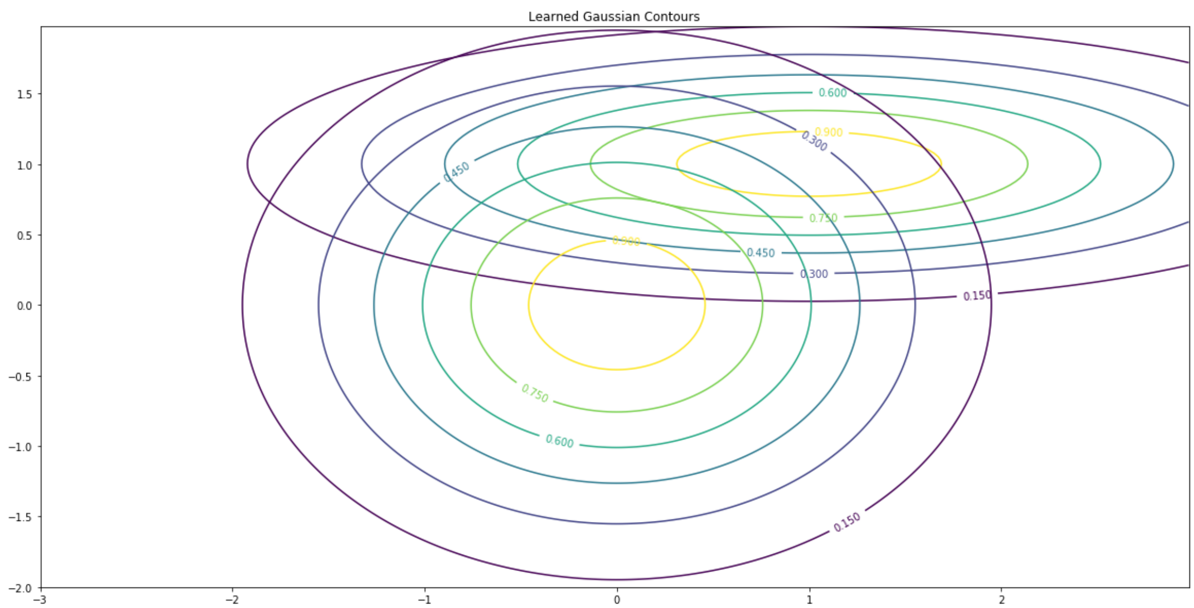
In [108]:

```
# -*- coding: utf-8 -*-
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  7 18:10:29 2018


In Class Exercise:  xx/100  (+ 10 extra credit if submitted before end of day
 of class)
******************************************************
Assume: Single Gaussian
Goal: Implement functions, given data X.
mu0, sigma0 = gaussianML(X)
mu1, sigmaGuess = gaussianMAP(X, priorMu, sigmaGuess)
******************************************************

The functions should compute the ML and MAP estimates respectively. Plot
the results of the learned parameters (using a contour like display) for the
purposes of comparison. Feel free  to use the code snippets provided below
or feel free to edit as you see fit. Assume covariance matrices are diagonal.
Also assume \mu has Gaussian Prior. We will not solve for the MAP covariance,
instead, We will discuss more when we dive into Bayesian
Methods.

Construct synthetic 2-d data X or find a simple data set to use to test your c
ode.

********************************************************



Take-Home Assignment:  xx/100
******************************************************
To Do: Type-up responses and supporting figures and submit as a PDF.
******************************************************

#1 Assume a Single Gaussian Classifier with training data X. Assume a fixed
mean parameter \mu. What is the Maximum Likelihood Objective one would use to
solve for the variance parameter \sigma?

#2 Find (solve for) the ML estimate for the variance. Show all steps
 starting with the objective in #1. Show / explain all steps :)

#3 Show the MAP objective and assuming a fixed mean \mu. Show all steps
starting with the MAP objective. Show / explain all steps :)

#4 Briefly compare and contrast the ML and MAP estimates for a single Gaussian
 Classifier.

#5 Prove that the ML and MAP mean estimates are similar when the prior probabi
lity is uniform.
Prove this semi-formally -- use math to help support your claims. ALSO Create
 plots from
your in-class exercise to support your claim.

********************************************************

For supplemental help ...
