ECE276A: Sensing & Estimation in Robotics
Lecture 5: Gaussian Discriminant Analysis

Instructor:

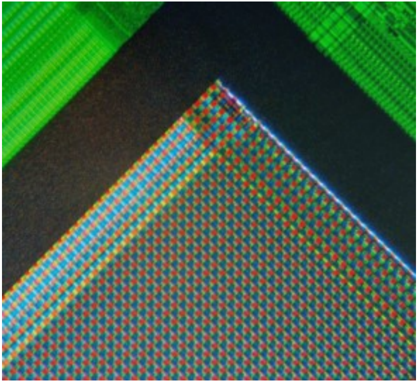
Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qjfeng@ucsd.edu
Arash Asgharivaskasi: aasghari@eng.ucsd.edu
Ehsan Zobeidi: ezobeidi@ucsd.edu
Rishabh Jangir: rjangir@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Color Imaging



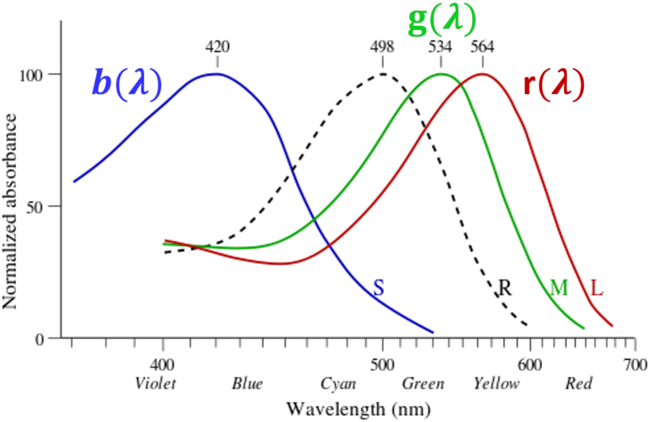
- Image sensor: converts light into small bursts of current
- Analog imaging technology uses charge-coupled devices (CCD) or complementary metal-oxide semiconductors (CMOS)
- CCD/CMOS photosensor array:
  - A phototransistor converts light into current
  - Each transistor charges a capacitor to measure:

    #photons/sampling time
  - R,G,B filters are used to modify the absorption profiles of photons
- Analog-to-digital conversion of R,G,B transistor values to pixel values:

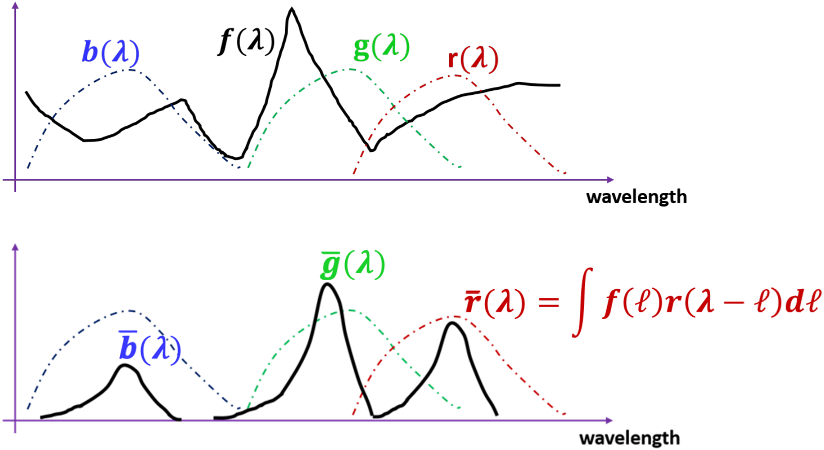
$$\underbrace{R = 127}_{\text{8 bits (0-255)}}, \quad G = 200, \quad B = 103 \quad (\textbf{24-bit color})$$

Why RGB, Why 3?

- **Retina**: types of photoreceptors: **rod** & **cone** cells (S,M,L)
- **Rod cells**:
  - insensitive to wavelength but highly sensitive to intensity
  - mostly saturated during daylight conditions



- **Cone cells**:
  - Given an arbitrary light spectral distribution $f(\lambda)$, the cone cells act as filters that provide a **convolution-like signal** to the brain:

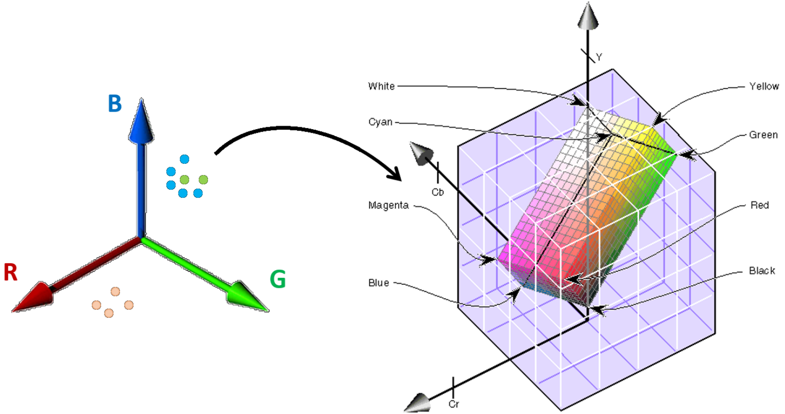


  - Color blind people are deficient in 1 or more of these cones
  - Other animals (e.g., fish) have more than 3 cones

