ECE276A: Sensing & Estimation in Robotics
Lecture 3: Supervised Learning

Lecturer:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Siwei Guo: s9guo@eng.ucsd.edu

Anwesan Pal: a2pal@eng.ucsd.edu

UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Supervised Learning

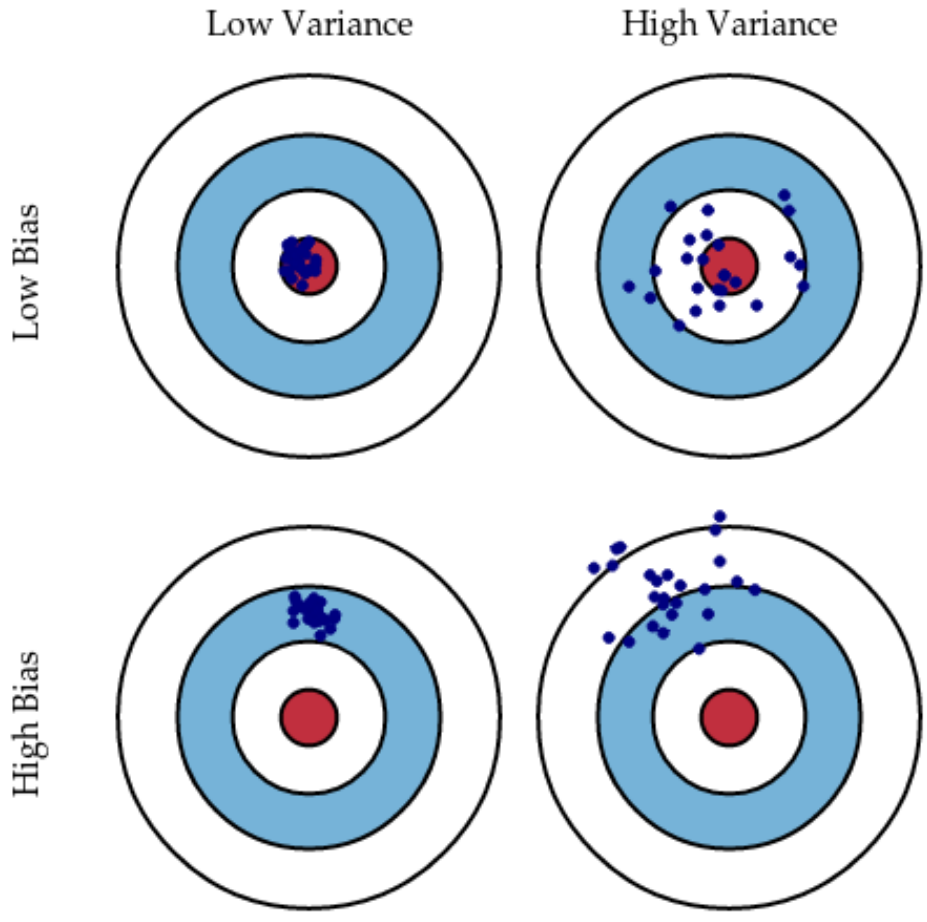
- Given **iid** training data $D := \{\mathbf{x}_i, y_i\}_{i=1}^n$ of examples $\mathbf{x}_i \in \mathbb{R}^d$ with associated labels $y_i \in \mathbb{R}$ (often also written as $D = (X, \mathbf{y})$), generated from an unknown joint pdf
- **Goal**: learn a function: $h : \mathbb{R}^d \to \mathbb{R}$ 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
- The function h should perform "well":
  - **Classification** (discrete $\mathbf{y} \in \{-1, 1\}^n$):
    $\min_h Loss_{0-1}(h) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{h(\mathbf{x}_i) \neq y_i}$
  - **Regression** (continuous $\mathbf{y} \in \mathbb{R}^n$):
    $\min_h RMSE(h) := \sqrt{\frac{1}{n} \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2}$

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.