```

```
See Resources:
    https://matplotlib.org/
    [TK], [Bp], [DHS]


@author: jerem
"""







import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp

# Training Data
# Choose a training data set.
# X =


# Create Method to Learn ML estimate of 2d Gaussia
#def gaussianML(X):
#
# ... Fill this in
#    return  mu, sigma

#def  gaussianMAP(X, priorMu, sigmaGuess):
#
# ... Fill this in
#    return  mu, sigma



def gaussian_2d(x, y, x0, y0, xsig, ysig):
    return np.exp(-0.5*(((x-x0) / xsig)**2 + ((y-y0) / ysig)**2))


def plt_2dGaussians(mu0, mu1, sig0, sig1):
    delta = 0.025
    x = np.arange(-3.0, 3.0, delta)
    y = np.arange(-2.0, 2.0, delta)
    X, Y = np.meshgrid(x, y)
    Z1 = gaussian_2d(X, Y, mu0[0], mu0[1], sig0[0],sig0[1])
    Z2 = gaussian_2d(X, Y, mu1[0], mu1[1],  sig1[0],sig1[1])

    # Create a contour plot with labels using default colors.  The
    # inline argument to clabel will control whether the labels are draw
    # over the line segments of the contour, removing the lines beneath
    # the label
```

```
    # For use help, see https://matplotlib.org/
    plt.clf()
    plt.figure(figsize=(20,10))
    CS1 = plt.contour(X, Y, Z2)
    plt.clabel(CS1, inline=1, fontsize=10)
    CS2 = plt.contour(X, Y, Z1)
    plt.clabel(CS2, inline=1, fontsize=10)

    plt.title('Learned Gaussian Contours')


# Main
# Define values so simple example runs
mu0 = [0, 0]
mu1 = [1, 1]
sig0 = [1, 1]
sig1 = [1.5, 0.5]



plt_2dGaussians(mu0, mu1, sig0, sig1)
```

```
<Figure size 432x288 with 0 Axes>
```

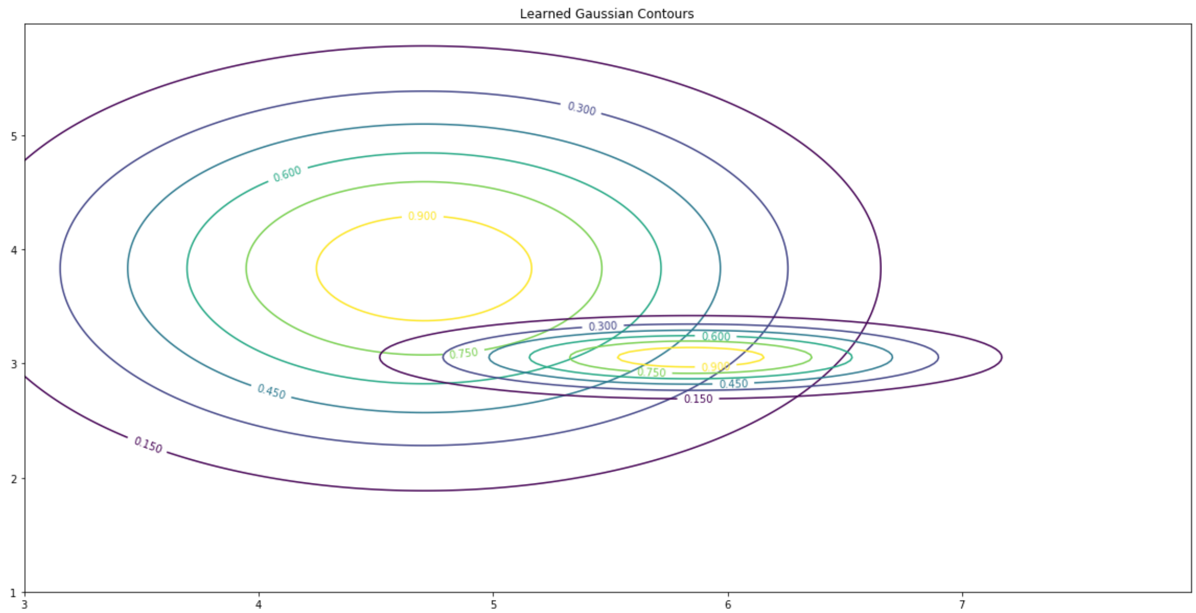


In Class

In [131]:
```
from sklearn import svm, datasets
import numpy as np


def gaussianML(X):
    mu0 = X.mean(axis = 0)
    sigma_matrix = 0
    for i in range(len(X)):
        gap = np.array([X[i] - mu0])
        sigma_matrix = sigma_matrix + np.matmul(gap.T, gap)
    sigma0 = (sigma_matrix/len(X)).diagonal().tolist()
    return mu0, sigma0


def gaussianMAP(X, priorMu, sigmaGuess):
    sigma0_0 = 0.1
    sigma0_1 = 0.1
    mu_0 = (priorMu[0] + ((sigma0_0**2.0)/(sigmaGuess[0]**2.0)) * sum(X[:, 0
]))/(1 + ((sigma0_0**2.0)/(sigmaGuess[0]**2.0) * len(X[:, 0])))
    mu_1 = (priorMu[1] + ((sigma0_1**2.0)/(sigmaGuess[1]**2.0)) * sum(X[:, 1
]))/(1 + ((sigma0_1**2.0)/(sigmaGuess[1]**2.0) * len(X[:, 1])))
    mu = [mu_0, mu_1]
    return mu, sigmaGuess

def gaussian_2d(x, y, x0, y0, xsig, ysig):