Luma-Chroma Color Space

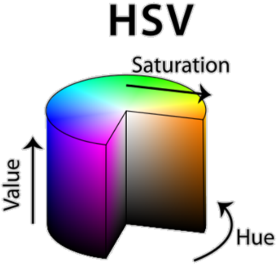
- **YUV (YCbCr)**: a linear transformation of RGB
  - Luminance/Brightness (Y) $\approx (R + G + B)/3$ } **gray-scale image**
  - Blueness (U/Cb) $\approx (B - G)$
  - Redness (V/Cr) $\approx (R - G)$

  } **chrominance**
- Used in analog TV for PAL/SECAM composite color video standards



HSV and LAB Color Spaces

- **HSV**: cylindrical coordinates of RGB points
  - **Hue (H)**: angular dimension (red $\approx 0°$, green $\approx 120°$, blue $\approx 240°$)
  - **Saturation (S)**: pure red has saturation 1, while tints have saturation < 1
  - **Value/Brightness (V)**: achromatic/gray colors ranging from black ($V = 0$, bottom) to white ($V = 1$, top)



- **LAB**: nonlinear transformation of RGB; device independent
  - Lightness (L): from black ($L = 0$) to white ($L = 100$)
  - Position between green and red/magenta (A)
  - Position between blue and yellow (B)

Image Formation

- Pixel values depend on:
  - Scene **geometry**
  - Scene **photometry** (illumination and reflective properties)
  - Scene **dynamics** (moving objects)
- Using camera images to infer a representation of the world is challenging because the shape, material properties, and motion of the observed scene are in general unknown
- We will study several basic problems:
  - Classifying pixels into semantic colors, such as red, blue, green, etc.
  - Projecting pixel coordinates from 2-D image space to 3-D world space
  - Extracting and tracking image point features
  - Estimating the 3-D world-frame postions of image point features
  - Estimating the position and orientation of a camera observing image point features

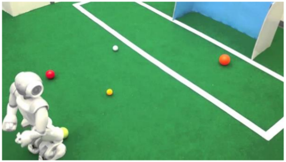
Color Segmentation as a Classification Problem

- **Color Segmentation Problem**: segment the 3-D color space into a set of volumes associated with different colors:
  - Each pixel is a **3-D vector**: $\mathbf{x} = (R, G, B)$ or (Y, Cb, Cr) or (H, S, V)
  - Discrete color labels: $y \in \{1, \ldots, K\}$, e.g., $\{\text{red}, \text{blue}, \text{yellow}, \ldots\}$
- Pixel values are noisy. We will use a **probabilistic model** $p(y \mid \mathbf{x})$ for the color class $y \in \{1, \ldots, K\}$ of a given pixel $\mathbf{x} \in \mathbb{R}^3$
- **Training**: given data $D = \{(\mathbf{x}_i, y_i)\}_i$ of pixel values $\mathbf{x}_i \in \mathbb{R}^3$, labeled with colors y_i, optimize the parameters of a probabilistic model $p(y \mid \mathbf{x})$
- **Testing**: use the optimized model $p(y \mid \mathbf{x})$ to define a function that transforms a new color-space input $\mathbf{x}_*$ into a discrete color label y_*:

$$\mathbf{x}_* \xrightarrow{\text{classifier}} y_* \in \arg\max_y p(y \mid \mathbf{x}_*)$$

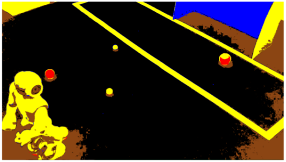
Color-based Object Detection

Robot Soccer Example: real-time robot vision system



RGB color image at 30 fps from camera

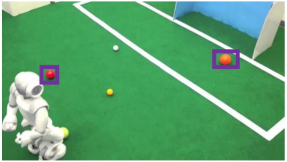
Color Segmentation



Each pixel is labelled by symbolic colors

Union-find algorithm

Connected components (blobs)



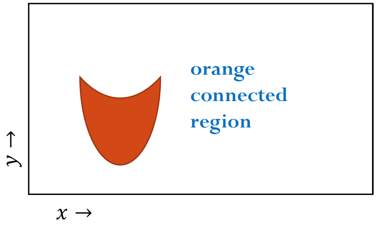
Extract region properties: centroid, bounding box, major/minor axis, etc.

Classify objects based on shape

Shape-based Orange Ball Detection

- Given a set of orange pixels with image plane coordinates $\{(u_p, v_p)\}_{p=1}^{N_p}$
- Center of mass:
  $(c_u, c_v) = (\frac{1}{N_p} \sum_p u_p, \frac{1}{N_p} \sum_p v_p)$

Color image:



- Fit an ellipse: $\left\{ (u, v) \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} u - c_u \\ v - c_v \end{bmatrix}^\top \begin{bmatrix} E_{uu} & E_{uv} \\ E_{uv} & E_{vv} \end{bmatrix}^{-1} \begin{bmatrix} u - c_u \\ v - c_v \end{bmatrix} \leq 1 \right\}$

$$E_{uu} = \frac{2}{N_p} \sum_p (u_p - c_u)^2 \qquad E_{vv} = \frac{2}{N_p} \sum_p (v_p - c_v)^2 \qquad E_{uv} = \frac{2}{N_p} \sum_p (u_p - c_u)(v_p - c_v)$$

- Ball detection: use thresholds ϵ_0, ϵ_1 on the eigenvalues λ_0, λ_1 of $\begin{bmatrix} E_{uu} & E_{uv} \\ E_{uv} & E_{vv} \end{bmatrix}$
  - size: $\min \lambda_1, \lambda_2 \geq \epsilon_0$
  - eccentricity: $1 - \epsilon_1 \leq \frac{\lambda_1}{\lambda_2} \leq 1 + \epsilon_1$

