Lecture 11: Gaussian and Gaussian Mixture Classifiers

Mark Hasegawa-Johnson

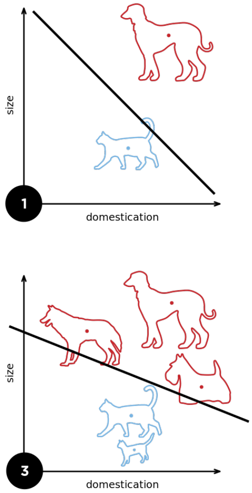
ECE 417: Multimedia Signal Processing, Fall 2021

Outline

What is a Classifier?

A **classifier** is a function $f : \mathcal{X} \to \mathcal{Y}$ that takes as input feature $x \in \mathcal{X}$, and generates an output classification label $y \in \mathcal{Y}$.

Example: A Dog/Cat Classifier



CC-BY 4.0, Elizabeth Goodspeed, 2015

Defining the Classifier

- Often, $\vec{x}$ is a real-valued feature vector of some kind, $\vec{x} \in \Re^d$, and y is a class label (a word specifying the type of object).
- Then we say that $y = f(\vec{x})$ is the label generated by the classifier.
- The task of classifier design is the task of designing the function $f(\cdot)$.

Outline

Minimum Probability of Error

Suppose we want to minimize the probability of an error. In order to analyze this problem, define three random variables like this:

- $\vec{X}$ is the observation vector, and $\vec{x}$ is its instance value.
- Y is the (unknown) true class label, and y is its instance value.
- $\hat{Y} = f(\vec{X})$ is the classifier's output hypothesis, and $\hat{y} = f(\vec{x})$ is its instance value. It might be right ($\hat{y} = y$) or it might be wrong ($\hat{y} \neq y$).

Minimum Probability of Error

Suppose we know $\vec{x}$, but we don't know y. The classifier generates $\hat{y}$. The probability of an error is

$$\Pr\{E\} = \Pr\left\{Y \neq \hat{y}\right\}$$

For any particular $\vec{x}$, we want to choose the value of $\hat{y}$ that has the lowest probability of error:

$$\begin{aligned} f(\vec{x}) &= \underset{\hat{y}}{\operatorname{argmin}} \Pr\left\{Y \neq \hat{y} | \vec{X} = \vec{x}\right\} \\ &= \underset{\hat{y}}{\operatorname{argmax}} \Pr\left\{Y = \hat{y} | \vec{X} = \vec{x}\right\} \\ &= \underset{\hat{y}}{\operatorname{argmax}}\, p_{Y|\vec{X}}(\hat{y}|\vec{x}) \end{aligned}$$

where the last line introduces a pmf notation:

$$p_{Y|\vec{X}}(y|\vec{x}) = \Pr\left\{Y = y | \vec{X} = \vec{x}\right\}$$

The Bayesian Classifier

Unfortunately, $p_{Y|\vec{X}}(y|\vec{x})$ is often very hard to calculate directly. Instead, it is often easier to calculate it using Bayes' rule:

$$\begin{aligned} f(\vec{x}) &= \operatorname*{argmax}_y p_{Y|\vec{X}}(y|\vec{x}) \\ &= \operatorname*{argmax}_y \frac{p_{\vec{X}|Y}(\vec{x}|y)p_Y(y)}{p_{\vec{X}}(\vec{x})} \end{aligned}$$

It is often much easier to learn $p_{\vec{X}|Y}(\vec{x}|y)$ from data, instead of $p_{Y|\vec{X}}(y|\vec{x})$. Let's discuss why.

The Four Bayesian Probabilities

The four probabilities in Bayes' rule are:

- The **prior** $p_Y(y)$ is the probability that $Y = y$ before you see any observations.
- The **posterior** $p_{Y|\vec{X}}(y|\vec{x})$ is the probability that $Y = y$ after you see an observation.
- The **likelihood** $p_{\vec{X}|Y}(\vec{x}|y)$ is the probability density that the class $Y = y$ might "generate" the observation $\vec{X} = \vec{x}$.
- The **evidence** $p_{\vec{X}}(\vec{x})$ is the probability density that the observation $\vec{x}$ gets generated by any natural process, regardless of the class.