Bias-Variance Trade-off



Bias-Variance Decomposition

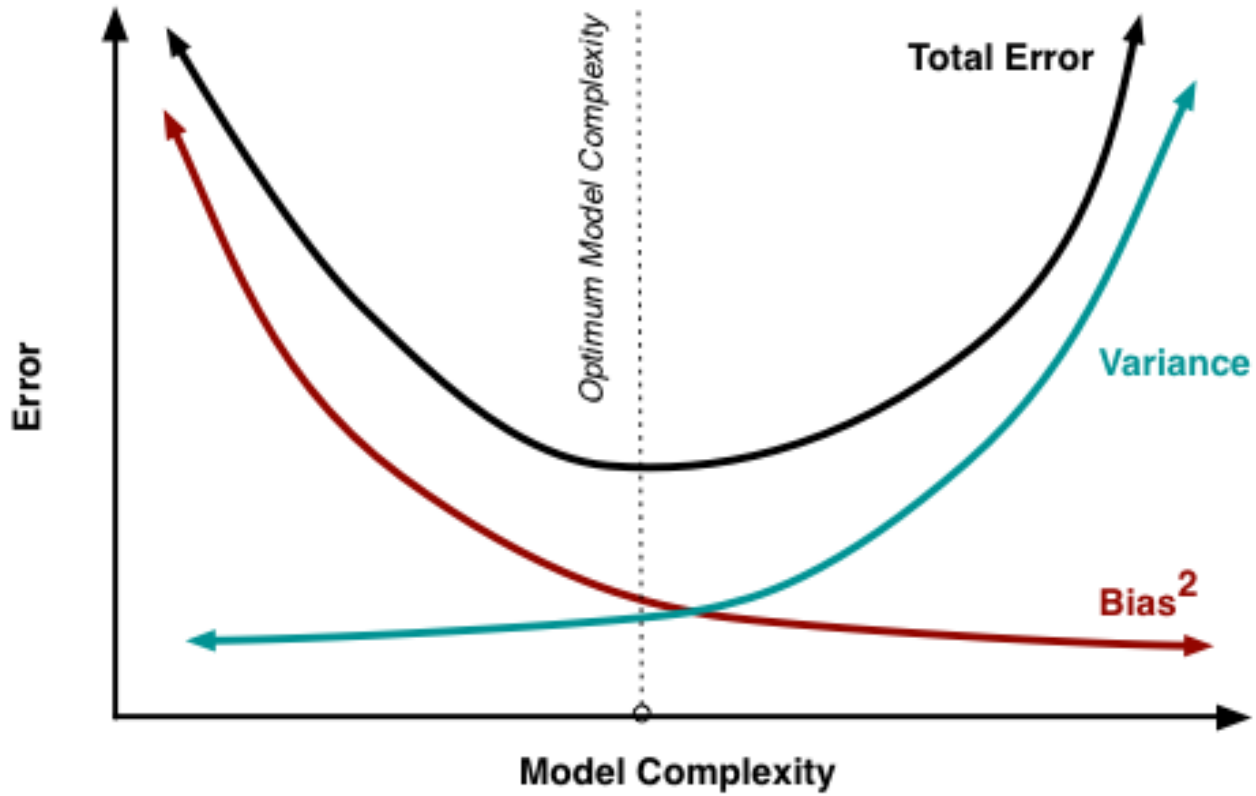
- Given **iid** data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a new independent sample $(\mathbf{x}, y)$, and a regression model $h(\mathbf{x}, D) = \hat{y}$, the expected squared error of h is:

$$\mathbb{E}_{(\mathbf{x},y),D}(h(\mathbf{x}, D) - y)^2 = \underbrace{\mathbb{E}_{\mathbf{x},D}(h(\mathbf{x}, D) - \bar{h}(\mathbf{x}))^2}_{\textbf{Variance (overfitting)}} + \underbrace{\mathbb{E}_{\mathbf{x}}(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2}_{\textbf{Bias}^2\ \textbf{(underfitting)}}$$
$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y)}(\bar{y}(\mathbf{x}) - y)^2}_{\textbf{Noise}}$$

  where $\bar{h}(\mathbf{x}) := \mathbb{E}_D h(\mathbf{x}, D)$ and $\bar{y}(\mathbf{x}) := \mathbb{E}[y \mid \mathbf{x}]$.

- Bias is independent of n and decreases with model complexity
- Variance decreases with n and increases with model complexity
- **Occum's Razor** the best h from a family of good models $\mathcal{H}$ is the one with the lowest complexity

Bias-Variance Decomposition



Parameteric Learning

- Represent the pdfs $p(y|\mathbf{x};\omega)$ (discriminative) or $p(y,\mathbf{x};\omega)$ (generative) with parameters ω
- Estimate/optimize/learn ω based on the training set D in a way that produces good results on the test set
- In either case (discriminative or generative), we distinguish between two types of parameter estimation:
  - **Maximum Likelihood Estimation (MLE)**:
    $\omega_{MLE} := \arg\max_{\omega} p(D \mid \omega)$
  - **Maximum A Posteriori (MAP)**:
    $\omega_{MAP} := \arg\max_{\omega} p(\omega \mid D) = \arg\max_{\omega} p(D \mid \omega)p(\omega)$

  depending on if we use a prior pdf $p(\omega)$ for the parameters.
- **Regularization**: motivated by Occum's Razor, we impose a penalty on the complexity of the learned model h or more concretely on the size $\|\omega\|$ and dimensionality of its parameters

