model with 7 neighbours on train set: 0.13142857142857145
error rate on knn model with 7 neighbours on test set: 0.18

error rate on knn model with 8 neighbours on train set: 0.13857142857142857
error rate on knn model with 8 neighbours on test set: 0.16333333333333333

error rate on knn model with 9 neighbours on train set: 0.14428571428571424
error rate on knn model with 9 neighbours on test set: 0.16333333333333333

error rate on knn model with 10 neighbours on train set: 0.13428571428571423
error rate on knn model with 10 neighbours on test set: 0.16333333333333333

error rate on knn model with 11 neighbours on train set: 0.1271428571428571
error rate on knn model with 11 neighbours on test set: 0.15

error rate on knn model with 12 neighbours on train set: 0.12571428571428567
error rate on knn model with 12 neighbours on test set: 0.16333333333333333

error rate on knn model with 13 neighbours on train set: 0.13571428571428568
error rate on knn model with 13 neighbours on test set: 0.15666666666666668

error rate on knn model with 14 neighbours on train set: 0.13571428571428568
error rate on knn model with 14 neighbours on test set: 0.16333333333333333

error rate on knn model with 15 neighbours on train set: 0.13857142857142857
error rate on knn model with 15 neighbours on test set: 0.15333333333333332

error rate on knn model with 16 neighbours on train set: 0.1328571428571429
error rate on knn model with 16 neighbours on test set: 0.16

error rate on knn model with 17 neighbours on train set: 0.13857142857142857
error rate on knn model with 17 neighbours on test set: 0.14333333333333334

error rate on knn model with 18 neighbours on train set: 0.13428571428571423
error rate on knn model with 18 neighbours on test set: 0.15
```

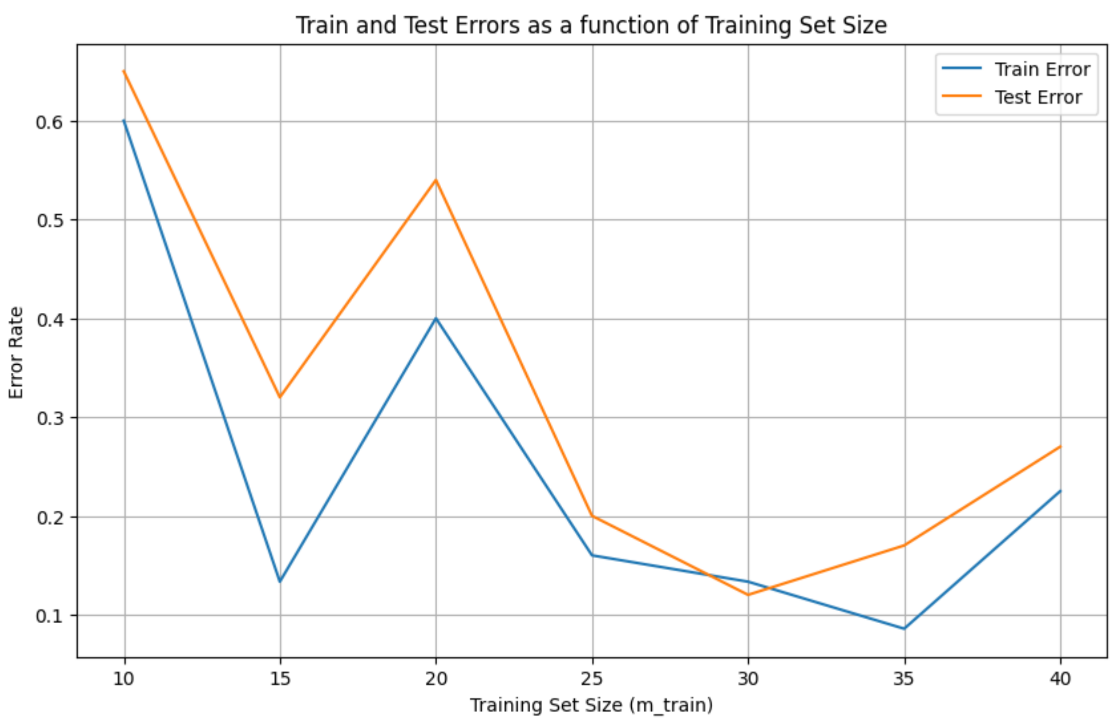
```
error rate on knn model with 19 neighbours on train set: 0.1328571428571429
error rate on knn model with 19 neighbours on test set: 0.14333333333333334