The Four Bayesian Probabilities

The prior and the likelihood are usually very easy to estimate from training data. The posterior and the evidence are usually much harder to estimate. But fortunately, using Bayes' rule, we don't need to estimate the evidence! That's because:

$$\begin{aligned} f(\vec{x}) &= \underset{y}{\operatorname{argmax}}\, p_{Y|\vec{X}}(y|\vec{x}) \\ &= \underset{y}{\operatorname{argmax}}\, \frac{p_{\vec{X}|Y}(\vec{x}|y)p_Y(y)}{p_{\vec{X}}(\vec{x})} \qquad = \underset{y}{\operatorname{argmax}}\, p_{\vec{X}|Y}(\vec{x}|y)p_Y(y) \end{aligned}$$

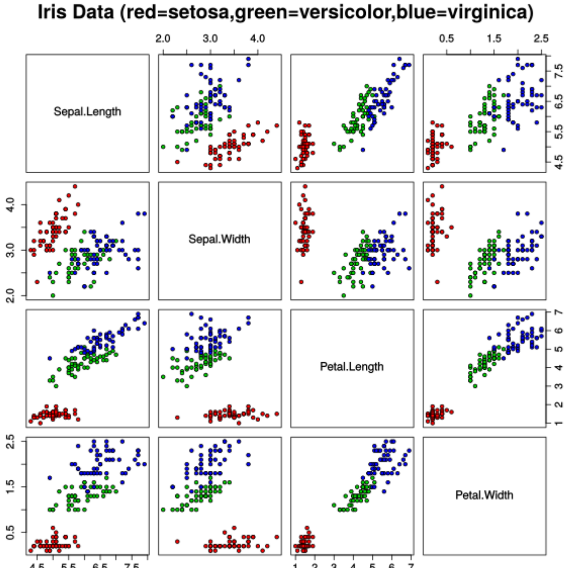
Example: Fisher's Iris Data

The rest of this lecture will use example data from Ronald Fisher's paper "The use of multiple measurements in taxonomic problems," in which he classified three different species of irises: Iris setosa, Iris versicolor, and Iris virginica.

CC-SA 3.0, Radomil, 2005; CC-SA 3.0, Dianglois, 2005; CC-SA 4.0, Eric Hunt, 2018

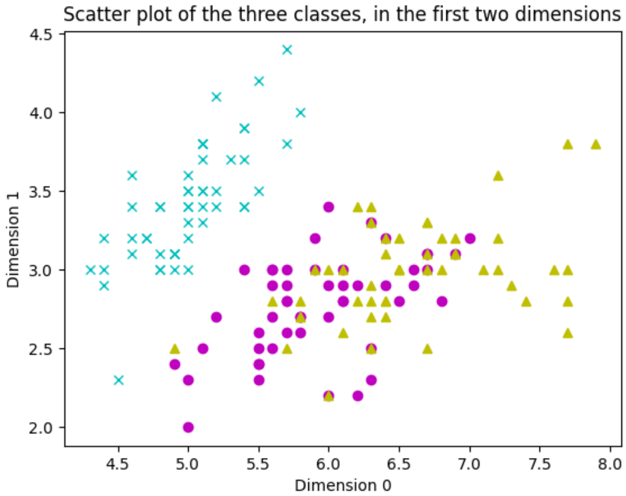
Example: Fisher's Iris Data: Likelihood

Fisher made four measurements from each flower:



Example: Fisher's Iris Data: Likelihood

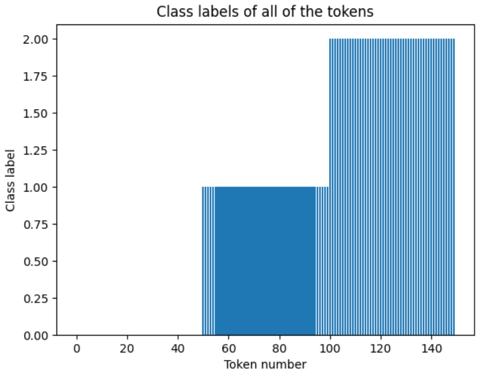
The rest of this lecture will focus on just the first two measurements from each flower: sepal length, and sepal width.



