

ECE276A: Sensing & Estimation in Robotics

Lecture 4-5: Supervised Learning

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qif007@eng.ucsd.edu

Arash Asgharivaskasi: aasghari@eng.ucsd.edu

Thai Duong: tduong@eng.ucsd.edu

Yiran Xu: y5xu@eng.ucsd.edu

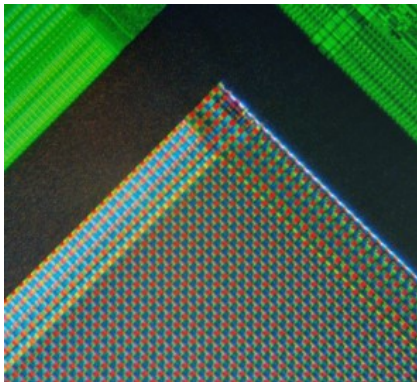
UC San Diego

JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Color Imaging

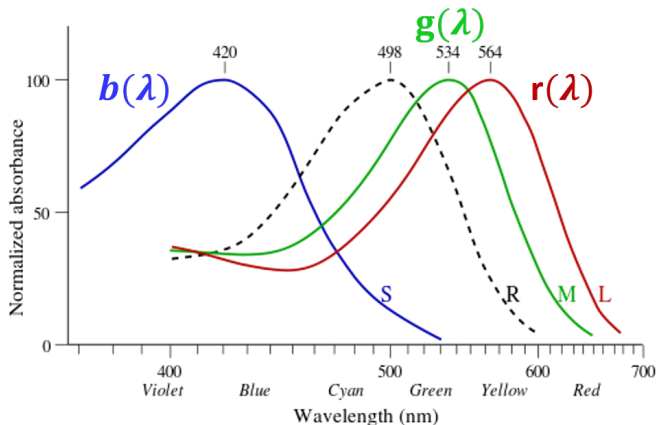
- ▶ Image sensor: converts light into small bursts of current
- ▶ Analog imaging technology uses charge-coupled devices (CCD) or complementary metal-oxide semiconductors (CMOS)
- ▶ CCD/CMOS photosensor array:
 - ▶ A phototransistor converts light into current
 - ▶ Each transistor charges a capacitor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

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



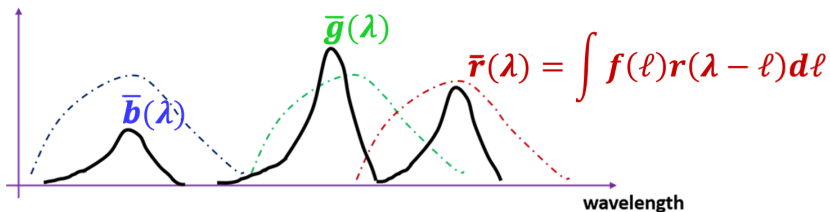
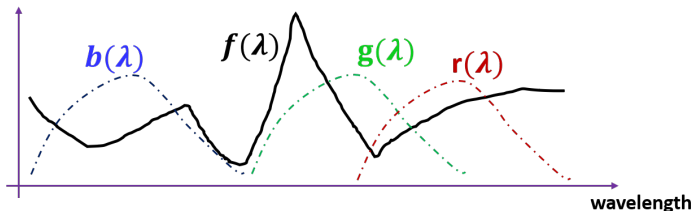
Why RGB, Why 3?

- ▶ **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:**

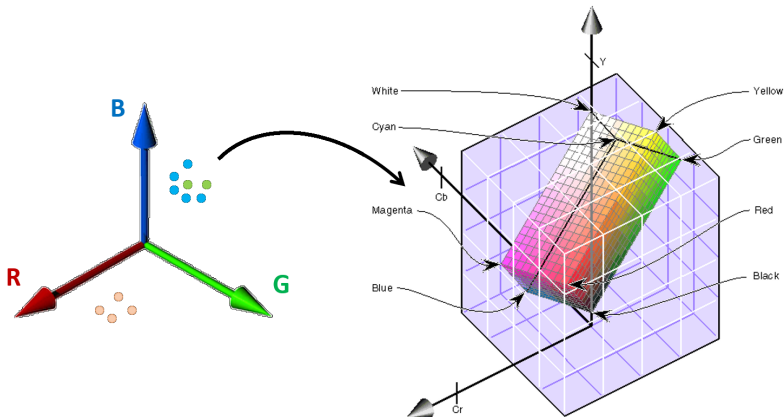
- Given an arbitrary light spectral distribution $f(\lambda)$, the cone cells act as filters that provide a **convolution-like signal** to the brain:



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

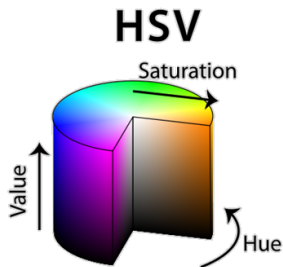
Luma-Chroma Color Space

- ▶ **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 $\approx 0^\circ$, green $\approx 120^\circ$, blue $\approx 240^\circ$)
 - ▶ **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 values from 2-D image space to 3-D world space
 - ▶ Extracting and tracking image point features
 - ▶ Estimating the 3-D world frame positions of image point features
 - ▶ Estimating the position and orientation of a camera observing image point features