error rate on knn model with 20 neighbours on train set: 0.1328571428571429
error rate on knn model with 20 neighbours on test set: 0.15333333333333332
```

Q1.6 I expect the two graphs to be very similar since it's a model we expect the training error and test error to be similar and thus result in similar graphs. It does match our expectations in this manner. We can see that they don't match perfectly but they do have similar shapes. Another expectation is that within a certain range the error rates should remain more or less consistent. Here it doesn't happen, Looking between k=15-35, there is a peak where k=30 where the train and test error rates are approx 0.4, 0.55 which isn't consistant with the rest of the error rates in that range.

Q1.7 Yes, the line plots change between trials. Overall, similar behavior as I articulated in step 6 so it meets and doesn't in a similar manner to step 6 as written above.

Q1.8 each point will be assigned it's importance in relation to the surrounding points which we"ll do using the distances. We would use the weighted average of the labels of the k closest points. Formula: h_s(x) = sum (i=1,...,k) (((w(x, x'*(pi_i(x)))) / (sum (j=1,...,k) (w(x, x'*(pi_j(x))))) * y_pi * x)

[ ]:
```
Q1_p6()
```



1.

Let $w_1, w_2 \in \mathbb{R}^d$

Choose $w = w_1 - w_2$

$$p = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} = \frac{e^{(w_1 - w_2)^T x_i}}{1 + e^{(w_1 - w_2)^T x_i}} = \frac{e^{w_1^T x_i - w_2^T x_i}}{1 + e^{w_1^T x_i - w_2^T x_i}} \cdot \frac{e^{w_2^T x_i}}{e^{w_2^T x_i}} = \frac{e^{w_1^T x_i}}{e^{w_2^T x_i} + e^{w_1^T x_i}} = p_1$$

$$1 - p = 1 - p_1 = 1 - \frac{e^{w_1^T x_i}}{e^{w_2^T x_i} + e^{w_1^T x_i}} = \frac{e^{w_2^T x_i}}{e^{w_2^T x_i} + e^{w_1^T x_i}} = p_2$$

■

Let $w \in \mathbb{R}^d$

Choose $w_1 = w, w_2 = 0$

$$p_1 = \frac{e^{w_1^T x_i}}{e^{w_2^T x_i} + e^{w_1^T x_i}} = \frac{e^{w_1^T x_i}}{e^0 + e^{w_1^T x_i}} = \frac{e^{w_1^T x_i}}{1 + e^{w_1^T x_i}} = p$$

$$p_2 = \frac{e^{w_2^T x_i}}{e^{w_2^T x_i} + e^{w_1^T x_i}} = \frac{1}{1 + e^{w_1^T x_i}} = 1 - \frac{e^{w_1^T x_i}}{1 + e^{w_1^T x_i}} = 1 - p$$

■

2.

$$\sum_{i=1}^{m} \log\big(P_w(Y_i = y_i | x_i)\big) = \sum_{i=1}^{m} \log\left(\frac{e^{w_{y_i}^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}\right) = \sum_{i=1}^{m} \left(w_{y_i}^T x_i - \log\left(\sum_{j=1}^{K} e^{w_j^T x_i}\right)\right)$$

We'll take partial derivatives with respect to w_k of the expression in order to find the maximum:

$$\forall k\colon \frac{\partial}{\partial w_k} \sum_{i=1}^{m} \left(w_{y_i}^T x_i - \log\left(\sum_{j=1}^{K} e^{w_j^T x_i}\right)\right) = \sum_{i=1}^{m} \frac{\partial}{\partial w_k} w_{y_i}^T x_i - \sum_{i=1}^{m} \frac{\partial}{\partial w_k} \log\left(\sum_{j=1}^{K} e^{w_j^T x_i}\right)$$