Project 1: Color Segmentation

- Train a probabilistic color model using a set of labeled pixel values
- Use the model to classify the colors on unseen test images
- Detect a blue recycling bin based on the color segmentation and the known bin shape

Project 1 Tips

- Define K color classes that you want to distinguish, e.g., recycling-bin-blue, other-blue, green, brown, etc. At the very least, you should have $K = 2$ color classes: blue and not-blue.
- Label examples (by selecting pixel regions via **roipoly**) for each color class to obtain a training dataset $D = \{\mathbf{x}_i, y_i\}$.
- Train a generative (Gaussian Discriminant) model $p(y, \mathbf{x})$ or discriminative (Logistic Regression) model $p(y \mid \mathbf{x})$
- Given a test image, classify each pixel into one of the K color classes using your model
- Find recycling-bin-blue regions (via OpenCV's **findContours**)
- Enumerate blue region combinations and score them based on how much they resemble a bin shape (via Scikit-Image's **regionprops**)
- Experiment with different color spaces and parameters

Supervised Learning

- **Data**: a set $D := \{\mathbf{x}_i, y_i\}_{i=1}^n$ of **iid** examples $\mathbf{x}_i \in \mathbb{R}^d$ with associated scalar labels y_i generated from an **unknown** joint pdf $p_*(y, \mathbf{x})$
- The training dataset is often also written in matrix notation, $D = (X, \mathbf{y})$, with $X \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$
- **Goal**: define a function: $h : \mathbb{R}^d \to \mathbb{R}$ that can assign a label y to a given data point $\mathbf{x}$, either from the training dataset D or from an unseen test set generated from the **same** unknown pdf $p_*(y, \mathbf{x})$
- The function h should perform "well":
  - **Classification** (discrete $y \in \{-1, 1\}$):
    $$\min_h Loss_{0\text{-}1}(h) := \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{h(\mathbf{x}_i) \neq y_i} = \#\text{ of times } h \text{ is wrong about the labels}$$
  - **Regression** (continuous $y \in \mathbb{R}$):
    $$\min_h MSE(h) := \frac{1}{n}\sum_{i=1}^n (h(\mathbf{x}_i) - y_i)^2 = \text{mean square error of } h$$

Generative vs Discriminative Models

- **Generative model**
  - $h(\mathbf{x}) := \arg\max_y p(y, \mathbf{x})$
  - Choose $p(y, \mathbf{x})$ so that it approximates the unknown data-generating pdf
  - Can generate new examples $\mathbf{x}$ with labels y by sampling from $p(y, \mathbf{x})$
  - **Examples**: Naïve Bayes, Gaussian Discriminant Analysis, Hidden Markov Models, Restricted Boltzmann Machines, Latent Dirichlet Allocation, etc.

- **Discriminative model**
  - $h(\mathbf{x}) := \arg\max_y p(y|\mathbf{x})$
  - Choose $p(y|\mathbf{x})$ so that it approximates the unknown label-generating pdf
  - Because it models $p(y|\mathbf{x})$ directly, a discriminative model cannot generate new examples $\mathbf{x}$ but given $\mathbf{x}$ it can predict (discriminate) y.
  - **Examples**: Linear Regression, Logistic Regression, Support Vector Machines, Neural Networks, Random Forests, Conditional Random Fields, etc.

Parametric Learning

- Represent the pdfs $p(y|\mathbf{x};\boldsymbol{\omega})$ (discriminative) or $p(y,\mathbf{x};\boldsymbol{\omega})$ (generative) using parameters $\boldsymbol{\omega} \in \mathbb{R}^m$
- Estimate/optimize/learn $\boldsymbol{\omega}$ based on the training set $D = (X, \mathbf{y})$ in a way that the optimized parameters $\boldsymbol{\omega}^*$ produce good results on a test set $D_* = (X_*, \mathbf{y}_*)$
- Parameter estimation strategies:
  - **Maximum Likelihood Estimation (MLE)**: maximize the likelihood of the data D given the parameters $\boldsymbol{\omega}$
  - **Maximum A Posteriori (MAP)**: maximize the likelihood of the parameters $\boldsymbol{\omega}$ given the data D
  - **Bayesian Inference**: estimate the whole distribution of the parameters $\boldsymbol{\omega}$ given the data D

Parametric Learning

- **Maximum Likelihood Estimation (MLE)**:

MLE	Discriminative Model	Generative Model
Training	$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} p(\mathbf{y} \mid X, \boldsymbol{\omega})$	$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} p(\mathbf{y}, X \mid \boldsymbol{\omega})$
Testing	$y_* \in \arg\max_{y} p(y \mid \mathbf{x}_*, \boldsymbol{\omega}^*)$	$y_* \in \arg\max_{y} p(y, \mathbf{x}_* \mid \boldsymbol{\omega}^*)$

- **Maximum A Posteriori (MAP)**:

MAP	Discriminative Model	Generative Model
Training	$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} p(\boldsymbol{\omega} \mid \mathbf{y}, X) = \arg\max_{\boldsymbol{\omega}} p(\mathbf{y} \mid X, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid X)$	$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} p(\boldsymbol{\omega} \mid \mathbf{y}, X) = \arg\max_{\boldsymbol{\omega}} p(\mathbf{y}, X \mid \boldsymbol{\omega}) p(\boldsymbol{\omega})$
Testing	$y_* \in \arg\max_{y} p(y \mid \mathbf{x}_*, \boldsymbol{\omega}^*)$	$y_* \in \arg\max_{y} p(y, \mathbf{x}_* \mid \boldsymbol{\omega}^*)$