Example: Fisher's Iris Data: Prior

For convenience, we can, say, 50 examples from each of the three species, so that our prior probabilities are

$$p_Y(0) = p_Y(1) = p_Y(2) = \frac{1}{3}$$



Example: Fisher's Iris Data: Prior

Recap: in order to classify these flowers, we need to know the prior, and the likelihood. By design, we know the prior:

$$p_Y(0) = p_Y(1) = p_Y(2) = \frac{1}{3}$$

We can estimate the likelihood based on the scatterplots. But **how** can we estimate the likelihood from the scatterplots?

Outline

Gaussian Likelihoods

Notice that, in Fisher's feature space, the scatter plot for each class looks kind of elliptical. Gaussians have an elliptical shape. So let's use a Gaussian:

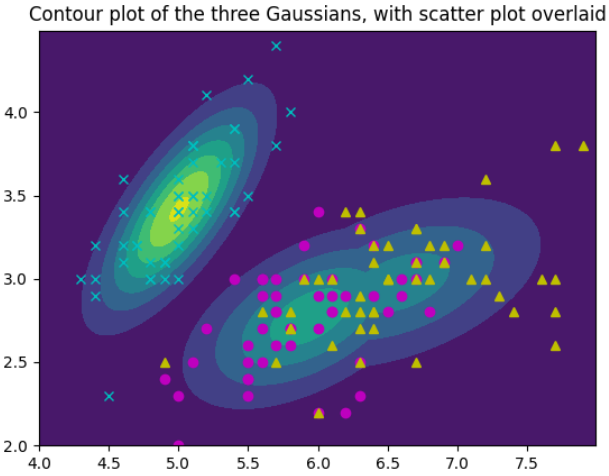
$$\begin{aligned} p_{\vec{X}|Y}(\vec{x}|y) &= \mathcal{N}(\vec{x}|\vec{\mu}_y, \Sigma_y) \\ &= \frac{1}{(2\pi)^{D/2}|\Sigma_y|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_y)^T \Sigma_y^{-1}(\vec{x}-\vec{\mu}_y)} \end{aligned}$$

where D is the vector dimension ($D = 2$, in our case), and

- $\vec{\mu}_y$ is the mean of class y,
- Σ_y is the covariance matrix of class y

Gaussian Likelihoods

Here are the contour plots of the Gaussian likelihood models that best fit each of the three data clouds:



Gaussian Classifiers

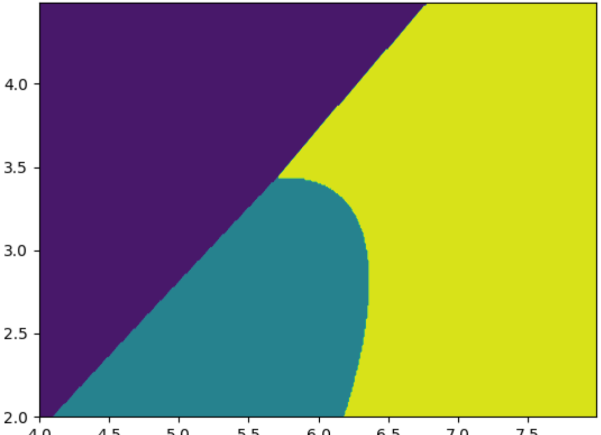
If the likelihoods are Gaussian, then the classification rule for three classes ($y \in \{0, 1, 2\}$) becomes:

$$f(\vec{x}) = \underset{y}{\operatorname{argmax}}\, p_Y(y)\mathcal{N}(\vec{x}|\vec{\mu}_y, \Sigma_y)$$

Gaussian Classifiers

Here are the classification regions. Any flower with a feature vector in the blue region is Iris sentosa, any in the green region is Iris versicolor, any in the blue region is Iris virginica.