$$= \sum_{i=1}^{m} I[y_i = k] x_i - \sum_{i=1}^{m} \frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}} x_i = 0$$

$$\Rightarrow \sum_{i=1}^{m} I[y_i = k] \cdot x_i = \sum_{i=1}^{m} \frac{e^{w_k^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}} \cdot x_i$$

3. a) Each weight represents the score of the matching class in logistic regression and we need to split the hyperplane to match the three classes.

b) $X \in \mathbb{R}^2$ since each piece of data is represented by two real numbers.

$W_i \in \mathbb{R}^3$ since we add an additional bias parameter to W such that $(w_1, w_2)^T \cdot (x_1, x_2) + \underbrace{w_0}_{bias}$ to make a correction/fix.

c)

$$P_{11} = P_w(r_i = 1 \mid x_1) = P_w(r_i = 0 \mid (1,8)) = \frac{1}{1+e^{w_1^T x_1}} = \frac{1}{e^{21.5}+1} = 0.99999995$$

$w_1^T x_i = 8 - 2.5 \cdot 1 + 2 \cdot 8 = 21.5$

$$P_{12} = P_w(r_i = 1 \mid x_2) = P_w(r_i = 0 \mid (6,-2))$$

$$= \frac{e^{w_1^T x_2}}{1+e^{w_1^T x_2}} = \frac{e^{-11}}{1+e^{-11}} = 1.67 \cdot 10^{-5} \approx 0$$

$w_1^T x_2 = 8 + 6 \cdot (-2.5) + 2 \cdot (-2) = 4 - 15 = -11$

$$P_{13} = P_w(r_i = 1 \mid x_3) = P_w(r_i = 1 \mid (12,4)) = \frac{e^{w_i^T x_3}}{1+e^{w_i^T x_3}} = 9.35 \cdot 10^{-14} \approx 0$$

$w_i^T \cdot x_3 = 8 - 2.5 \cdot 12 - 2 \cdot 4 = -30$

Since we will use maximum likelihood we will choose the label 1

$$P_{21} = P_w(t_i = 2 \mid \overset{(1,8)}{x_1}) = \frac{e^{\langle w_2, (1,8)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (1,8)\rangle}} = \frac{e^{2+0.5-12}}{e^{8-2.5+16} + e^{2+0.5-14} + e^{-10+2-4}} \approx 0$$

$$P_{22} = P_w(t_i = 2 \mid \overset{(6,2)}{x_2}) = \frac{e^{\langle w_2, (6,2)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (6,2)\rangle}} = \frac{e^{2+3+3}}{e^{8-15-4} + e^{2+3+3} + e^{-10+12+1}} \approx 0.993$$

$$P_{23} = P_w(t_i = 2 \mid \overset{(12,4)}{x_3}) = \frac{e^{\langle w_2, (12,4)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (12,4)\rangle}} = \frac{e^{2+6-6}}{e^{8-30+8} + e^{2+6-6} + e^{-10+24-2}} = 4.53 \cdot 10^{-5} \approx 0$$

לכן נבחר במקרה זה 2.

$$P_{21} = P_w(t_i = 3 \mid \overset{1,8}{x_1}) = \frac{e^{\langle w_3, (1,8)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (1,8)\rangle}} = \frac{e^{-10+2-4}}{e^{8-2.5+16} + e^{2+0.5-14} + e^{-10+2-4}} \approx 0$$

$$P_{22} = P_w(t_i = 3 \mid \overset{(6,2)}{x_2}) = \frac{e^{\langle w_3, (6,2)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (6,2)\rangle}} = \frac{e^{-10+12+1}}{e^{8-15-4} + e^{2+3+3} + e^{-10+12+1}} \approx 0$$

$$P_{23} = P_w(t_i = 3 \mid \overset{(12,4)}{x_3}) = \frac{e^{\langle w_3, (12,4)\rangle}}{\sum_{k=1}^{3} e^{\langle w_j, (12,4)\rangle}} = \frac{e^{-10+24-2}}{e^{8-30+8} + e^{2+6-6} + e^{-10+24-2}} \approx 0.99$$

לכן נבחר פה את 3