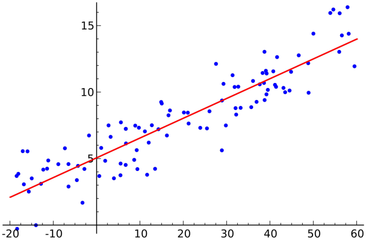
- **Bayesian Inference**:

BI	Discriminative Model	Generative Model
Training	$p(\boldsymbol{\omega} \mid \mathbf{y}, X) \propto p(\mathbf{y} \mid X, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid X)$	$p(\boldsymbol{\omega} \mid \mathbf{y}, X) \propto p(\mathbf{y}, X \mid \boldsymbol{\omega}) p(\boldsymbol{\omega})$
Testing	$p(y_* \mid \mathbf{x}_*, \mathbf{y}, X) = \int p(y_* \mid \mathbf{x}_*, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid \mathbf{y}, X) d\boldsymbol{\omega}$	$p(y_*, \mathbf{x}_* \mid \mathbf{y}, X) = \int p(y_*, \mathbf{x}_* \mid \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid \mathbf{y}, X) d\boldsymbol{\omega}$

Discriminative Regression via a Linear Gaussian Model

- Model the relationship between an example $x \in \mathbb{R}$ and its label $y \in \mathbb{R}$ as linear but corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$

$$y = \omega_1 x + \omega_0 + \epsilon$$



- For simplicity, assume σ^2 is known
- Given data $D = \{(x_i, y_i)\}_i$, estimate the slope ω_1 and intercept ω_0
- Combine the two parameters into a vector by augmenting x with 1, i.e.,

$$\omega_1 x + \omega_0 = \underbrace{\begin{bmatrix} \omega_1 & \omega_0 \end{bmatrix}}_{\boldsymbol{\omega}} \underbrace{\begin{bmatrix} x \\ 1 \end{bmatrix}}_{\mathbf{x}}$$

- We have a discriminative model using the Gaussian pdf ϕ:

$$y \sim \mathcal{N}(\boldsymbol{\omega}^\top \mathbf{x}, \sigma^2) \quad \Leftrightarrow \quad p(y|\mathbf{x}, \boldsymbol{\omega}) = \phi(y; \boldsymbol{\omega}^\top \mathbf{x}, \sigma^2)$$

Discriminative Regression via a Linear Gaussian Model

- **Linear regression** uses a discriminative model $p(\mathbf{y}|X, \boldsymbol{\omega})$ for the continuous labels $\mathbf{y} \in \mathbb{R}^n$ that is Gaussian and linear in $X \in \mathbb{R}^{n \times d}$:

$$p(\mathbf{y}|X, \boldsymbol{\omega}) = \phi(\mathbf{y}; X\boldsymbol{\omega}, V)$$

- Use MLE to estimate the parameters:

$$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} p(\mathbf{y}|X, \boldsymbol{\omega}) = \arg\max_{\boldsymbol{\omega}} \log p(\mathbf{y}|X, \boldsymbol{\omega})$$

- Transforming the objective by a monotone function (log) does not affect the maximizer but conditions the data numerically

$$\begin{aligned}\log p(\mathbf{y}|X, \boldsymbol{\omega}) &= \log\left(\frac{1}{\sqrt{(2\pi)^n \det(V)}} \exp\left(-\frac{1}{2}(\mathbf{y} - X\boldsymbol{\omega})^\top V^{-1}(\mathbf{y} - X\boldsymbol{\omega})\right)\right) \\ &= \underbrace{-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det V}_{\text{independent of } \boldsymbol{\omega}} - \frac{1}{2}(\mathbf{y} - X\boldsymbol{\omega})^\top V^{-1}(\mathbf{y} - X\boldsymbol{\omega})\end{aligned}$$

Discriminative Regression via a Linear Gaussian Model

- MLE using the data log-likelihood we derived:

$$\begin{aligned}\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} \log p(\mathbf{y} \mid X, \boldsymbol{\omega}) &= \arg\min_{\boldsymbol{\omega}} \frac{1}{2}(\mathbf{y} - X\boldsymbol{\omega})^\top V^{-1}(\mathbf{y} - X\boldsymbol{\omega}) \\ &= \arg\min_{\boldsymbol{\omega}} \frac{1}{2}\|V^{-1/2}(\mathbf{y} - X\boldsymbol{\omega})\|_2^2\end{aligned}$$

- To solve the unconstrained optimization, set the gradient equal to 0:

$$0 = \nabla_{\boldsymbol{\omega}} \left(\frac{1}{2}\|V^{-1/2}(\mathbf{y} - X\boldsymbol{\omega})\|_2^2\right) = -X^\top V^{-1}(\mathbf{y} - X\boldsymbol{\omega})$$

- and solve for $\boldsymbol{\omega}$:

$$\boldsymbol{\omega}^* = (X^\top V^{-1} X)^{-1} X^\top V^{-1} \mathbf{y}$$

Discriminative Regression via a Linear Gaussian Model

- **Ridge regression**: obtains a MAP estimate for $\boldsymbol{\omega}$
- Assume a Gaussian prior $\boldsymbol{\omega} \sim \mathcal{N}(0, \Lambda)$ on the parameters so that:

$$\log p(\boldsymbol{\omega}) \propto -\frac{1}{2}\boldsymbol{\omega}^\top \Lambda^{-1} \boldsymbol{\omega}$$