    return np.exp(-0.5*(((x-x0) / xsig)**2 + ((y-y0) / ysig)**2))


def plt_2dGaussians(mu0, mu1, sig0, sig1):
    delta = 0.025
    x = np.arange(3.0, 8.0, delta)
    y = np.arange(1.0, 6.0, delta)
    X, Y = np.meshgrid(x, y)
    Z1 = gaussian_2d(X, Y, mu0[0], mu0[1], sig0[0],sig0[1])
    Z2 = gaussian_2d(X, Y, mu1[0], mu1[1],  sig1[0],sig1[1])

    # Create a contour plot with labels using default colors.  The
    # inline argument to clabel will control whether the labels are draw
    # over the line segments of the contour, removing the lines beneath
    # the label

    # For use help, see https://matplotlib.org/
    plt.clf()
    plt.figure(figsize=(20,10))
    CS1 = plt.contour(X, Y, Z2)
    plt.clabel(CS1, inline=1, fontsize=10)
    CS2 = plt.contour(X, Y, Z1)
    plt.clabel(CS2, inline=1, fontsize=10)

    plt.title('Learned Gaussian Contours')


iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
```

```
mu0, sigma0 = gaussianML(X)

sigmaGuess = [1, 1]
priorMu = [3, 5]
mu1, sigmaGuess = gaussianMAP(X, priorMu, sigmaGuess)

plt_2dGaussians(mu0,mu1, sigma0, sigmaGuess)
```

```
<Figure size 432x288 with 0 Axes>
```



Take Home

In [9]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
img1 = mpimg.imread('IMG_0270.PNG')
plt.figure(figsize = (50, 40))
plt.imshow(img1)
plt.show()
```

HW2 Take home. Yifan Wu (yw515)

#1. $L(\Sigma) = \ln \prod_{i=1}^{N} p(x_i|\mu)$, $x_i \in \mathbb{R}^{\ell}$.

$$= \sum_{i=1}^{N} \ln p(x_i|\mu)$$

objective: $$= \sum_{i=1}^{N} \left(\ln\left[\frac{1}{2\pi^{(\ell/2)}|\Sigma|^{1/2}}\right] + \ln\left[e^{-1/2(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)}\right] \right)$$

#2. $$= \sum_{i=1}^{N} \left(\ln\left[\frac{1}{2\pi^{(\ell/2)}|\Sigma|^{1/2}}\right] - \frac{1}{2}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu) \right)$$

$$= \sum_{i=1}^{N} \ln\left[\frac{1}{2\pi^{(\ell/2)}|\Sigma|^{1/2}}\right] - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

$$= N \ln\left[\frac{1}{2\pi^{(\ell/2)}|\Sigma|^{1/2}}\right] - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

$$= N\left[\ln(1) - \ln(2\pi^{(\ell/2)}|\Sigma|^{1/2})\right] - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

$$= -N \ln(2\pi^{(\ell/2)}|\Sigma|^{1/2}) - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

$$= -N \ln(2\pi^{(\ell/2)}) + N\ln(|\Sigma|^{-1/2}) - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

Gone when taking derivative.