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 $X \in \mathbb{R}^{n\times d}$:

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

- An MLE estimate for ω is obtained as follows:

$$\omega_{MLE} = \arg\max_{\omega} \log p(\mathbf{y} \mid X,\omega) = \arg\min_{\omega} (\mathbf{y} - X\omega)^T V^{-1}(\mathbf{y} - X\omega)$$

$$= \arg\min_{\omega} \| V^{-1/2}(\mathbf{y} - X\omega)\|_2^2 = \boxed{(X^T V^{-1} X)^{-1} X^T V^{-1}\mathbf{y}}$$

- Given a test example $\mathbf{x} \in \mathbb{R}^d$, linear regression produces the output:

$$\hat{y} = \arg\max_{y} \log p(y \mid \mathbf{x}, \omega_{MLE}) = \mathbf{x}^T \omega_{MLE}$$

Discriminative Regression via a Linear Gaussian Model

- **Ridge regression**: obtains a MAP estimate for ω using a Gaussian prior $\omega \sim \mathcal{N}(0, \Lambda)$, which is equivalent to (Tikhonov) regularization on ω:

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

Logits

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

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

- **softmax function**: used to convert continuous preferences $z \in \mathbb{R}^K$ into a cathegorical distribution over K classes:

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

Discriminative Classification via a Logistic Model

- **Logistic regression**: uses a discriminative model $p(\mathbf{y}|X,\omega)$ for the discrete labels $\mathbf{y} \in \{-1,1\}^n$ that is a product of sigmoid functions:

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

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

$$\omega_{MLE} = \arg\max_{\omega} \log p(\mathbf{y} \mid X,\omega) = \arg\min_{\omega} \sum_{i=1}^{n} \log\Big(1+\exp(-y_i \mathbf{x}_i^T \omega)\Big)$$

$$\omega_{MAP} = \arg\max_{\omega} \log p(\mathbf{y} \mid X,\omega) + \log p(\omega)$$

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

Discriminative Classification via a Logistic Model

- Logistic regression requires minimizing a convex in ω function and can only be done iteratively:

$$\begin{aligned}
\omega_{MLE}^{(t+1)} &= \omega_{MLE}^{(t)} + \alpha \nabla_\omega \log p(\mathbf{y}|X,\omega) \\
&= \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^{n} y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^T \omega_{MLE}^{(t)})) \\
\omega_{MAP}^{(t+1)} &= \omega_{MAP}^{(t)} + \alpha (\nabla_\omega \log p(\mathbf{y}|X,\omega) + \log p(\omega)) \\
&= \omega_{MAP}^{(t)} + \alpha \left(\sum_{i=1}^{n} y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^T \omega_{MAP}^{(t)})) - \Lambda^{-1} \omega_{MAP}^{(t)} \right)
\end{aligned}$$

- Given a test example $\mathbf{x} \in \mathbb{R}^d$, logistic regression produces the output:

$$\hat{y} = \begin{cases} 1 & \mathbf{x}^T \omega_{MLE} \geq 0 \\ -1 & \mathbf{x}^T \omega_{MLE} < 0 \end{cases}$$

Discriminative Classification via a Logistic Model

- Logistic regression generates a **linear decision boundary**: $0 = \log\left(\frac{p(1|\mathbf{x},\omega)}{p(-1|\mathbf{x},\omega)}\right) = \mathbf{x}^T\omega$.
- This is equivalent to **Gaussian Naive Bayes with shared variance among the classes**
- Logistic regression has **lower bias** but **higher variance** than Gaussian Naive Bayes.
- **Logistic regression with K-classes** ($\mathbf{y} \in \{1, \ldots, K\}^n$) uses a softmax model with parameters $W \in \mathbb{R}^{K \times d}$:

$$p(\mathbf{y}|X, W) = \prod_{i=1}^{n} e_{y_i}^T \mathbf{softmax}(W\mathbf{x}_i) := \prod_{i=1}^{n} e_{y_i}^T \frac{\exp(W\mathbf{x}_i)}{\mathbf{1}^T \exp(W\mathbf{x}_i)}$$

  where e_j is the j-th standard basis vector
- The rest of the derivation is equivalent to the binary logistic regression.