- The MAP estimate of $\boldsymbol{\omega}$ is:

$$\begin{aligned}
\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega}} \; & \log p(\mathbf{y} \mid X, \boldsymbol{\omega}) + \log p(\boldsymbol{\omega}) \\
= \arg\min_{\boldsymbol{\omega}} \; & \frac{1}{2}(\mathbf{y} - X\boldsymbol{\omega})^\top V^{-1}(\mathbf{y} - X\boldsymbol{\omega}) + \frac{1}{2}\boldsymbol{\omega}^\top \Lambda^{-1}\boldsymbol{\omega} \\
= \arg\min_{\boldsymbol{\omega}} \; & \frac{1}{2}\|V^{-1/2}(\mathbf{y} - X\boldsymbol{\omega})\|_2^2 + \underbrace{\frac{1}{2}\|\Lambda^{-1/2}\boldsymbol{\omega}\|_2^2}_{\textbf{regularization}} \\
= \; & \boxed{(X^\top V^{-1} X + \Lambda^{-1})^{-1} X^\top V^{-1} \mathbf{y}}
\end{aligned}$$

- The optimization is equivalent to the MLE setting but includes (Tikhonov) regularization on $\boldsymbol{\omega}$

Linear Regression Summary

- **Linear Regression** uses a discriminative model: $p(\mathbf{y}|X,\boldsymbol{\omega}) = \phi(\mathbf{y}; X\boldsymbol{\omega}, V)$
- **Ridge Regression** uses a prior $p(\boldsymbol{\omega}) = \phi(\boldsymbol{\omega}; \mathbf{0}, \Lambda)$ in addition
- **Training**: given data $D = (X, \mathbf{y})$, optimize the model parameters:
  - MLE: $\boldsymbol{\omega}^* = (X^\top V^{-1} X)^{-1} X^\top V^{-1} \mathbf{y}$
  - MAP: $\boldsymbol{\omega}^* = (X^\top V^{-1} X + \Lambda^{-1})^{-1} X^\top V^{-1} \mathbf{y}$
- **Testing**: given a test example $\mathbf{x}_* \in \mathbb{R}^d$, use the optimized parameters $\boldsymbol{\omega}^*$ to predict the label:

$$y_* = \arg\max_y \log p(y \mid \mathbf{x}_*, \boldsymbol{\omega}^*) = \mathbf{x}_*^\top \boldsymbol{\omega}^*$$

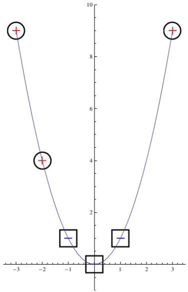
- The test expression is obtained from the gradient of the log-likelihood with respect to y:

$$0 = \nabla_y \left(\frac{1}{2} \| V^{-1/2} (y - \mathbf{x}_*^\top \boldsymbol{\omega}^*) \|_2^2 \right) = V^{-1} (y - \mathbf{x}_*^\top \boldsymbol{\omega}^*)$$

Linear Regression Example

- Consider the following dataset:

$$X = \begin{bmatrix} -3 & 9 & 1 \\ -2 & 4 & 1 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 3 & 9 & 1 \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad \mathbf{y} = \begin{bmatrix} +1 \\ +1 \\ -1 \\ -1 \\ -1 \\ +1 \end{bmatrix} \in \mathbb{R}^{n}$$



- Adding an extra dimension of 1s is a trick to allow an affine model:

$$X'\boldsymbol{\omega}_1 + \omega_0 \mathbf{1} = \underbrace{\begin{bmatrix} X' & \mathbf{1} \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \boldsymbol{\omega}_1 \\ \omega_0 \end{bmatrix}}_{\boldsymbol{\omega}}$$

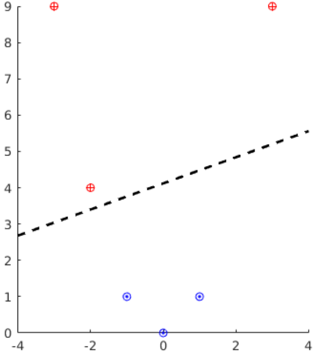
- Let the discrminative model be:

$$p(\mathbf{y}|X, \boldsymbol{\omega}) = \phi(\mathbf{y}; X\boldsymbol{\omega}, V) \qquad V = I_n$$
$$p(\boldsymbol{\omega} \mid X) = \phi(\boldsymbol{\omega}; \mathbf{0}, \Lambda) \qquad \Lambda = 2I_d$$

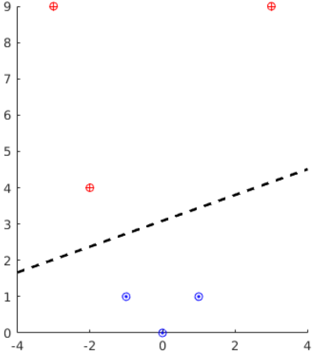
Linear Regression Example

- **Training**:
  - MLE: $\boldsymbol{\omega}^* = (X^\top V^{-1} X)^{-1} X^\top V^{-1} \mathbf{y} = \begin{bmatrix} -0.0857 \\ 0.2381 \\ -0.9810 \end{bmatrix}$
  - MAP: $\boldsymbol{\omega}^* = (X^\top V^{-1} X + \Lambda^{-1})^{-1} X^\top V^{-1} \mathbf{y} = \begin{bmatrix} -0.0643 \\ 0.1806 \\ -0.5580 \end{bmatrix}$
- **Testing**: the decision boundary is a line with equation $0 = \mathbf{x}^\top \boldsymbol{\omega}^*$:

MLE:



MAP:



Generative Classification via a Naïve Bayes Model

- **Naïve Bayes** uses a **generative model** $p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta})$ for discrete labels $\mathbf{y} \in \{1, \ldots, K\}^n$ and *assumes* (naively) that, when conditioned on y_i, the dimensions of an example x_{il} for $l = 1, \ldots, d$ are independent

- Naïve Bayes uses one set of parameters $\boldsymbol{\theta}$ to model the marginal pdf $p(y)$ of a label y and one set of parameters $\boldsymbol{\omega}$ to model the conditional pdf $p(x_l|y, \boldsymbol{\omega})$ of a single dimension of an example $\mathbf{x}$:

$$p(y, \mathbf{x} \mid \boldsymbol{\omega}, \boldsymbol{\theta}) = p(y \mid \boldsymbol{\theta}) p(\mathbf{x} \mid y, \boldsymbol{\omega}) \overset{\text{naïve}}{\underset{\text{assumption}}{=\!=\!=\!=\!=}} p(y \mid \boldsymbol{\theta}) \prod_{l=1}^{d} p(x_l \mid y, \boldsymbol{\omega})$$

$$\begin{aligned} p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta}) &= p(\mathbf{y} \mid \boldsymbol{\theta}) p(X \mid \mathbf{y}, \boldsymbol{\omega}) \\ &= \left(\prod_{i=1}^{n} p(y_i \mid \boldsymbol{\theta}) \right) \left(\prod_{i=1}^{n} \prod_{l=1}^{d} p(x_{il} \mid y_i, \boldsymbol{\omega}) \right) \end{aligned}$$

Gaussian Naïve Bayes

- GNB uses a Categorical distribution to model $p(y_i \mid \boldsymbol{\theta})$ and a Gaussian distribution to model $p(x_{il} \mid y_i, \boldsymbol{\omega})$ for $x_{il} \in \mathbb{R}$ and $\boldsymbol{\omega} := \{\mu_{kl}, \sigma_{kl}^2\}$

$$p(y_i \mid \boldsymbol{\theta}) := \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \qquad p(x_{il} \mid y_i = k, \boldsymbol{\omega}) := \phi\left(x_{il}; \mu_{kl}, \sigma_{kl}^2\right)$$

- GNB obtains the following MLE estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$:

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = k\}$$

$$\mu_{kl}^{MLE} = \frac{\sum_{i=1}^{n} x_{il} \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}} \quad \sigma_{kl}^{MLE} = \sqrt{\frac{\sum_{i=1}^{n} (x_{il} - \mu_{kl}^{MLE})^2 \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}}$$

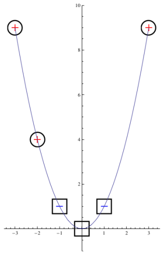
- Given a test example $\mathbf{x}_* \in \mathbb{R}^d$, the GNB classifier produces the output:

$$y_* = \underset{y \in \{1, \ldots, K\}}{\arg\max} \; \log \theta_y^{MLE} + \sum_{l=1}^{d} \log \phi\left(x_{*l}; \mu_{yl}^{MLE}, \left(\sigma_{yl}^{MLE}\right)^2\right)$$

Gaussian Naïve Bayes Example

- Consider the same data as before:

$$X = \begin{bmatrix} -3 & 9 \\ -2 & 4 \\ -1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 3 & 9 \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad \mathbf{y} = \begin{bmatrix} +1 \\ +1 \\ -1 \\ -1 \\ -1 \\ +1 \end{bmatrix} \in \mathbb{R}^{n}$$



- **Training**: The GNB MLE parameters are:

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = k\} = \frac{1}{2} \quad \text{for } k = 1, -1$$

$$\mu_{kl}^{MLE} = \frac{\sum_{i=1}^{n} x_{il} \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}} =$$

	$l = 1$	$l = 2$
$k = 1$	-0.66	7.33
$k = -1$	0.00	0.66

$$\sigma_{kl}^{MLE} = \sqrt{\frac{\sum_{i=1}^{n} (x_{il} - \mu_{kl}^{MLE})^2 \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}} =$$

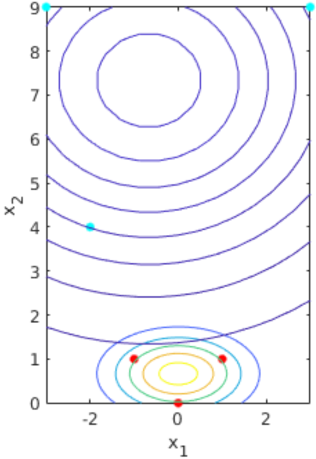
	$l = 1$	$l = 2$
$k = 1$	2.62	2.36
$k = -1$	0.82	0.47