$$= \underline{-N \ln(2\pi^{(\ell/2)})} + \frac{1}{2}N\ln(|\Sigma|^{-1}) - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)$$

"Trace" of a matrix is the sum of diagonal elements.
Let tr[A] denote the trace of matrix A.

Property of trace: $\begin{cases} ① \ tr[ABC] = tr[CAB] = tr[BCA] \\ ② \ x^T A x = tr[x^T A x] = tr[x x^T A] \end{cases}$

We take derivative w/ respect to A.

$$tr[AB] = tr\begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vdots \\ \vec{a}_n \end{bmatrix}\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_n \end{bmatrix}$$

$$= tr\begin{bmatrix} \vec{a}_1^T\vec{b}_1 & \cdots & \cdots \\ \vdots & \vec{a}_2^T\vec{b}_2 & \\ & & \ddots \ \vec{a}_n^T\vec{b}_n \end{bmatrix}$$

$$= \sum_{i=1}^{m} a_{1i}b_{i1} + \sum_{i=1}^{m} a_{2i}b_{i2} + \cdots + \sum_{i=1}^{m} a_{ni}b_{in}$$

$$\frac{\partial tr[AB]}{\partial A} = \frac{\partial tr[AB]}{\partial a_{ij}} = b_{ji} = B^T \Rightarrow \boxed{\frac{\partial tr[BA]}{\partial a_{ij}} = B^T}$$

In [10]:
```
img2 = mpimg.imread('IMG_0265.PNG')
plt.figure(figsize = (50, 40))
plt.imshow(img2)
plt.show()
```

Also: $\frac{\partial}{\partial a_{ij}} \log|A| = \frac{1}{|A|}\frac{\partial}{\partial a_{ij}}|A| = A^{-T}$

Back to the original equation:

Since determinant of inverse is the inverse of the determinant:

$$L(\Sigma) = -N\ln(2\pi^{(d/2)}) + \frac{1}{2}N\ln(|\Sigma|^{-1}) - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)$$

$$= -N\ln(2\pi^{d/2}) + \frac{1}{2}N(\ln(|\Sigma^{-1}|)) - \frac{1}{2}\sum_{i=1}^{N}tr[(x_i-\mu)^T(x_i-\mu)\Sigma^{-1}]$$

$$\frac{\partial L(\Sigma)}{\partial \Sigma^{-1}} = \frac{N}{2}\Sigma - \frac{1}{2}\sum_n (x_i-\mu)(x_i-\mu)^T = 0$$

$$\Rightarrow \Sigma = \frac{1}{N}\sum_n (x_i-\hat{\mu})(x_i-\hat{\mu})^T$$

#3.

$$P(x|\mu) = \frac{1}{2\pi^{(d/2)}\sigma^{1/2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

$$P(\mu|\mu_0,\Sigma_0) = \frac{1}{2\pi^{(d/2)}\sigma_0^{1/2}} e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}$$

objective: $L(\mu) = \ln P(\underline{x}|\mu)P(\mu)$, Assume diagonal Covariance matrix

$$= \ln P(x|\mu)\cdot\ln(P(\mu))$$

$$\frac{L(\mu)}{\partial\mu} = \frac{\partial}{\partial\mu}\left[\ln\prod_{i=1}^{N}p(x_i|\mu) + \ln(p(\mu))\right]$$

$$= \frac{\partial}{\partial\mu}\ln\prod_{n=1}^{N}P(x_i|\mu) + \frac{\partial}{\partial\mu}\ln(P(\mu))$$

Since we already have $\frac{\partial}{\partial\mu}\ln\prod_{n=1}^{N}P(x_i|\mu) = \sum_{n=1}^{N}\frac{(x_i-\mu)}{\sigma^2}$ from lecture note,