Classifier output, yhat, plotted as a function of x



Outline

The parameter estimation problem

- In the previous slides, I showed you the "optimum" Gaussians for this data.
- Those included some "optimum" values of the parameters $\vec{\mu}_0, \vec{\mu}_1, \vec{\mu}_2, \Sigma_0, \Sigma_1, \Sigma_2$.
- How did I find those values? "Optimum" in what sense?

Maximum likelihood parameter estimation

I estimated those using a method called **maximum likelihood** parameter estimation. Maximum likelihood (ML) parameter estimation finds the parameter set, $\Theta = \{\vec{\mu}_0, \vec{\mu}_1, \vec{\mu}_2, \Sigma_0, \Sigma_1, \Sigma_2\}$, according to the following rule:

$$\Theta = \text{argmax} \sum_{i=1}^{n} \ln p(\vec{x}_i | y_i; \Theta)$$

in other words, we just choose the parameters that best explain the training data.
The **training dataset**, $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$, is a set of labeled examples: for the i^{th} flower, I know both its feature vector $\vec{x}_i$, and its true class label y_i.

ML Parameter Estimates for a Gaussian

Gaussian likelihood functions are very nice because the log-likelihood is a quadratic form:

$$\mathcal{L} = \sum_{i=1}^{n} \ln p(\vec{x}_i | y_i; \Theta)$$
$$= -\frac{1}{2} \sum_{i=1}^{n} \left(D \ln(2\pi) + \ln |\Sigma_{y_i}| + (\vec{x}_i - \vec{\mu}_{y_i})^T \Sigma_{y_i}^{-1} (\vec{x}_i - \vec{\mu}_{y_i}) \right)$$

You already know how to differentiate that:

$$\nabla_{\vec{\mu}_y} \mathcal{L} = \sum_{i:y_i=y} \Sigma_y^{-1} (\vec{x}_i - \vec{\mu}_y)$$

Left as an exercise for the reader: set that equation to 0, and solve for the optimum value of $\vec{\mu}_y$.

ML Parameter Estimates for a Gaussian

The ML estimates of the parameters for a Gaussian therefore wind up with an incredibly simple form.

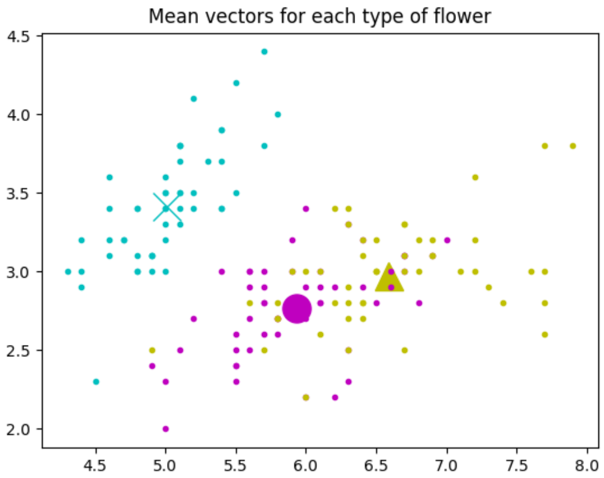
$$\vec{\mu}_y = \frac{\sum_{i=1}^n \delta_i(y)\vec{x}_i}{\sum_{i=1}^n \delta_i(y)}$$
$$\Sigma_y = \frac{\sum_{i=1}^n \delta_i(y)(\vec{x}_i - \vec{\mu}_y)(\vec{x}_i - \vec{\mu}_y)^T}{\sum_{i=1}^n \delta_i(y)}$$

where $\delta_i(y)$ is just an indicator function, counting up the number of examples in each class:

$$\delta_i(y) = \begin{cases} 1 & y_i = y \\ 0 & \text{otherwise} \end{cases}$$

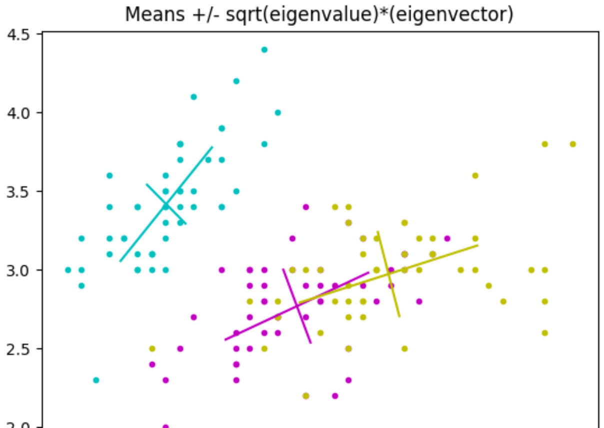
ML Estimates of the Gaussian Mean

Each mean vector is just the average of the training data for its class:



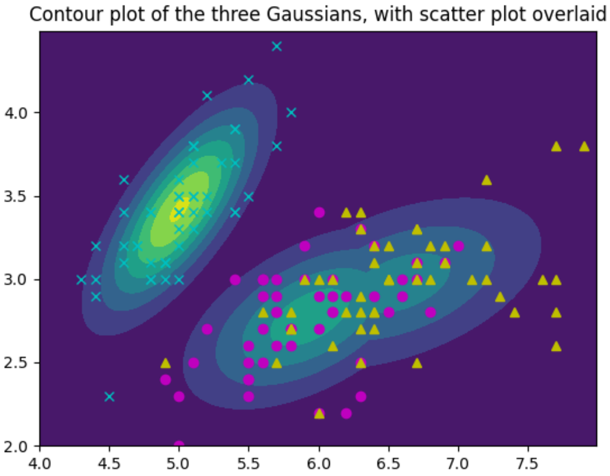
ML Estimates of the Gaussian Covariance

Each covariance matrix is just the average quadratic spread of data points around the mean. Here I'm plotting the eigenvectors of each covariance matrix, scaled by the square root of their corresponding eigenvalues:



ML Estimates of the Gaussian Parameters

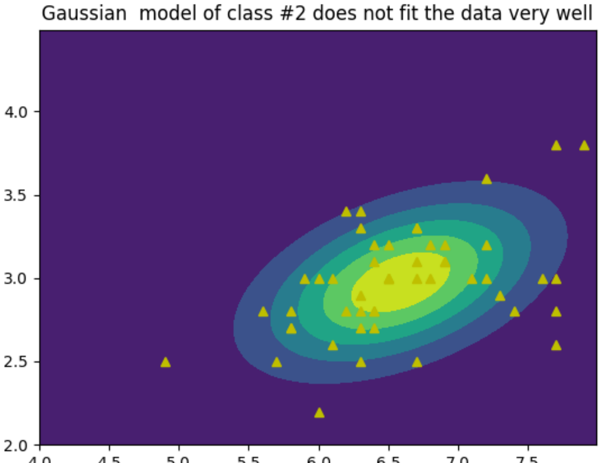
. . . and again, that gives us these Gaussian likelihood functions:



Outline

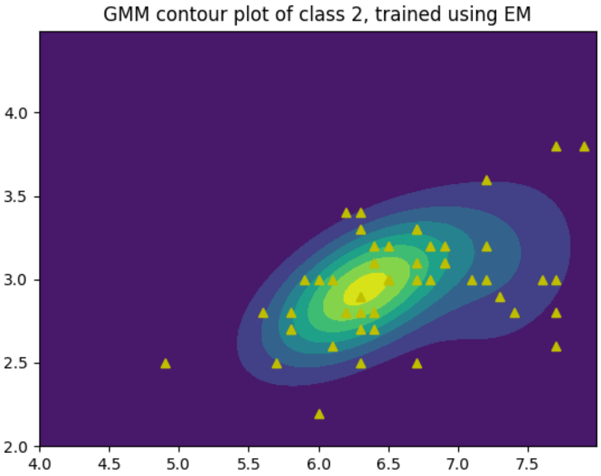
What if a Gaussian is not a good model?

Sometimes, a Gaussian is not a good model of the data. For example, in the iris dataset, classes 0 and 1 are well fit by Gaussians, but class 2 doesn't really fit very well:



Can we have more than one Gaussian per class?