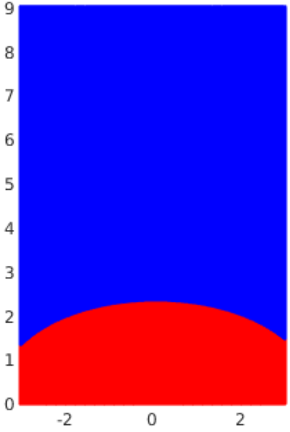
Gaussian Naïve Bayes Example

- **Testing**: evaluate the most likely class:

$$y_* = \arg\min_{k \in \{-1,+1\}} \left\{ \log \frac{1}{\theta_k^2} + \sum_{l=1}^{d} \log \sigma_{kl}^2 + \frac{(x_{*l} - \mu_{kl})^2}{\sigma_{kl}^2} \right\}$$

- The decision boundary is **not linear** (in contrast to logistic regression):





Categorical Naïve Bayes

- CNB is used when both the labels y_i and the examples $\mathbf{x}_i$ are discrete

- CNB uses a Categorical distribution to model $p(y_i \mid \boldsymbol{\theta})$ and $p(x_{il} \mid y_i, \boldsymbol{\omega})$ for $x_{il} \in \{1, \ldots, J\}$ as follows:

$$p(y_i \mid \boldsymbol{\theta}) := \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \qquad p(X_{il} \mid y_i, \boldsymbol{\omega}) := \prod_{k=1}^{K} \prod_{j=1}^{J} \omega_{kj}^{\mathbb{1}\{X_{il} = j, y_i = k\}}$$

- CNB obtains these MLE estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ with regularization $r \in \mathbb{N}$:

$$\theta_k^{MLE} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} + r}{n + rK} \quad \omega_{kj}^{MLE} = \frac{\sum_{i=1}^{n} \sum_{l=1}^{d} \mathbb{1}\{x_{il} = j, y_i = k\} + r}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} + rJ}$$

- Given a test example $\mathbf{x}_* \in \{1, \ldots, J\}^d$, CNB predicts:

$$y_* = \underset{y \in \{1, \ldots, K\}}{\arg\max} \; \log \theta_y^{MLE} + \sum_{l=1}^{d} \log \omega_{y, x_{*l}}^{MLE}$$

Gaussian Discriminant Analysis

- Removes the naïve assumption from Gaussian Naive Bayes
- Uses a **generative model** $p(y, \mathbf{x} \mid \boldsymbol{\omega})$ for discrete labels $y \in \{1, \ldots, K\}$ without any conditional independence assumptions on $p(\mathbf{x}_i \mid y_i, \boldsymbol{\omega})$:

$$p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta}) = p(\mathbf{y} \mid \boldsymbol{\theta}) p(X \mid \mathbf{y}, \boldsymbol{\omega}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{\theta}) p(\mathbf{x}_i \mid y_i, \boldsymbol{\omega})$$

$$p(y_i \mid \boldsymbol{\theta}) := \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \qquad p(\mathbf{x}_i \mid y_i = k, \boldsymbol{\omega}) := \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k\right)$$

where $\boldsymbol{\omega} := \{\boldsymbol{\mu}_k, \Sigma_k\}$ and obtains these MLE estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$:

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = k\} \qquad \boldsymbol{\mu}_k^{MLE} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}$$

$$\Sigma_k^{MLE} = \frac{\sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu}_k^{MLE})(\mathbf{x}_i - \boldsymbol{\mu}_k^{MLE})^\top \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}$$

Determining the MLE Parameters

- To determine the MLE parameters for a Gaussian generative model, we need to solve the following **constrained** optimization:

$$\max_{\boldsymbol{\theta},\boldsymbol{\omega}} \log p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta}) \quad \text{subject to} \quad \sum_{k=1}^{K} \theta_k = 1$$

- $\log p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \left(\log \theta_k + \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)\right)$

- The cost function is separable and leads to three independent optimization problems:
  - $\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \theta_k$ subject to $\sum_{k=1}^{K} \theta_k = 1$
  - $\max_{\{\boldsymbol{\mu}_k\}} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$
  - $\max_{\{\Sigma_k\}} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$

Maximum Likelihood $\boldsymbol{\theta}$

- Constrained optimization wrt $\boldsymbol{\theta}$:
  - $\boldsymbol{\theta}$ is restricted to a simplex, i.e., $\boldsymbol{\theta} \in \{\mathbf{v} \in \mathbb{R}^K \mid \mathbf{1}^\top \mathbf{v} = 1\}$
  - cannot simply set the gradient of the cost function to 0
- **Handling simplex constraints**: express θ_k using a softmax function:

$$\theta_k = \frac{e^{\gamma_k}}{\sum_j e^{\gamma_j}} \qquad \frac{d\theta_k}{d\gamma_j} = \begin{cases} \theta_k(1-\theta_k), & \text{if } j = k \\ -\theta_j\theta_k, & \text{else} \end{cases}$$

- The softmax representation automatically enforces the simplex constraints and makes the optimization unconstrained!
- Now, we can just set the gradient with respect to γ_j to 0:

$$0 = \frac{d}{d\gamma_j}\sum_{i=1}^{n}\sum_{k=1}^{K}\mathbb{1}\{y_i = k\}\log\theta_k = \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{\mathbb{1}\{y_i = k\}}{\theta_k}\frac{d\theta_k}{d\gamma_j}$$

$$= \sum_{i=1}^{n}\mathbb{1}\{y_i = j\}(1-\theta_j) - \sum_{k\neq j}\mathbb{1}\{y_i = k\}\theta_j \Rightarrow \boxed{\theta_j^{MLE} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{y_i = j\}}$$

Maximum Likelihood Mean

- $$\max_{\{\boldsymbol{\mu}_k\}} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$$

- $$\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \frac{d}{d\boldsymbol{\mu}_j} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j) = 0$$