Now we only need to calculate: $\frac{\partial L}{\partial\mu}\ln(P(\mu))$

In [11]:
```
img3 = mpimg.imread('IMG_0272.PNG')
plt.figure(figsize = (50, 40))
plt.imshow(img3)
plt.show()
```

$$\ln(p(\mu)) = \ln\left(\frac{1}{2\pi^{(k/2)}\,\sigma_0^{1/2}} \cdot e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}\right)$$

$$= \sum_{i=1}^{k} \ln\left[\frac{1}{2\pi^{(k/2)}\,\sigma_0^{1/2}}\right] + \ln\left[e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}\right]$$

$$= \sum_{i=1}^{k} \ln\left[\frac{1}{2\pi^{(k/2)}\,\sigma_0^{1/2}}\right] - \frac{(\mu-\mu_0)^2}{2\sigma_0^2}$$

$$\frac{\partial L}{\partial \mu} p(\mu) = \frac{\partial}{\partial \mu}\left(-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right)$$

$$= -\frac{1}{2\sigma_0^2} \cdot 2(\mu-\mu_0)$$

$$= -\frac{\mu-\mu_0}{\sigma_0^2}$$

Back to original objective:

$$\frac{\partial L}{\partial \mu}\left[\ln \prod_{i=1}^{N} p(x_i|\mu) + \ln(p(\mu))\right]$$

$$= \frac{\partial L}{\partial \mu} \ln \prod_{i=1}^{N} p(x_i|\mu) + \frac{\partial L}{\partial \mu} \cdot p(\mu)$$

$$= \left(\sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2}\right) - \frac{\mu-\mu_0}{\sigma_0^2}$$, Let's set this to 0:

$$\left(\sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2}\right) - \frac{\mu-\mu_0}{\sigma_0^2} = 0$$

$$\sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} = \frac{\mu-\mu_0}{\sigma_0^2}$$

$$\frac{\Sigma_{i=1}^{N} x_i}{\sigma^2} - \frac{N\mu}{\sigma^2} = \frac{\mu-\mu_0}{\sigma_0^2}$$

$$\frac{\Sigma_{i=1}^{N} x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} = \frac{\mu}{\sigma_0^2} + \frac{N\mu}{\sigma^2}$$

$$\frac{(\Sigma_{i=1}^{N} x_i)\cdot\sigma_0^2 + \mu_0\sigma^2}{\sigma^2\sigma_0^2} = \frac{\mu(\sigma^2 + N\cdot\sigma_0^2)}{\sigma^2\cdot\sigma_0^2}$$

$$\hat{\mu}_{MAP} = \frac{(\Sigma_{i=1}^{N} x_i)\sigma_0^2 + \mu_0\sigma^2}{\sigma^2 + N\sigma_0^2}$$

$$\hat{\mu}_{MAP} = \frac{\mu_0 + \frac{\sigma_0^2}{\sigma^2}(\Sigma_{i=1}^{N} x_i)}{1 + \frac{\sigma_0^2}{\sigma^2}\cdot N}$$

In [12]:
```
img4 = mpimg.imread('IMG_0273.PNG')
plt.figure(figsize = (50, 40))
plt.imshow(img4)
plt.show()
```

#4

	ML	MAP
$\hat{\mu}$.	$\frac{\sum_{i=1}^{N} x_i}{N}$	$\frac{\mu_0 + \frac{\sigma_0^2}{\sigma^2}(\sum_{i=1}^{N} x_i)}{1 + \frac{\sigma_0^2}{\sigma^2}\cdot N}$
$\hat{\sigma}^2$	$\frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{\mu})(x_i-\hat{\mu})^T$	

If we have low sample size, many prior/historical data, useful prior that is different than our observed data, we use MAP as oppose to ML.

#5.