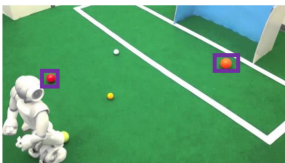
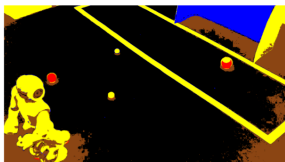
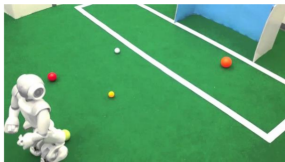
Color Segmentation as a Classification Problem

- ▶ **Color Segmentation Problem:** requires segmenting the 3-D color space into a set of discrete volumes associated with semantic colors:
 - ▶ Each pixel is a **3-D vector**: $\mathbf{x} = (R, G, B)$ or (Y, Cb, Cr) or (H, S, V)
 - ▶ Discrete color labels: $y \in \{1, \dots, N\}$, e.g., {red, blue, green, yellow, ...}
- ▶ Pixel values are noisy
- ▶ We will use a **probabilistic model** $p(y | \mathbf{x})$ for the color class y of a given pixel \mathbf{x}
- ▶ **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 probabilistic model $p(y | \mathbf{x})$
- ▶ **Testing:** use the optimized model $p(y | \mathbf{x})$ to define a function that transforms a new color-space input \mathbf{x}_* into a discrete color label y_* :

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

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 Recognition

- ▶ Given a set of orange pixels with coordinates $\{(u_p, v_p)\}_{p=1}^{N_p}$

- ▶ Center of mass:

$$(c_u, c_v) = \left(\frac{1}{N_p} \sum_p u_p, \frac{1}{N_p} \sum_p v_p \right)$$

- ▶ Fit an ellipse:

$$E_{uu} = \frac{1}{N_p} \sum_p (u_p - c_u)^2$$

$$E_{uv} = \frac{1}{N_p} \sum_p (u_p - c_u)(v_p - c_v)$$

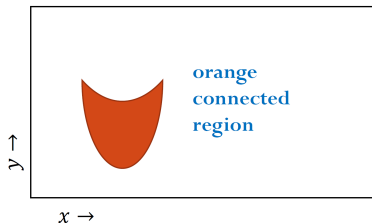
$$E_{vv} = \frac{1}{N_p} \sum_p (v_p - c_v)^2$$

- ▶ Recognize a spherical ball based on thresholds ϵ_0, ϵ_1 on the eigenvalues

$$\lambda_0, \lambda_1 \text{ of } \begin{bmatrix} E_{uu} & E_{uv} \\ E_{uv} & E_{vv} \end{bmatrix}$$

- ▶ size: $\min \lambda_1, \lambda_2 \geq \epsilon_0$
- ▶ eccentricity: $1 - \epsilon_1 \leq \frac{\lambda_1}{\lambda_2} \leq 1 + \epsilon_1$

Color image:



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 stop sign based on the color segmentation and the known stop-sign shape



Project 1 Tips

- ▶ Define K color classes that you want to distinguish, e.g., stop-sign-red, other-red, brown, orange
- ▶ Label examples for each color class to obtain a training dataset $D = \{\mathbf{x}_i, y_i\}$ by selecting pixel regions (use **roipoly**) that correspond to the desired color
- ▶ Train a discriminative $p(y | \mathbf{x})$ (Logistic Regression) or $p(y, \mathbf{x})$ generative (Gaussian or Gaussian Mixture) model
- ▶ Given a test image, classify each pixel into one of the K color classes using your model
- ▶ Find stop-sign-red regions (use OpenCV's **findContours**)
- ▶ Enumerate red region combinations and score them based on how much they resemble a stop sign shape (use Scikit-Image's **regionprops**)
- ▶ Experiment with different colorspace 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: $h : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 $y \in \{-1, 1\}$):

$$\min_h \text{Loss}_{0-1}(h) := \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{h(\mathbf{x}_i) \neq y_i} = \# \text{ of times } h \text{ is wrong about the labels}$$

- ▶ **Regression** (continuous $y \in \mathbb{R}$):

$$\min_h \text{MSE}(h) := \frac{1}{n} \sum_{i=1}^n (h(\mathbf{x}_i) - y_i)^2 = \text{mean square error of } h$$

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 associated labels y by sampling from $p(y, \mathbf{x})$
- ▶ **Examples:** Naive Bayes, Mixture Models, 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 $\boldsymbol{\omega}$
- ▶ Estimate/optimize/learn $\boldsymbol{\omega}$ based on the training set $D = (X, \mathbf{y})$ in a way that $\boldsymbol{\omega}^*$ produces 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	$\omega_{MLE} := \arg \max_{\omega} p(\mathbf{y} X, \omega)$	$\omega_{MLE} := \arg \max_{\omega} p(\mathbf{y}, X \omega)$
Testing	$y_* \in \arg \max_y p(y \mathbf{x}_*, \omega_{MLE})$	$y_* \in \arg \max_y p(y, \mathbf{x}_* \omega_{MLE})$

▶ Maximum A Posteriori (MAP):

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

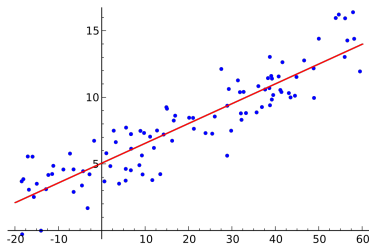
▶ Bayesian Inference:

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

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 $D = \{(x_i, y_i)\}$, 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:

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

Discriminative Regression via a Linear Gaussian Model

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

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

- ▶ Use MLE to estimate the parameters:

$$\omega_{MLE} := \arg \max_{\omega} p(\mathbf{y}|X, \omega) = \arg \max_{\omega} \log p(\mathbf{y}|X, \omega)$$