- $$\frac{d}{d\boldsymbol{\mu}} \log \phi(x; \boldsymbol{\mu}, \Sigma) = -\frac{1}{2} \frac{d}{d\boldsymbol{\mu}} (x - \boldsymbol{\mu})^\top \Sigma^{-1} (x - \boldsymbol{\mu}) = (x - \boldsymbol{\mu})^\top \Sigma^{-1}$$

- $$\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} (\mathbf{x}_i - \boldsymbol{\mu}_j)^\top \Sigma_j^{-1} = 0 \quad \Rightarrow \quad \boxed{\boldsymbol{\mu}_j^{MLE} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = j\}}}$$

Maximum Likelihood Covariance

- $$\max_{\{\Sigma_k\}} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$$

- $$\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \frac{d}{d\Sigma_j} \log \phi(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j) = 0$$

- $$\begin{aligned} \frac{d}{d\Sigma} \log \phi(x; \boldsymbol{\mu}, \Sigma) &= -\frac{1}{2} \frac{d}{d\Sigma} \log \det \Sigma - \frac{1}{2} \frac{d}{d\Sigma} (x - \boldsymbol{\mu})^\top \Sigma^{-1} (x - \boldsymbol{\mu}) \\ &= -\frac{1}{2} \Sigma^{-1} + \frac{1}{2} \Sigma^{-1} (x - \boldsymbol{\mu})(x - \boldsymbol{\mu})^\top \Sigma^{-1} \end{aligned}$$

- $$\frac{1}{2} \sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \left(\Sigma_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j^{MLE})(\mathbf{x}_i - \boldsymbol{\mu}_j^{MLE})^\top \Sigma_j^{-1} - \Sigma_j^{-1} \right) = 0$$

$$\Rightarrow \boxed{\Sigma_j^{MLE} = \frac{\sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu}_j^{MLE})(\mathbf{x}_i - \boldsymbol{\mu}_j^{MLE})^\top \mathbb{1}\{y_i = j\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = j\}}}$$

Gaussian Discriminant Analysis

- If the training set D is small, one might restrict the covariance to:
  - **diagonal**: $\Sigma_k^{MLE} = \frac{\sum_{i=1}^n \textbf{diag}(\mathbf{x}_i - \boldsymbol{\mu}_k^{MLE})^2 \mathbb{1}\{y_i = k\}}{\sum_{i=1}^n \mathbb{1}\{y_i = k\}}$
  - **spherical**: $\Sigma_k^{MLE} = \frac{\sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\mu}_k^{MLE}\|_2^2 \mathbb{1}\{y_i = k\}}{n \sum_{i=1}^n \mathbb{1}\{y_i = k\}}$
- If the training set D is large, one can obtain a more complex model by using a **Gaussian Mixture** with J components to model $p(\mathbf{x}_i \mid y_i, \boldsymbol{\omega})$:

$$p(y_i \mid \boldsymbol{\theta}) := \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \quad p(\mathbf{x}_i \mid y_i = k, \boldsymbol{\omega}) := \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_{kj}, \Sigma_{kj}\right)$$

- While an MLE estimate for $\boldsymbol{\theta}$ can be obtained as before, obtaining MLE estimates for $\boldsymbol{\omega} := \{\alpha_{kj}, \boldsymbol{\mu}_{kj}, \Sigma_{kj}\}$ is no longer straight-forward and we need to resort to the **Expectation Maximization** algorithm

Gaussian Mixture MLE via Expectation Maximization

- Start with initial guess $\boldsymbol{\omega}^{(t)} := \left\{\alpha_{kj}^{(t)}, \boldsymbol{\mu}_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right\}$ for $t = 0$, $k = 1, \ldots, K$, $j = 1, \ldots, J$ and iterate:

(E step)

$$\boxed{r_k^{(t)}(j \mid \mathbf{x}_i) = \frac{\alpha_{kj}^{(t)} \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right)}{\sum_{l=1}^{J} \alpha_{kl}^{(t)} \phi\left(\mathbf{x}_i; \boldsymbol{\mu}_{kl}^{(t)}, \Sigma_{kl}^{(t)}\right)}}$$

(M step)

$$\boxed{\alpha_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}}$$

$$\boxed{\boldsymbol{\mu}_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}}$$

$$\boxed{\Sigma_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right) \left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right)^T}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}}$$

Gaussian Mixture MLE via Expectation Maximization

- Sometimes the data is not enough to estimate all these parameters:
  - Fix the weights $\alpha_{kj} = \frac{1}{J}$
  - Fix diagonal $\Sigma_{kj} = \textbf{diag}\left([\sigma^2_{kj1}, \ldots, \sigma^2_{kjn}]^T\right)$ or spherical $\Sigma_{kj} = \sigma^2_{kj} I_n$
  - Estimate a **diagonal covariance**:

$$\Sigma_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \textbf{diag}\left(\mathbf{x}_i - \boldsymbol{\mu}_{kj}^{(t+1)}\right)^2}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}$$

  - Estimate a **spherical covariance**:

$$\sigma_{kj}^{2,(t+1)} = \frac{1}{d} \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \left\|\mathbf{x}_i - \boldsymbol{\mu}_{kj}^{(t+1)}\right\|^2}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}, \qquad \mathbf{x}_i \in \mathbb{R}^d$$

- How should we initialize $\boldsymbol{\omega}^{(0)}$? Use k-**means++**!

- If $\sigma_{kj} \to 0$, the GM component assignments of EM become hard and EM works like k-means.