For MAP:

objective: $L(\mu) = \ln P(\underline{x}|\mu)P(\mu)$, Assume diagonal Covariance matrix

$$= \ln P(x|\mu)\cdot \ln(P(\mu))$$

$$\frac{L(\mu)}{\partial\mu} = \frac{\partial}{\partial\mu}\left[\ln \prod_{i=1}^{N} p(x_i|\mu) + \ln(p(\mu))\right]$$

$$= \frac{\partial}{\partial\mu}\ln\prod_{i=1}^{N} P(x_i|\mu) + \frac{\partial}{\partial\mu}\ln(P(\mu))$$

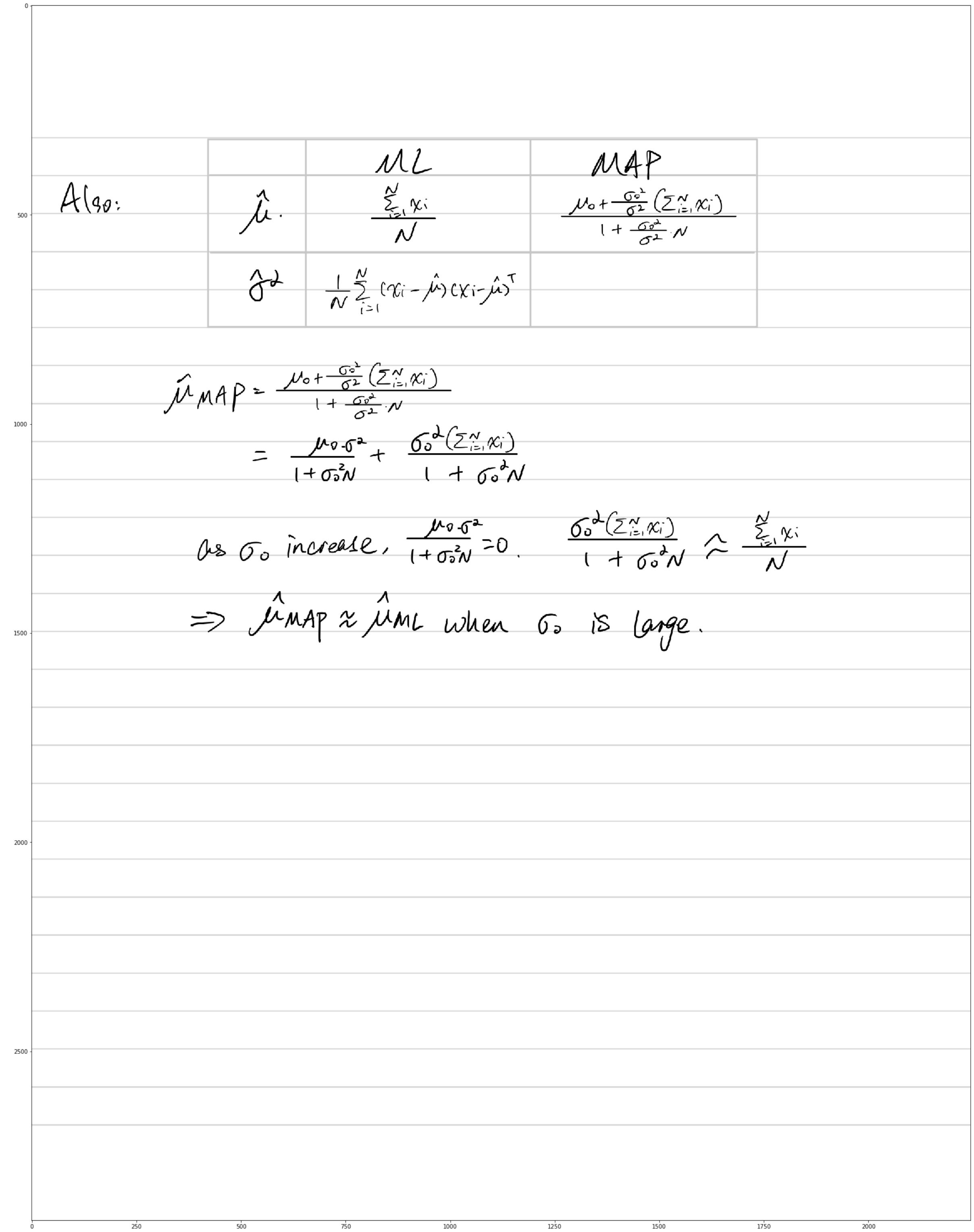
if prior $P(\mu)$ is uniform distribution, $P(\mu) = \frac{1}{a}$ for some constant a.