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

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

Discriminative Regression via a Linear Gaussian Model

- ▶ MLE using the data log-likelihood we derived:

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

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

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

- ▶ and solve for ω :

$$\omega_{MLE} = (X^{\top} V^{-1} X)^{-1} X^{\top} V^{-1} \mathbf{y}$$

Discriminative Regression via a Linear Gaussian Model

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

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

- ▶ The MAP estimate of ω is:

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

- ▶ The optimization is equivalent to the MLE setting but includes (Tikhonov) regularization on ω

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

- ▶ MAP: $\boldsymbol{\omega}_{MAP} = (X^\top V^{-1}X + \Lambda^{-1})^{-1}X^\top V^{-1}\mathbf{y}$

- ▶ **Testing:** given a test example $\mathbf{x}_* \in \mathbb{R}^d$, use the optimized parameters $\boldsymbol{\omega}^*$ to predict the label:

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

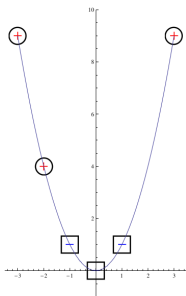
- ▶ The test expression is obtained from the gradient of the log-likelihood with respect to y :

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

Linear Regression Example

- ▶ Consider the following dataset:

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



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

$$X' \omega_1 + \omega_0 \mathbf{1} = \underbrace{[X' \quad \mathbf{1}]}_X \underbrace{\begin{bmatrix} \omega_1 \\ \omega_0 \end{bmatrix}}_{\omega}$$

- ▶ Let the discriminative model be:

$$\begin{aligned} p(\mathbf{y} | X, \omega) &= \phi(\mathbf{y}; X\omega, V) \\ p(\omega | X) &= \phi(\omega; \mathbf{0}, \Lambda) \end{aligned}$$

$$V = I_n$$

$$\Lambda = 2I_d$$

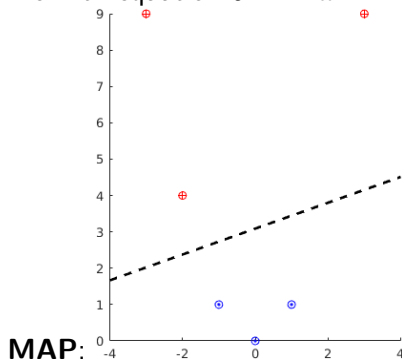
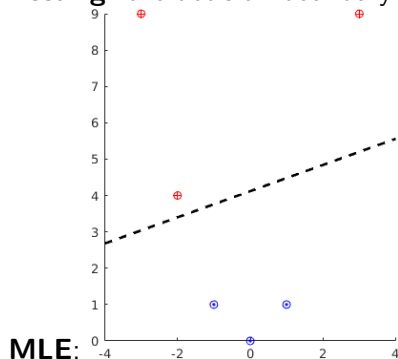
Linear Regression Example

▶ Training:

▶ MLE: $\omega_{MLE} = (X^T V^{-1} X)^{-1} X^T V^{-1} \mathbf{y} = \begin{bmatrix} -0.0857 \\ 0.2381 \\ -0.9810 \end{bmatrix}$

▶ MAP: $\omega_{MAP} = (X^T V^{-1} X + \Lambda^{-1})^{-1} X^T V^{-1} \mathbf{y} = \begin{bmatrix} -0.0643 \\ 0.1806 \\ -0.5580 \end{bmatrix}$

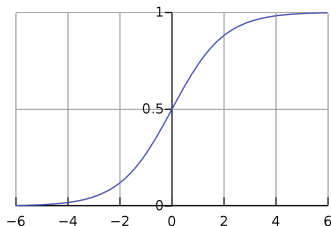
▶ Testing: the decision boundary is a line with equation $0 = \mathbf{x}^T \omega$:



Logits

- ▶ The following functions are useful for converting continuous (regression) estimates into discrete distributions for the purpose of **classification**
- ▶ **sigmoid function**: converts continuous preferences $z \in \mathbb{R}$ into a Bernoulli distribution over two classes:

$$\begin{aligned}\sigma(z) &:= \frac{1}{1 + \exp(-z)} = \frac{\exp(z)}{\exp(z) + \exp(0)} \\ &= 1 - \sigma(-z) = \frac{\sigma'(z)}{(1 - \sigma(z))}\end{aligned}$$



- ▶ **softmax function**: converts continuous preferences $z \in \mathbb{R}^K$ into a categorical distribution over K classes:

$$\text{softmax}(z) := \left[\frac{\exp(z_1)}{\sum_j \exp(z_j)} \quad \cdots \quad \frac{\exp(z_K)}{\sum_j \exp(z_j)} \right] = \underbrace{\text{softmax}(z - \max_i z_i)}_{\text{useful for numerical conditioning}}$$

Discriminative Classification via a Logistic Model

- ▶ Given an example $\mathbf{x} \in \mathbb{R}^d$ and binary label $y \in \{-1, 1\}$, use a sigmoid function to map the continuous value $\boldsymbol{\omega}^\top \mathbf{x}$ to a discrete (Bernoulli) distribution for y :

$$p(y = 1 | \mathbf{x}, \boldsymbol{\omega}) = \sigma(\boldsymbol{\omega}^\top \mathbf{x})$$