One way to improve the fit is by having more than one Gaussian per class:



Gaussian Mixture Model

This is called a **Gaussian mixture model** of the likelihood (GMM). It has the following form:

$$p_{\vec{X}|Y}(\vec{x}|y) = \sum_{k=0}^{K-1} c_{y,k}\mathcal{N}(\vec{x}|\vec{\mu}_{y,k},\Sigma_{y,k})$$

where

$$c_{y,k} > 0, \quad \sum_{k=1}^{K} c_{y,k} = 1$$

Advantages and Disadvantages of GMM vs. Gaussian

Disadvantage: The GMM has more parameters to train. For example, if there are three classes, and $K = 2$ Gaussians per class, then you have to learn $3 \times 2 = 6$ mean vectors, and $3 \times 2 = 6$ covariance matrices.

Advantage: The GMM is more flexible. For example, a Gaussian can only learn a pdf with one cluster, but a GMM can learn a pdf with K clusters.

Terminology: Each of the Gaussians is called a **cluster**, the mean vector $\vec{\mu}_{y,k}$ is called the **cluster centroid** or **cluster mean**, and the covariance matrix $\Sigma_{y,k}$ is called the **cluster covariance**.

Another GMM Example

Here's an example of a 1D Gaussian mixture model with $K = 5$ clusters, from Wikipedia:
`https://commons.wikimedia.org/wiki/File:Movie.gif`

Outline

Estimating parameters for a GMM

Now we have a much larger parameter set:

$$\Theta = \{c_{y,k}, \vec{\mu}_{y,k}, \Sigma_{y,k} : y \in \{0,1,2\}, k \in \{0,1\}\}$$

How do we estimate those parameters from training data?
There are many ways it can be done. I'm going to teach you the best one, which is called the **expectation maximization** algorithm (EM).

Estimating parameters for a GMM

Surprise! The answer is exactly the same as for the Gaussian:

$$\vec{\mu}_{y,k} = \frac{\sum_{i=1}^{n} \gamma_i(y,k)\vec{x}_i}{\sum_{i=1}^{n} \gamma_i(y,k)}$$

$$\Sigma_{y,k} = \frac{\sum_{i=1}^{n} \gamma_i(y,k)(\vec{x}_i - \vec{\mu}_y)(\vec{x}_i - \vec{\mu}_y)^T}{\sum_{i=1}^{n} \gamma_i(y,k)}$$

$$c_{y,k} = \frac{\sum_{i=1}^{n} \gamma_i(y,k)}{\sum_{k=0}^{K-1} \sum_{i=1}^{N} \gamma_i(y,k)}$$

When learning the parameters for a Gaussian, we had a counting variable, $\delta_i(y)$, which was 1 if the i^{th} token was from class y, and 0 otherwise. When learning a GMM, we have a probability instead, $0 \le \gamma_i(y,k) \le 1$:

$$\gamma_i(y,k) = \Pr\left\{\vec{x}_i \text{ came from the } k^{\text{th}} \text{ cluster in class } y\right\}$$

The cluster posterior

$$
\begin{aligned}
\gamma_i(y,k) = \Pr\left\{\vec{x}_i \text{ came from the } k^{\text{th}} \text{ Gaussian of class } y\right\} \\
= \delta_i(y) \Pr\left\{\text{cluster} = k | \vec{x}_i, y\right\} \\
= \delta_i(y) \frac{\Pr\left\{\text{cluster} = k, \vec{x}_i | y\right\}}{\Pr\left\{\vec{x}_i | y\right\}} \\
= \delta_i(y) \frac{c_{y,k}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,k},\Sigma_{y,k})}{\sum_{\ell=0}^{K-1} c_{y,\ell}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,\ell},\Sigma_{y,\ell})}
\end{aligned}
$$

The cluster posterior

Wait a minute. . .

- In order to estimate the model parameters $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$, we need to know the cluster posteriors $\gamma_i(y,k)$.
- But in order to estimate the cluster posteriors, we need to know $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$!
- This is sometimes called a chicken-and-egg problem.