Generative Classification via a Naive Bayes Model

- **Naive Bayes** uses a generative model $p(\mathbf{y}, X \mid \omega, \theta)$ for the discrete labels $\mathbf{y} \in \{1, \ldots, K\}^n$ and *assumes* (naively) that, when conditioned on the label y_i, the elements of the examples $\mathbf{x}_i$ are independent:

$$p(\mathbf{y}, X \mid \omega, \theta) = p(\mathbf{y} \mid \theta) p(X \mid \mathbf{y}, \omega) = p(\mathbf{y} \mid \theta) \prod_{i=1}^{n} \prod_{l=1}^{d} p(X_{il} \mid y_i, \omega)$$

Gaussian Naive Bayes

- uses a Categorical distribution to model $p(\mathbf{y} \mid \theta)$ and a Gaussian distribution to model $p(X_{i,l} \mid y_i, \omega)$ for $X_{il} \in \mathbb{R}$ as follows:

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

where $\omega := \{\mu_{kl}, \sigma_{kl}^2\}$ and obtains these MLE estimates of θ and ω:

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

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

- Given a test example $\mathbf{x} \in \mathbb{R}^d$, the Gaussian Naive Bayes classifier produces the output:

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

Categorical Naive Bayes

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

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

- obtains these MLE estimates of θ and ω with regularization $l \in \mathbb{N}$:

$$\theta_k^{MLE} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} + l}{n + lK} \quad \omega_{kj}^{MLE} = \frac{\sum_{i=1}^{n} \sum_{l=1}^{d} \mathbb{1}\{X_{il} = j, y_i = k\} + l}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} + lJ}$$

Given a test example $\mathbf{x} \in \{1, \ldots, J\}^d$, the categorical Naive Bayes classifier produces the output:

$$\hat{y} = \underset{y \in \{1, \ldots, K\}}{\arg\max} \log \theta_y^{MLE} + \sum_{l=1}^{d} \log \omega_{y, \mathbf{x}_l}^{MLE}$$

Gaussian Discriminant Analysis

- Removes the naive assumption from Gaussian Naive Bayes
- Uses a generative model $p(\mathbf{y}, X \mid \omega)$ for the discrete labels $\mathbf{y} \in \{1, \ldots, K\}^n$ without any conditional independence assumptions:

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

$$p(\mathbf{y} \mid \theta) := \prod_{i=1}^{n} \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \qquad p(\mathbf{x}_i \mid y_i = k, \omega) := \phi(\mathbf{x}_i; \mu_k, \Sigma_k)$$

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

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = k\} \quad \mu_k^{MLE} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}$$

$$\Sigma_k^{MLE} = \frac{\sum_{i=1}^{n} (\mathbf{x}_i - \mu_k^{MLE})(\mathbf{x}_i - \mu_k^{MLE})^T \mathbb{1}\{y_i = k\}}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}$$

Determining MLE Parameters

- $\max_{\theta,\omega} \log p(\mathbf{y}, X \mid \omega, \theta)$ subject to $\sum_{k=1}^{K} \theta_k = 1$
- $\log p(\mathbf{y}, X \mid \omega, \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\}(\log \theta_k + \log \phi(\mathbf{x}_i; \mu_k, \Sigma_k))$
- Need to solve the following:
  - $\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \theta_k$ subject to $\sum_{k=1}^{K} \theta_k = 1$
  - $\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \frac{d}{d\mu_j} \log \phi(\mathbf{x}_i; \mu_j, \Sigma_j) = 0$
  - $\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \frac{d}{d\Sigma_j} \log \phi(\mathbf{x}_i; \mu_j, \Sigma_j) = 0$

Maximum Likelihood θ

- Constrained optimization wrt θ:
  - θ is restricted to a simplex
  - cannot simply take gradient of the cost function

- **Handling simplex constraints**: express θ_k using a softmax function:

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

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

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

$$\Rightarrow \boxed{\theta_j^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = j\}}$$

Maximum Likelihood Mean

- $\frac{d}{d\mu} \log \phi(x; \mu, \Sigma) = -\frac{1}{2} \frac{d}{d\mu} (x - \mu)^T \Sigma^{-1} (x - \mu) = -(x - \mu)^T \Sigma^{-1}$
- $-\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} (\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} = 0 \quad \Rightarrow \quad \boxed{\mu_j^{MLE} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = j\} \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = j\}}}$