- ▶ For $p(y | \mathbf{x}, \boldsymbol{\omega})$ to be a valid pmf, it needs to sum to 1 over $\{-1, 1\}$:

$$p(y = -1 | \mathbf{x}, \boldsymbol{\omega}) = 1 - p(y = 1 | \mathbf{x}, \boldsymbol{\omega}) = 1 - \sigma(\boldsymbol{\omega}^\top \mathbf{x}) \stackrel{\substack{\text{sigmoid} \\ \text{properties}}}{=} \sigma(-\boldsymbol{\omega}^\top \mathbf{x})$$

- ▶ Combine the two cases to obtain the final model for the distribution of y in terms of the example \mathbf{x} and the parameters $\boldsymbol{\omega}$:

$$p(y | \mathbf{x}, \boldsymbol{\omega}) = \sigma(y \mathbf{x}^\top \boldsymbol{\omega})$$

Discriminative Classification via a Logistic Model

- ▶ **Logistic regression**: approximates the unknown label-generating pdf with a sigmoid function:

$$p(y|\mathbf{x}, \boldsymbol{\omega}) = \sigma(y\mathbf{x}^\top \boldsymbol{\omega})$$

- ▶ Since the data (X, \mathbf{y}) is **iid**, the joint data likelihood is:

$$p(\mathbf{y}|X, \boldsymbol{\omega}) = \prod_{i=1}^n \sigma(y_i \mathbf{x}_i^\top \boldsymbol{\omega}) = \prod_{i=1}^n \frac{1}{1 + \exp(-y_i \mathbf{x}_i^\top \boldsymbol{\omega})}$$

- ▶ Leads to these MLE and MAP (with $\boldsymbol{\omega} \sim \mathcal{N}(0, \Lambda)$) estimates for $\boldsymbol{\omega}$:

$$\boldsymbol{\omega}_{MLE} = \arg \max_{\boldsymbol{\omega}} \log p(\mathbf{y} | X, \boldsymbol{\omega}) = \arg \min_{\boldsymbol{\omega}} \sum_{i=1}^n \log \left(1 + \exp(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}) \right)$$

$$\boldsymbol{\omega}_{MAP} = \arg \max_{\boldsymbol{\omega}} \log p(\mathbf{y} | X, \boldsymbol{\omega}) + \log p(\boldsymbol{\omega})$$

$$= \arg \min_{\boldsymbol{\omega}} \sum_{i=1}^n \log \left(1 + \exp(-y_i \mathbf{x}_i^\top \boldsymbol{\omega}) \right) + \frac{1}{2} \boldsymbol{\omega}^\top \Lambda^{-1} \boldsymbol{\omega}$$

Discriminative Classification via a Logistic Model

- ▶ $\nabla_{\omega} (-\log p(\mathbf{y} | X, \omega)) = 0$ does not have a closed-form solution :-(
 - ▶ The negative log-likelihood $-\log p(\mathbf{y} | X, \omega)$ is **convex** in ω :
 - ▶ The composition of an affine function $f_i(\omega) := -y_i \mathbf{x}_i^\top \omega$ and a convex function $g(z) := \log(1 + \exp^z)$ is convex
 - ▶ The sum of convex functions $\sum_{i=1}^n g(f_i(\omega))$ is convex
- ▶ The negative log-likelihood can be minimized iteratively to obtain a **global** minimum:

$$\begin{aligned}\omega_{MLE}^{(t+1)} &= \omega_{MLE}^{(t)} - \alpha \nabla_{\omega} (-\log p(\mathbf{y}|X, \omega)) \Big|_{\omega=\omega_{MLE}^{(t)}} \\ &= \omega_{MLE}^{(t)} - \alpha \sum_{i=1}^n \frac{1}{1 + \exp(-y_i \mathbf{x}_i^\top \omega_{MLE}^{(t)})} \exp(-y_i \mathbf{x}_i^\top \omega_{MLE}^{(t)}) (-y_i \mathbf{x}_i) \\ &= \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^n y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^\top \omega_{MLE}^{(t)}))\end{aligned}$$

Logistic Regression Summary

- ▶ **Model:** discriminative model $p(\mathbf{y}|X, \omega)$ for discrete labels $\mathbf{y} \in \{-1, 1\}^n$:

$$p(\mathbf{y}|X, \omega) = \prod_{i=1}^n \sigma(y_i \mathbf{x}_i^\top \omega) = \prod_{i=1}^n \frac{1}{1 + \exp(-y_i \mathbf{x}_i^\top \omega)}$$

- ▶ **Training:** given data $D = (X, \mathbf{y})$, optimize the model parameters:

- ▶ MLE: $\omega_{MLE}^{(t+1)} = \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^n y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^\top \omega_{MLE}^{(t)}))$

- ▶ MAP: $\omega_{MAP}^{(t+1)} = \omega_{MAP}^{(t)} + \alpha \left(\sum_{i=1}^n y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^\top \omega_{MAP}^{(t)})) - \Lambda^{-1} \omega_{MAP}^{(t)} \right)$

- ▶ **Testing:** given a test example $\mathbf{x}_* \in \mathbb{R}^d$, use the optimized parameters ω^* to predict the label:

$$y_* = \begin{cases} 1 & \mathbf{x}_*^\top \omega^* \geq 0 \\ -1 & \mathbf{x}_*^\top \omega^* < 0 \end{cases}$$

- ▶ Logistic regression generates a **linear decision boundary**, i.e., the values for \mathbf{x}_* for which the predictions $y_* = 1$ and $y_* = -1$ are equally likely is:

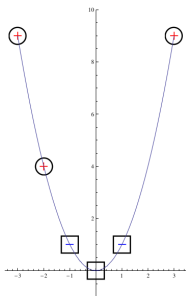
$$1 = \frac{p(y = 1 | \mathbf{x}_*, \omega^*)}{p(y = -1 | \mathbf{x}_*, \omega^*)} = \frac{1 + \exp(\mathbf{x}_*^\top \omega^*)}{1 + \exp(-\mathbf{x}_*^\top \omega^*)} \Leftrightarrow \mathbf{x}_*^\top \omega^* = 0$$

Logistic Regression Example

- Consider the same data as before:

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

$$\mathbf{y} = \begin{bmatrix} +1 \\ +1 \\ -1 \\ -1 \\ -1 \\ +1 \end{bmatrix} \in \mathbb{R}^n$$

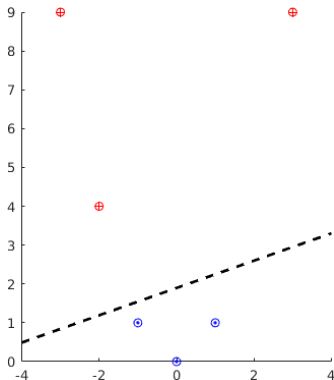


Logistic Regression Example

- ▶ **Training:** start with $\omega_{MLE}^{(0)} = \mathbf{0} \in \mathbb{R}^3$ and iterate:

$$\omega_{MLE}^{(t+1)} = \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^n y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^\top \omega_{MLE}^{(t)}))$$

- ▶ After 10 iterations with $\alpha = 0.1$, we have: $\omega_{MLE}^{(10)} = \begin{bmatrix} 0.2115 \\ -0.6015 \\ 1.1408 \end{bmatrix}$
- ▶ **Testing:** the decision boundary is a line with equation $0 = \mathbf{x}^\top \omega$:



K-ary Logistic Regression

- ▶ **Logistic regression with K -classes** ($\mathbf{y} \in \{1, \dots, K\}^n$) uses a softmax model with parameters $W \in \mathbb{R}^{K \times d}$:

$$p(\mathbf{y}|X, W) = \prod_{i=1}^n \mathbf{e}_{y_i}^\top \mathbf{s}(W\mathbf{x}_i) := \prod_{i=1}^n \mathbf{e}_{y_i}^\top \frac{\exp(W\mathbf{x}_i)}{\mathbf{1}^\top \exp(W\mathbf{x}_i)}$$

where \mathbf{e}_j is the j -th standard basis vector and $\mathbf{s}(\mathbf{z})$ is the **softmax** function:

$$\mathbf{s}(\mathbf{z}) := \frac{\mathbf{e}^{\mathbf{z}}}{\mathbf{1}^\top \mathbf{e}^{\mathbf{z}}} \in \mathbb{R}^K \quad \frac{ds_i}{dz_j} = \begin{cases} s_i(1 - s_i) & \text{if } i = j \\ -s_i s_j & \text{else} \end{cases}$$

- ▶ To optimize the parameters $W \in \mathbb{R}^{K \times d}$ via MLE, we need to compute the gradient of the data log-likelihood:

$$\begin{aligned} W_{MLE}^{(t+1)} &= W_{MLE}^{(t)} + \alpha \left(\nabla_W [\log p(\mathbf{y} | X, W)] \Big|_{W=W_{MLE}^{(t)}} \right) \\ &= W_{MLE}^{(t)} + \alpha \left(\sum_{i=1}^n \left(\mathbf{e}_{y_i} - \mathbf{s}(W_{MLE}^{(t)}\mathbf{x}_i) \right) \mathbf{x}_i^\top \right) \end{aligned}$$

K-ary Logistic Regression MLE

- ▶ Taking the gradient of $\log p(\mathbf{y} | X, W)$ with respect to W gives:

$$\nabla_W \left[\sum_{i=1}^n \log \mathbf{e}_{y_i}^\top \mathbf{s}(W\mathbf{x}_i) \right] = \sum_{i=1}^n \frac{1}{\mathbf{e}_{y_i}^\top \mathbf{s}(W\mathbf{x}_i)} \nabla_W \left[\mathbf{e}_{y_i}^\top \mathbf{s}(W\mathbf{x}_i) \right]$$

- ▶ Let $\mathbf{z}_i := W\mathbf{x}_i$. The derivative of the last term via chain rule is:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{z}_i} \mathbf{e}_{y_i}^\top \mathbf{s}(\mathbf{z}_i) \frac{\partial}{\partial W} W\mathbf{x}_i &= \frac{\partial}{\partial W} \mathbf{e}_{y_i}^\top \frac{\partial \mathbf{s}(\mathbf{z}_i)}{\partial \mathbf{z}_i} W\mathbf{x}_i = \frac{\partial}{\partial W} \text{tr} \left(\mathbf{x}_i \mathbf{e}_{y_i}^\top \frac{\partial \mathbf{s}(\mathbf{z}_i)}{\partial \mathbf{z}_i} W \right) \\ &= \mathbf{x}_i \mathbf{e}_{y_i}^\top \frac{\partial \mathbf{s}(\mathbf{z}_i)}{\partial \mathbf{z}_i} = \mathbf{x}_i \mathbf{e}_{y_i}^\top \frac{\partial \mathbf{s}}{\partial \mathbf{z}_i} (W\mathbf{x}_i) \end{aligned}$$