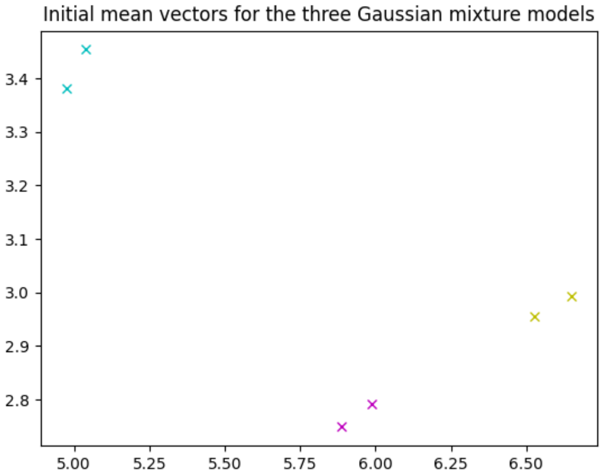
Initialization: Solving the Chicken-and-egg problem

- The way that we solve the chicken-and-egg problem is by "guessing" some initial values of $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$. Then, given our initial guess, we calculate $\gamma_i(y,k)$, and that allows us to **re-estimate** better values of $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$.
- Our initial guesses can be smart, or they can be stupid. Stupid guesses (totally random) will lead to OK solutions; smart guesses may lead to better solutions.
- One smart guess is to start with the global Gaussian means, and add some small random vector $\vec{\epsilon}$:

$$\vec{\mu}_{y,0} = \vec{\mu}_y + \vec{\epsilon}$$
$$\vec{\mu}_{y,1} = \vec{\mu}_y - \vec{\epsilon}$$

Initialization: Solving the Chicken-and-egg problem

Here are some initial guesses for the Iris dataset:



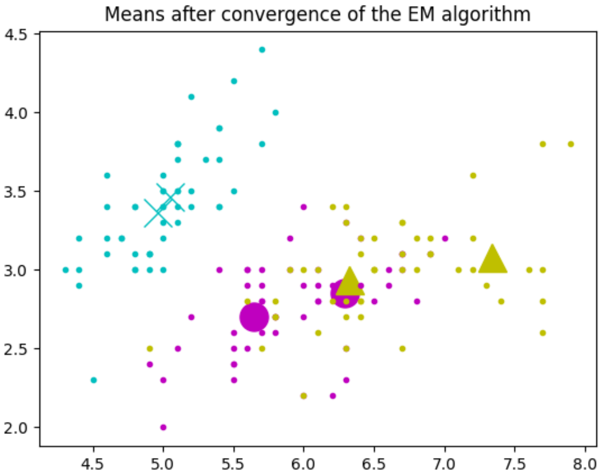
Iterate Until Convergence

Then we iterate the following until convergence:

1. Using $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$, calculate $\gamma_i(y,k)$.
2. Using $\gamma_i(y,k)$, re-estimate $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$, and $c_{y,k}$.
3. If $\vec{\mu}_{y,k}$ changed more than about 5%, then go back to step 1. Otherwise, you're done.

Iterate Until Convergence

Here are what the mean vectors look like after the EM algorithm converges:



Estimating parameters for a GMM

Putting it all together:

1. Using $\vec{\mu}_{y,k}$ and $\Sigma_{y,k}$, calculate $\gamma_i(y,k)$.

$$\gamma_i(y,k) = \delta_i(y)\frac{c_{y,k}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,k},\Sigma_{y,k})}{\sum_{\ell=0}^{K-1} c_{y,\ell}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,\ell},\Sigma_{y,\ell})}$$

2. Using $\gamma_i(y,k)$, re-estimate $\vec{\mu}_{y,k}$ and $\Sigma_{y,k}$.

$$\vec{\mu}_{y,k} = \frac{\sum_{i=1}^n \gamma_i(y,k)\vec{x}_i}{\sum_{i=1}^n \gamma_i(y,k)}$$

$$\Sigma_{y,k} = \frac{\sum_{i=1}^n \gamma_i(y,k)(\vec{x}_i-\vec{\mu}_y)(\vec{x}_i-\vec{\mu}_y)^T}{\sum_{i=1}^n \gamma_i(y,k)}$$

$$c_{y,k} = \frac{\sum_{i=1}^n \gamma_i(y,k)}{\sum_{k=0}^{K-1}\sum_{i=1}^N \gamma_i(y,k)}$$