$$\frac{L(\mu)}{\partial\mu_{MAP}} = \frac{\partial}{\partial\mu}\ln\prod_{i=1}^{N} P(x_i|\mu) + \boxed{\frac{\partial}{\partial\mu}\ln\left(\frac{1}{a}\right)} = 0$$

$$\frac{L(\mu)}{\partial\mu_{MAP}} = \frac{\partial}{\partial\mu}\ln\prod_{i=1}^{N} P(x_i|\mu)$$

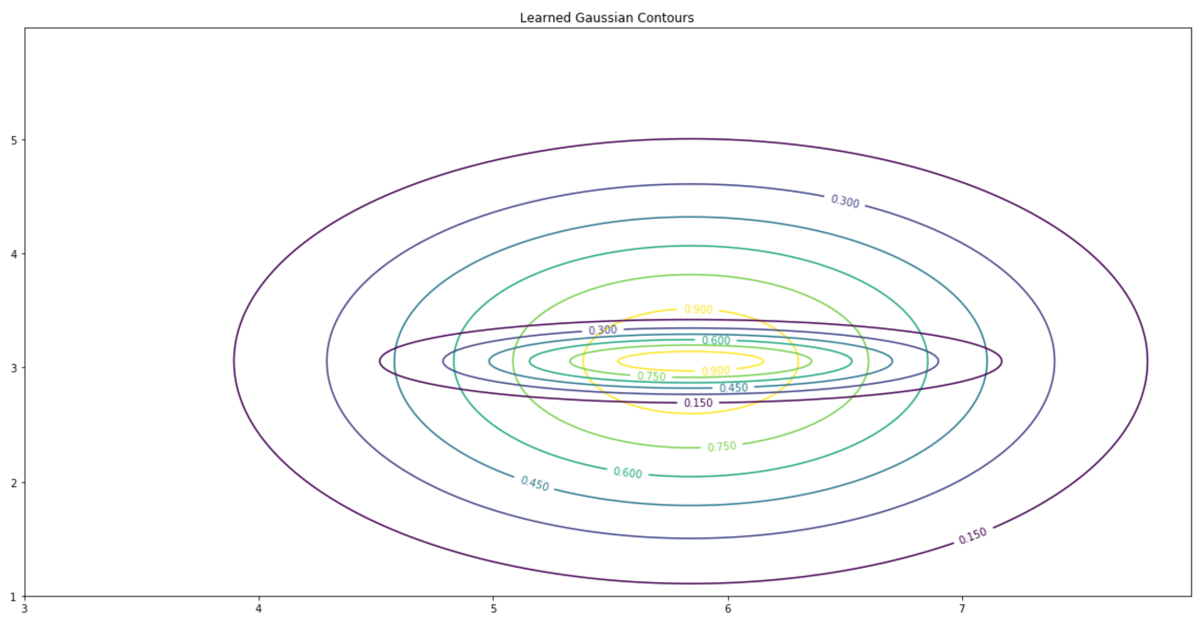
$$\frac{L(\mu)}{\partial\mu_{MAP}} = \frac{L(\mu)}{\partial\mu_{ML}}$$

In [13]:
```
img5 = mpimg.imread('IMG_0275.PNG')
plt.figure(figsize = (50, 40))
plt.imshow(img5)
plt.show()
```



In [133]:
```
#Take-home: Prove that the ML and MAP mean estimates are similar when the prio
r probability is uniform.
#Prove this semi-formally -- use math to help support your claims. ALSO Create
 plots from
#your in-class exercise to support your claim.

mu0_take_home, sigma0_take_home = gaussianML(X)

def gaussianMAP_takehome(X, priorMu, sigmaGuess):
    sigma0_0 = 10000000
    sigma0_1 = 10000000
    mu_0 = (priorMu[0] + (sigma0_0**2.0)/(sigmaGuess[0]**2.0) * sum(X[:, 0]))/
(1 + ((sigma0_0**2.0)/(sigmaGuess[0]**2.0) * len(X[:, 0])))
    mu_1 = (priorMu[1] + (sigma0_1**2.0)/(sigmaGuess[1]**2.0) * sum(X[:, 1]))/
(1 + ((sigma0_1**2.0)/(sigmaGuess[1]**2.0) * len(X[:, 1])))
    mu = [mu_0, mu_1]
    return mu, sigmaGuess


priorMu  = [3, 5]
sigmaGuess = [1, 1]
mu1_take_home, sigmaGuess_take_home = gaussianMAP_takehome(X, priorMu, sigmaGu
ess_take_home)

plt_2dGaussians(mu0_take_home, mu1_take_home, sigma0_take_home, sigmaGuess_tak
e_home)
```

<Figure size 432x288 with 0 Axes>



We can see that if we make the prior very very big, the two trojectory has the same center, which means that $\mu_{MAP} = \mu_{ML}$.