- ▶ Arrange the derivative of softmax in matrix form:

$$\frac{d\mathbf{s}(\mathbf{z})}{d\mathbf{z}} = \begin{bmatrix} s_1 - s_1^2 & -s_1 s_2 & \cdots & -s_1 s_d \\ -s_2 s_1 & s_2 - s_2^2 & \cdots & -s_2 s_d \\ \vdots & & \ddots & \vdots \\ -s_d s_1 & -s_d s_2 & \cdots & -s_d s_d \end{bmatrix} = \mathbf{diag}(\mathbf{s}(\mathbf{z})) - \mathbf{s}(\mathbf{z})\mathbf{s}(\mathbf{z})^\top$$

K-ary Logistic Regression MLE

- ▶ The gradient is the transpose of the derivative:

$$\begin{aligned}\nabla_W [\log p(\mathbf{y} \mid X, W)] &= \sum_{i=1}^n \frac{1}{\mathbf{e}_{y_i}^\top \mathbf{s}(W\mathbf{x}_i)} (\mathbf{diag}(\mathbf{s}(W\mathbf{x}_i)) - \mathbf{s}(W\mathbf{x}_i)\mathbf{s}(W\mathbf{x}_i)^\top) \mathbf{e}_{y_i} \mathbf{x}_i^\top \\ &= \sum_{i=1}^n (\mathbf{e}_{y_i} - \mathbf{s}(W\mathbf{x}_i)) \mathbf{x}_i^\top \in \mathbb{R}^{K \times d}\end{aligned}$$

- ▶ MLE leads to gradient ascent of the form:

$$W_{MLE}^{(t+1)} = W_{MLE}^{(t)} + \alpha \left(\sum_{i=1}^n (\mathbf{e}_{y_i} - \mathbf{s}(W_{MLE}^{(t)}\mathbf{x}_i)) \mathbf{x}_i^\top \right)$$

- ▶ MAP with $W \sim \mathcal{N}(0, \lambda I_{Kd \times Kd})$ leads to gradient ascent of the form:

$$W_{MAP}^{(t+1)} = W_{MAP}^{(t)} + \alpha \left(\sum_{i=1}^n (\mathbf{e}_{y_i} - \mathbf{s}(W_{MAP}^{(t)}\mathbf{x}_i)) \mathbf{x}_i^\top \right) - \alpha \lambda^{-1} W_{MAP}^{(t)}$$

Generative Classification via a Naive Bayes Model

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

$$p(y, \mathbf{x} \mid \omega, \theta) = p(y \mid \theta)p(\mathbf{x} \mid y, \omega) = p(y \mid \theta) \prod_{l=1}^d p(x_l \mid y, \omega)$$

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

Gaussian Naive Bayes

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

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

- ▶ GNB obtains the following MLE estimates of θ and ω :

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

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

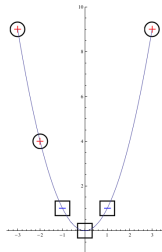
- ▶ Given a test example $\mathbf{x}_* \in \mathbb{R}^d$, the GNB classifier produces the output:

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

Gaussian Naive Bayes Example

- ▶ Consider the same data as before:

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



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

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

$$\mu_{kl}^{MLE} = \frac{\sum_{i=1}^n x_{il} \mathbb{1}\{y_i = k\}}{\sum_{i=1}^n \mathbb{1}\{y_i = k\}} = \begin{array}{|c|c|c|} \hline & l=1 & l=2 \\ \hline k=1 & -0.66 & 7.33 \\ \hline k=-1 & 0.00 & 0.66 \\ \hline \end{array}$$

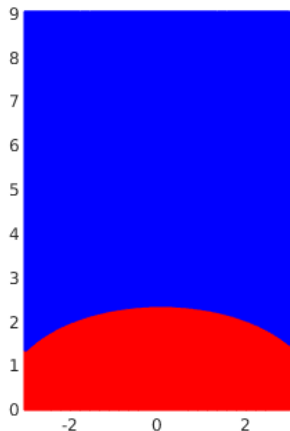
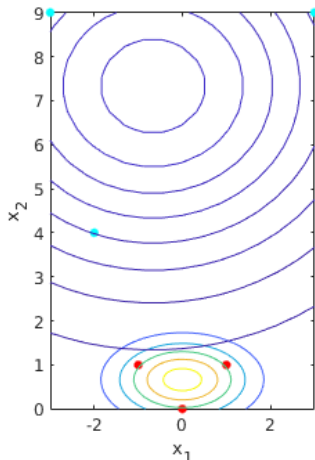
$$\sigma_{kl}^{MLE} = \sqrt{\frac{\sum_{i=1}^n (x_{il} - \mu_{kl}^{MLE})^2 \mathbb{1}\{y_i = k\}}{\sum_{i=1}^n \mathbb{1}\{y_i = k\}}} = \begin{array}{|c|c|c|} \hline & l=1 & l=2 \\ \hline k=1 & 2.62 & 2.36 \\ \hline k=-1 & 1.05 & 6.68 \\ \hline \end{array}$$

Gaussian Naive 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):



Logistic Regression vs Gaussian Naive Bayes

- ▶ The decision boundary of Gaussian Naive Bayes is a **quadratic** function. It looks like an ellipse, parabola, or hyperbola in 2-D.
- ▶ When the variance is shared among the classes:

$$\sigma_{kl}^{MLE} \equiv \sigma_l^{MLE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{il} - \mu_{y_i, l}^{MLE})^2} \quad \text{for } k = 1, \dots, K$$

the decision boundary of Gaussian Naive Bayes is **linear**.