3. If $\vec{\mu}_{y,k}$ changed more than about 5%, then go back to step 1. Otherwise, you're done.

Outline

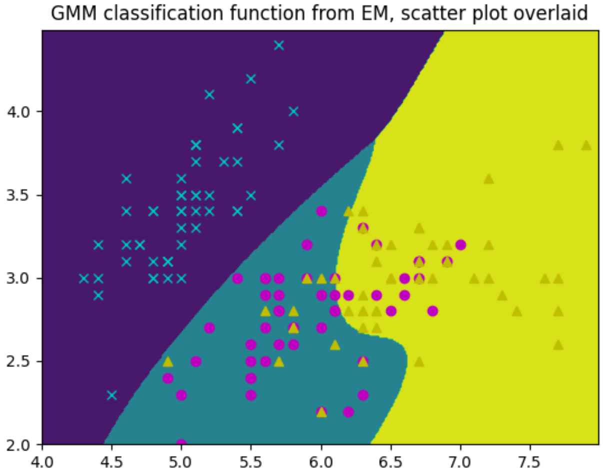
Gaussian Mixture Classifier

A Gaussian mixture classifier is a Bayesian classifier with GMM likelihoods:

$$f(\vec{x}) = \operatorname*{argmax}_y p_Y(y) p_{\vec{X}|Y}(\vec{x}|y)$$

Gaussian Mixture Classifier

Here's what that looks like for the Iris data:



Outline

Bayesian Classifier

$$f(\vec{x}) = \operatorname*{argmax}_y p_{\vec{X}|Y}(\vec{x}|y)p_Y(y)$$

ML Parameter Estimates for a Gaussian

$$\vec{\mu}_y = \frac{\sum_{i=1}^{n} \delta_i(y)\vec{x}_i}{\sum_{i=1}^{n} \delta_i(y)}$$

$$\Sigma_y = \frac{\sum_{i=1}^{n} \delta_i(y)(\vec{x}_i - \vec{\mu}_y)(\vec{x}_i - \vec{\mu}_y)^T}{\sum_{i=1}^{n} \delta_i(y)}$$

where $\delta_i(y)$ is just an indicator function, counting up the number of examples in each class:

$$\delta_i(y) = \begin{cases} 1 & y_i = y \\ 0 & \text{otherwise} \end{cases}$$

Estimating parameters for a GMM

Putting it all together:

1. Using $\vec{\mu}_{y,k}$ and $\Sigma_{y,k}$, calculate $\gamma_i(y,k)$.

$$\gamma_i(y,k) = \delta_i(y)\frac{c_{y,k}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,k},\Sigma_{y,k})}{\sum_{\ell=0}^{K-1} c_{y,\ell}\mathcal{N}(\vec{x}_i|\vec{\mu}_{y,\ell},\Sigma_{y,\ell})}$$

2. Using $\gamma_i(y,k)$, re-estimate $\vec{\mu}_{y,k}$ and $\Sigma_{y,k}$.

$$\vec{\mu}_{y,k} = \frac{\sum_{i=1}^{n}\gamma_i(y,k)\vec{x}_i}{\sum_{i=1}^{n}\gamma_i(y,k)}$$

$$\Sigma_{y,k} = \frac{\sum_{i=1}^{n}\gamma_i(y,k)(\vec{x}_i-\vec{\mu}_y)(\vec{x}_i-\vec{\mu}_y)^T}{\sum_{i=1}^{n}\gamma_i(y,k)}$$

$$c_{y,k} = \frac{\sum_{i=1}^{n}\gamma_i(y,k)}{\sum_{k=0}^{K-1}\sum_{i=1}^{N}\gamma_i(y,k)}$$

3. If $\vec{\mu}_{y,k}$ changed more than about 5%, then go back to step 1. Otherwise, you're done.