ECE276A: Sensing & Estimation in Robotics
Lecture 4: Supervised Learning

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qif007@eng.ucsd.edu
Tianyu Wang: tiw161@eng.ucsd.edu
Ibrahim Akbar: iakbar@eng.ucsd.edu
You-Yi Jau: yjau@eng.ucsd.edu
Harshini Rajachander: hrajacha@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.

Parameteric Learning

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

Parameteric Learning

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

MLE	Discriminative Model	Generative Model
Training	$\omega_{MLE} := \arg\max_{\omega} p(\mathbf{y} \mid X, \omega)$	$\omega_{MLE} := \arg\max_{\omega} p(\mathbf{y}, X \mid \omega)$
Testing	$\arg\max_{y^*} p(y^* \mid \mathbf{x}^*, \omega_{MLE})$	$\arg\max_{y^*} p(y^*, \mathbf{x}^* \mid \omega_{MLE})$

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

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

- **Bayesian Inference**:

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

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 serves to condition the data numerically

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

Discriminative Regression via a Linear Gaussian Model

- MLE using the data log-likelihood we derived:

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

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

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

- and solve for ω:

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

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:

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

- The MAP estimate of ω is:

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

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

Linear Regression Summary

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

$$y^*=\arg\max_y \log p(y\mid\mathbf{x}^*,\omega^*)=(\mathbf{x}^*)^T\omega^*$$

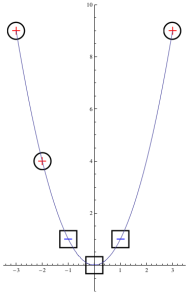
- The test expression is obtained from the gradient of the log-likelihood with respect to y:

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

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:

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

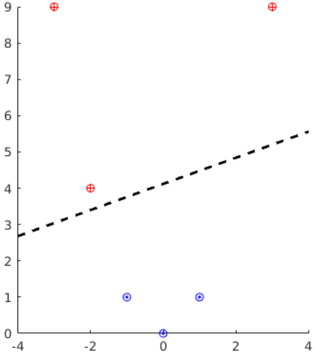
- Let the discrminative model be:

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

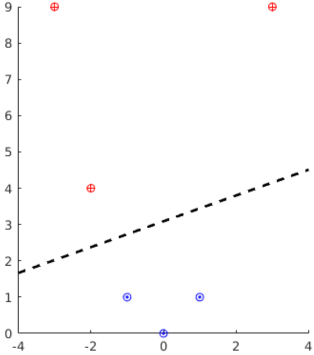
Linear Regression Example

- **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 $0 = \mathbf{x}^T \omega$:

MLE:



MAP:



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:

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

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

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

Discriminative Classification via a Logistic Model

- $\nabla_\omega \left(-\log p(\mathbf{y} \mid X, \omega)\right) = 0$ does not have a closed-form solution
- The negative log-likelihood $-\log p(\mathbf{y} \mid X, \omega)$ is **convex** in ω:
  - The composition of an affine function $f_i(\omega) := -y_i \mathbf{x}_i^T \omega$ 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:

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

Logistic Regression Summary

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

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

- Training: given data $D = (X, \mathbf{y})$, optimize the model parameters:
  - MLE: $\omega_{MLE}^{(t+1)} = \omega_{MLE}^{(t)} + \alpha \sum_{i=1}^{n} y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^T \omega_{MLE}^{(t)}))$
  - MAP: $\omega_{MAP}^{(t+1)} = \omega_{MAP}^{(t)} + \alpha \left(\sum_{i=1}^{n} y_i \mathbf{x}_i (1 - \sigma(y_i \mathbf{x}_i^T \omega_{MAP}^{(t)})) - \Lambda^{-1} \omega_{MAP}^{(t)} \right)$

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

$$y^* = \begin{cases} 1 & (\mathbf{x}^*)^T \omega^* \geq 0 \\ -1 & (\mathbf{x}^*)^T \omega^* < 0 \end{cases}$$

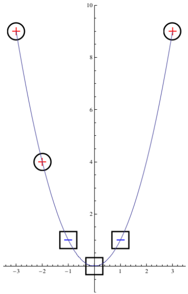
- Logistic regression generates a **linear decision boundary**:

$$0 = \log\left(\frac{p(1 \mid \mathbf{x}^*, \omega^*)}{p(-1 \mid \mathbf{x}^*, \omega^*)}\right) = (\mathbf{x}^*)^T \omega^*$$

Logistic Regression Example

- Consider the same data as before:

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

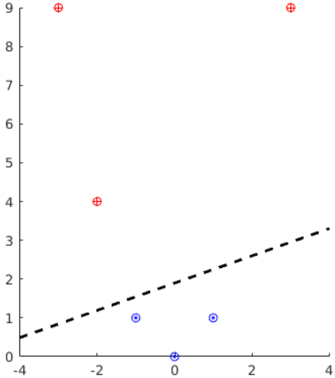


Logistic Regression Example

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

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

- 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 $0 = \mathbf{x}^T \omega$:



K-ary Logistic Regression

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

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

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

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

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

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

Generative Classification via a Naive Bayes Model

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

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

Gaussian Naive Bayes

- GNB 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 $\mathbf{x}_{il} \in \mathbb{R}$ and $\omega := \{\mu_{kl}, \sigma_{kl}^2\}$

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

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

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

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

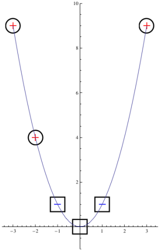
- Given a test example $\mathbf{x}^* \in \mathbb{R}^d$, the GNB classifier produces the output:

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

Gaussian Naive Bayes Example

- Consider the same data as before:

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



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

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

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

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

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

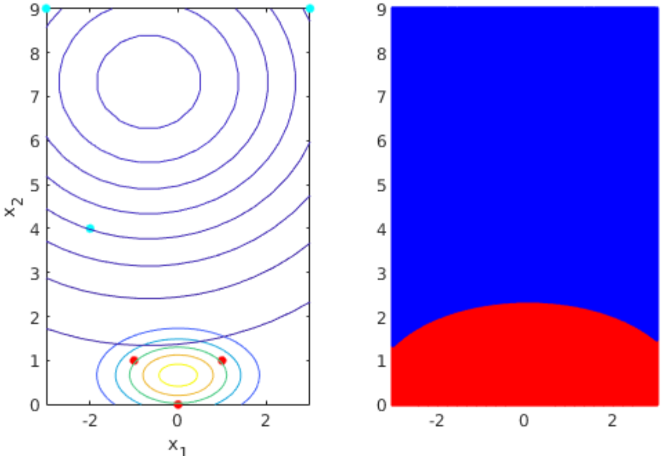
	$l = 1$	$l = 2$
$k = 1$	2.62	2.36
$k = -1$	1.05	6.68

Gaussian Naive Bayes Example

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

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

- 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:

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

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

- Logistic regression always generates a **linear decision boundary**:

$$0 = \log\left(\frac{p(1 \mid \mathbf{x}, \omega)}{p(-1 \mid \mathbf{x}, \omega)}\right) = \mathbf{x}^T\omega$$

- Logistic regression has **lower bias** but **higher variance** than Gaussian Naive Bayes.

Categorical Naive Bayes

- CNB 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:

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

- 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 $\mathbf{x}^* \in \{1, \ldots, J\}^d$, CNB predicts:

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

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 the MLE Parameters

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

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

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

- The cost function is separable and leads to three independent optimization problems:
  - $\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \theta_k$ 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:

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

- 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

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

Maximum Likelihood Covariance

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

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

Gaussian Discriminant Analysis

- If the training set D is small, one might restrict the covariance 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.