- ▶ Logistic regression always generates a **linear decision boundary**:
 $0 = \mathbf{x}^\top \boldsymbol{\omega}$
- ▶ Logistic regression has **lower bias** but **higher variance** than Gaussian Naive Bayes.

Categorical Naive Bayes

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

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

- ▶ CNB obtains these MLE estimates of θ and ω 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 $\mathbf{x}_* \in \{1, \dots, J\}^d$, CNB predicts:

$$y_* = \arg \max_{y \in \{1, \dots, K\}} \log \theta_y^{MLE} + \sum_{l=1}^d \log \omega_{y, x_{*l}}^{MLE}$$

Gaussian Discriminant Analysis

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

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

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

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

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

Determining the MLE Parameters

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

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

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

- ▶ $\max_{\theta} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{y_i = k\} \log \theta_k \quad \text{subject to} \quad \sum_{k=1}^K \theta_k = 1$

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

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

Maximum Likelihood θ

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

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

- ▶ 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:

$$\begin{aligned} 0 &= \frac{d}{d\gamma_j} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{y_i = k\} \log \theta_k = \sum_{i=1}^n \sum_{k=1}^K \frac{\mathbb{1}\{y_i = k\}}{\theta_k} \frac{d\theta_k}{d\gamma_j} \\ &= \sum_{i=1}^n \mathbb{1}\{y_i = j\}(1 - \theta_j) - \sum_{k \neq j} \mathbb{1}\{y_i = k\}\theta_j \Rightarrow \theta_j^{MLE} = \frac{1}{n} \sum_{i=1}^n \mathbb{1}\{y_i = j\} \end{aligned}$$

Maximum Likelihood Mean

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

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

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

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

Maximum Likelihood Covariance

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

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

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

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

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

Gaussian Discriminant Analysis

- ▶ If the training set D is small, one might restrict the covariance to:
 - ▶ **diagonal:** $\Sigma_k^{MLE} = \frac{\sum_{i=1}^n \text{diag}(\mathbf{x}_i - \boldsymbol{\mu}_k^{MLE})^2 \mathbb{1}\{y_i=k\}}{\sum_{i=1}^n \mathbb{1}\{y_i=k\}}$
 - ▶ **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 $p(\mathbf{x}_i | y_i, \boldsymbol{\omega})$:

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

- ▶ 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 Mixtures

Gaussian Mixtures

- ▶ Gaussian Mixtures are well suited for modeling clusters of points:
 - ▶ each cluster is assigned a Gaussian
 - ▶ the mean is somewhere in the middle of the cluster
 - ▶ the covariance measures the cluster spread
- ▶ **Sampling** from a Gaussian Mixture:
 - ▶ Draw an integer between 1 and J with probability α_{kj}
 - ▶ Draw a vector \mathbf{x} from the j -th Gaussian pdf $\phi(\mathbf{x}; \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj})$
- ▶ It is useful to understand the meaning of $q_k(j, \mathbf{x}) := \alpha_{kj} \phi(\mathbf{x}; \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj})$
- ▶ Given class k , $q_k(j, \mathbf{x}) d\mathbf{x}$ is the joint probability of drawing component j and data point \mathbf{x} in a volume $d\mathbf{x}$ around it
- ▶ The **membership probability** of data point \mathbf{x} is the conditional probability of having selected component j given \mathbf{x} :

$$r_k(j | \mathbf{x}) := \frac{q_k(j, \mathbf{x})}{\sum_{l=1}^J q_k(l, \mathbf{x})} \qquad \sum_{j=1}^J r_k(j | \mathbf{x}) = 1$$

Gaussian Mixture Discriminant Analysis

- ▶ A generative model that uses a **Gaussian Mixture** with J components to model $p(\mathbf{x}_i | y_i, \omega)$:

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

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

- ▶ Training via MLE: $\max_{\theta, \omega} p(\mathbf{y}, X | \theta, \omega)$
 - ▶ The MLE of θ can be obtained via the softmax trick and differentiation as we saw for the single-Gaussian discriminant analysis
 - ▶ Obtaining MLE estimates for $\omega := \{\alpha_{kj}, \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj}\}$ is no longer easy because $\log \sum_{j=1}^J \alpha_{kj} \phi(\mathbf{x}_i; \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj})$ is not convex/concave
 - ▶ Also, need to ensure that $\sum_{j=1}^J \alpha_{kj} = 1$ for all k .

Data Log Likelihood

- ▶ $\log p(\mathbf{y}, X \mid \boldsymbol{\omega}, \boldsymbol{\theta}) = \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{y_i = k\} \log \theta_k$
 $+ \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{y_i = k\} \log \left(\sum_{j=1}^J \alpha_{kj} \phi(\mathbf{x}_i; \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj}) \right)$
- ▶ Focus on max wrt $\boldsymbol{\omega} := \{\alpha_{kj}, \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj}\}$; the first term can be ignored
- ▶ To simplify notation, let $D_k := \{(\mathbf{x}_i, y_i) \mid y_i = k\} \subseteq D$ and define:

$$\lambda(X, \boldsymbol{\omega}) := \sum_{k=1}^K \sum_{\mathbf{x} \in D_k} \log \left(\sum_{j=1}^J \alpha_{kj} \phi(\mathbf{x}; \boldsymbol{\mu}_{kj}, \boldsymbol{\Sigma}_{kj}) \right)$$