Maximum Likelihood Covariance

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

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

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

Gaussian Discriminant Analysis

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

$$p(\mathbf{y} \mid \theta) := \prod_{i=1}^n \prod_{k=1}^K \theta_k^{\mathbb{1}\{y_i=k\}} \quad p(\mathbf{x}_i \mid y_i = k, \omega) := \sum_{j=1}^J \alpha_{kj} \phi(\mathbf{x}_i; \mu_{kj}, \Sigma_{kj})$$

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

Project 1 Tips

- Define K color classes, e.g., barrel-red, not-barrel-red, brown, yellow
- Label examples for each color class to obtain a training dataset $D = \{\mathbf{x}_i, y_i\}$ (use **roipoly**)
- Train a discriminative $p(y \mid \mathbf{x})$ (Logistic Regression) or $p(y \mid \mathbf{x})$ generative (Gaussian or Gaussian Mixture) model
- Given a test image, classify each pixel into one of the K color classes using your model
- Find red regions (use **findContours**)
- Enumerate red region combinations and score them based on "barrelness" (use **regionprops**)
- Experiment with different colorspaces and parameters (e.g., number of Gaussians)
- Use linear regression to obtain the distance to the barrel from the detected bounding box

Example: findContours

- Use the openCV function "findContours" to combine individual pixels into red regions:

```
import numpy as np
import cv2
im = cv2.imread('test.jpg')
binaryIm = myRedDetector(im)
contours, hierarchy = cv2.findContours(binaryIm,
    cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

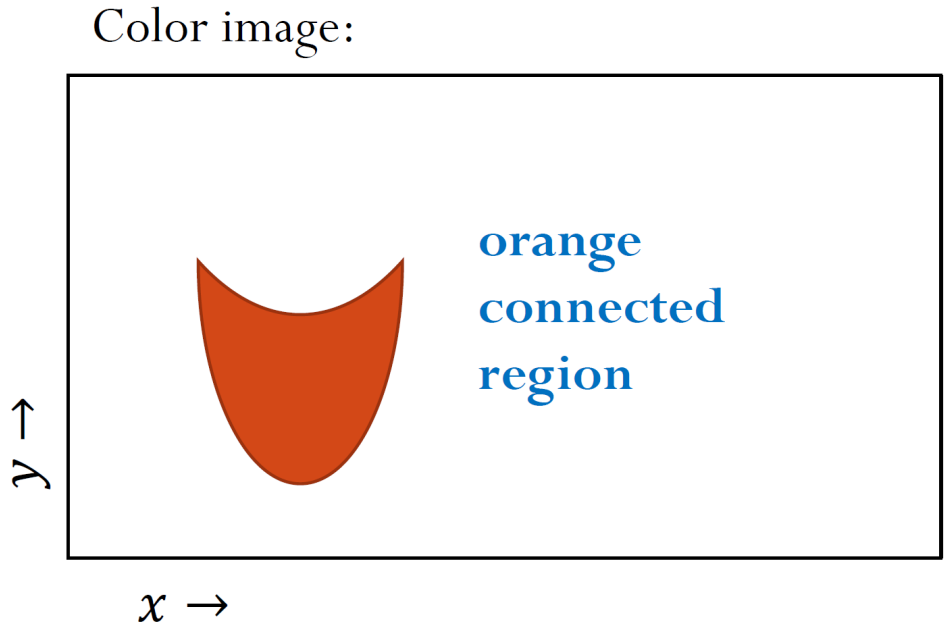
Orange Ball Recognition

- Center of mass:
  $(c_X, c_Y) = \frac{1}{N_p} \sum_p (x_p, y_p)$
- Fit an ellipse:

$$V_{XX} = \frac{1}{N_p} \sum_p (x_p - c_X)^2$$

$$V_{YY} = \frac{1}{N_p} \sum_p (y_p - c_Y)^2$$

$$V_{XY} = \frac{1}{N_p} \sum_p (x_p - c_X)(y_p - c_Y)$$



- Recognize a spherical ball based on thresholds ϵ_0, ϵ_1 on the eigenvalues λ_0, λ_1 of $\begin{bmatrix} V_{XX} & V_{XY} \\ V_{XY} & V_{YY} \end{bmatrix}$
  - size: $\min \lambda_1, \lambda_2 \geq \epsilon_0$
  - eccentricity: $1 - \epsilon_1 \leq \frac{\lambda_1}{\lambda_2} \leq 1 + \epsilon_1$