Local maxima of $\lambda(X, \omega)$

▶ Maxima of $\sum_{k=1}^K \sum_{\mathbf{x} \in D_k} \log \left(\sum_{j=1}^J \alpha_{kj} \phi(\mathbf{x}; \boldsymbol{\mu}_{kj}, \Sigma_{kj}) \right)$ occur at critical points

$$\begin{aligned} \text{▶ } \frac{d}{d\boldsymbol{\mu}_{lm}} \lambda(X, \omega) &= \sum_{\mathbf{x} \in D_l} \frac{\alpha_{lm}}{\sum_{j=1}^J \alpha_{lj} \phi(\mathbf{x}; \boldsymbol{\mu}_{lj}, \Sigma_{lj})} \frac{d}{d\boldsymbol{\mu}_{lm}} \phi(\mathbf{x}; \boldsymbol{\mu}_{lm}, \Sigma_{lm}) \\ &= \sum_{\mathbf{x} \in D_l} r_l(m | \mathbf{x}) (\boldsymbol{\mu}_{lm} - \mathbf{x})^\top \Sigma_{lm}^{-1} \end{aligned}$$

$$\text{▶ } \frac{d}{d\Sigma_{lm}} \lambda(X, \omega) = \frac{1}{2} \sum_{\mathbf{x} \in D_l} r_l(m | \mathbf{x}) \left(\Sigma_{lm}^{-1} (\boldsymbol{\mu}_{lm} - \mathbf{x}) (\boldsymbol{\mu}_{lm} - \mathbf{x})^\top \Sigma_{lm}^{-1} - \Sigma_{lm}^{-1} \right)$$

▶ Use the **softmax trick** for α_{kj} to handle simplex constraints:

$$\begin{aligned} \frac{d}{d\gamma_{lm}} \lambda(X, \omega) &= \sum_{\mathbf{x} \in D_l} \frac{1}{\sum_{j=1}^J \alpha_{lj} \phi(\mathbf{x}; \boldsymbol{\mu}_{lj}, \Sigma_{lj})} \sum_{j=1}^J \frac{d\alpha_{lj}}{d\gamma_{lm}} \phi(\mathbf{x}; \boldsymbol{\mu}_{lj}, \Sigma_{lj}) \\ &= \sum_{\mathbf{x} \in D_l} (r_l(m | \mathbf{x}) - \alpha_{lm}) \end{aligned}$$

Local maxima of $\lambda(\mathbf{X}, \boldsymbol{\omega})$

- ▶ Setting the previous derivatives to zero, we obtain:

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

$$\mu_{kj} = \frac{\sum_{i=1}^n \mathbb{1}\{y_i = k\} r_k(j | \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n \mathbb{1}\{y_i = k\} r_k(j | \mathbf{x}_i)}$$

$$\Sigma_{kj} = \frac{\sum_{i=1}^n \mathbb{1}\{y_i = k\} r_k(j | \mathbf{x}_i) (\mathbf{x}_i - \mu_{kj})(\mathbf{x}_i - \mu_{kj})^T}{\sum_{i=1}^n \mathbb{1}\{y_i = k\} r_k(j | \mathbf{x}_i)}$$

- ▶ The mixture weights are equal to the sample mean of the membership probabilities $r_k(j | \mathbf{x}_i)$ assuming a uniform distribution over D_k
- ▶ The latter are the sample mean and covariance of the data, weighted by the membership probabilities
- ▶ The three equations are coupled through $r_k(j | \mathbf{x})$, which depends on $\boldsymbol{\omega} := \{\alpha_{kj}, \boldsymbol{\mu}_{kj}, \Sigma_{kj}\}$, and hence are hard to solve directly
- ▶ **Optimization Idea:**
 - ▶ start with a guess $\boldsymbol{\omega}^{(0)}$ and use a descent method
 - ▶ iterate between updating $r_k(j | \mathbf{x}_i)$ and updating $\boldsymbol{\omega}^{(t)}$

Gaussian Mixture MLE via Expectation Maximization

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

$$\text{(E step)} \quad r_k^{(t)}(j | \mathbf{x}_i) = \frac{\alpha_{kj}^{(t)} \phi(\mathbf{x}_i; \mu_{kj}^{(t)}, \Sigma_{kj}^{(t)})}{\sum_{l=1}^J \alpha_{kl}^{(t)} \phi(\mathbf{x}_i; \mu_{kl}^{(t)}, \Sigma_{kl}^{(t)})}$$

$$\text{(M step)} \quad \alpha_{kj}^{(t+1)} = \frac{\sum_{i=1}^n \mathbb{1}\{y_i = k\} r_k^{(t)}(j | \mathbf{x}_i)}{\sum_{i=1}^n \mathbb{1}\{y_i = k\}}$$

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

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

Gaussian Mixture MLE via EM (comments)

- ▶ Sometimes the data is not enough to estimate all these parameters:
 - ▶ Fix the weights $\alpha_{kj} = \frac{1}{J}$
 - ▶ Fix diagonal $\Sigma_{kj} = \mathbf{diag}([\sigma_{kj1}^2, \dots, \sigma_{kjd}^2]^T)$ or spherical $\Sigma_{kj} = \sigma_{kj}^2 I_n$
 - ▶ Estimate a **diagonal covariance**:

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

- ▶ Estimate a **spherical covariance**:

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

- ▶ How should we initialize $\omega^{(0)}$? Use **k-means++**!
- ▶ If $\sigma_{kj} \rightarrow 0$, the GM component assignments of EM become hard and EM works like **k-means**.