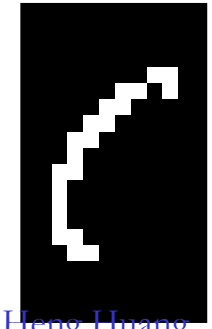
DATA, MSML, BIOI 602 Principles of Data Science (Fall 2024)

Classification Methods

Heng Huang
Department of Computer Science

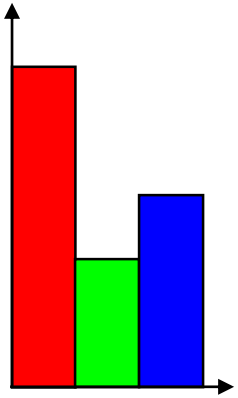
Basic Classification in ML

Input $\mathbf{x} \in \mathcal{X}$	Output $y \in \mathcal{Y}$
Spam filtering	 →  **Binary**
Character recognition	 → **C** **Multi-Class**

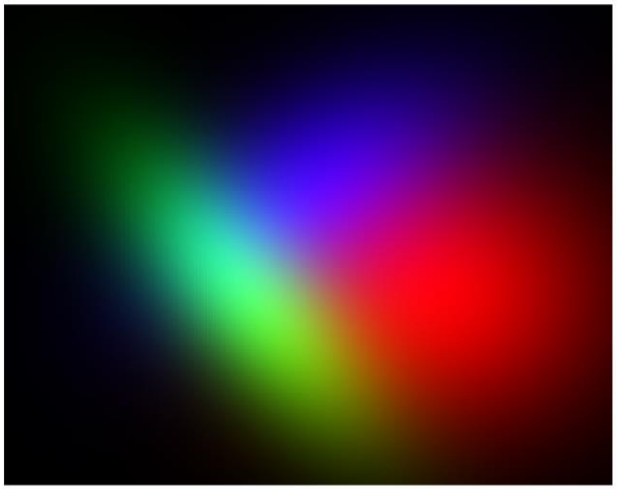
Generative Models

As in the binary case:

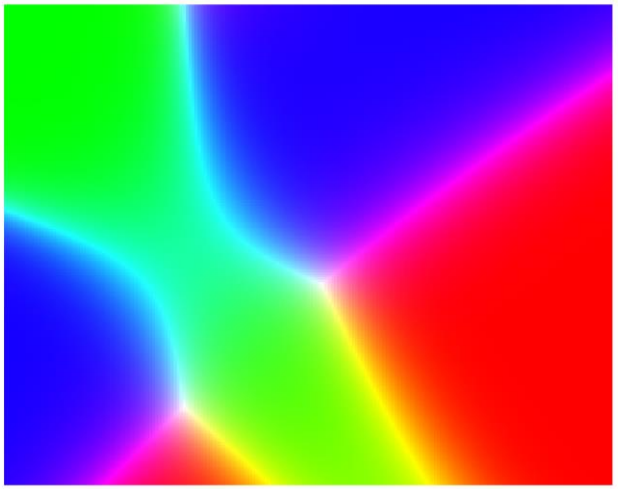
1. Learn p(y) and p(y|x)
2. Use Bayes rule: $p(y=k|x) = \frac{p(x|y=k)p(y=k)}{p(x)}$
3. Classify as $\hat{y}(x) = \text{argmax}_y \, p(y|x)$



p(y)



p(x|y)



p(y|x)

Linear Regression of An Indicator Matrix

If $\mathcal{G}$ has K classes, there will be K class indicators Y_k, $k = 1, ..., K$.

g	$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	$\mathbf{y}_4$
3	0	0	1	0
1	1	0	0	0
2	0	1	0	0
4	0	0	0	1
1	1	0	0	0

Fit a linear regression model for each Y_k, $k = 1, 2, ..., K$, using X:

$$\hat{\mathbf{y}}_k = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}_k \ .$$

Define $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k)$:

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \ .$$

Classification Procedure

Define $\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$.

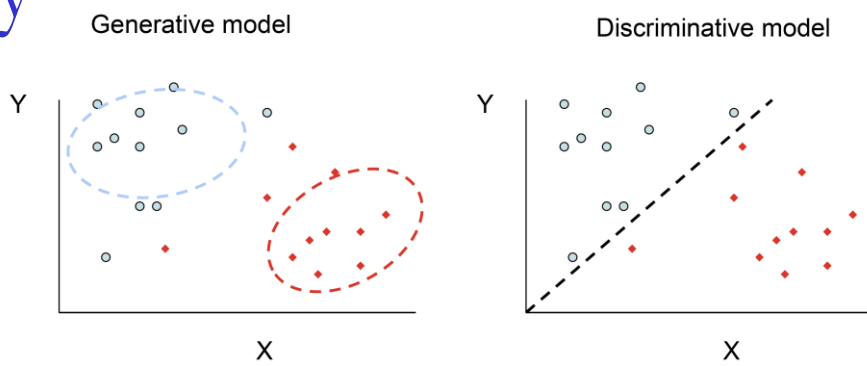
For a new observation with input x, compute the fitted output

$$\begin{aligned}\hat{f}(x) &= [(1,x)\hat{B}]^T \\ &= [(1,x_1,x_2,...,x_p)\hat{B}]^T \\ &= \begin{pmatrix} \hat{f}_1(x) \\ \hat{f}_2(x) \\ ... \\ \hat{f}_K(x) \end{pmatrix}\end{aligned}$$

Identify the largest component of $\hat{f}(x)$ and classify accordingly:

$$\hat{G}(x) = \arg\max_{k\in\mathcal{G}} \hat{f}_k(x) \ .$$

Classification - Summary



- **Want to Learn**: h:**X** ↦ Y
  - **X** – features
  - Y – target classes

- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some **functional form for P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X= x)
  - This is a '***generative***' **model**
    - **Indirect** computation of P(Y|X) through Bayes rule
    - But, **can generate a sample of the data**, P(X) = $\sum_y$ P(y) P(X|y)
- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some **functional form for P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the '***discriminative***' **model**
    - Directly learn P(Y|X)
    - But **cannot obtain a sample of the data**, because P(X) is not available

Ref: Carlos Guestrin

Linear Discriminant Analysis

Essentially minimum error Bayes' classifier

Assumes that the conditional class densities are (multivariate) Gaussian

Assumes equal covariance for every class

Posterior probability $\Pr(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$ ← Application of Bayes rule

π_k is the prior probability for class k

$f_k(x)$ is class conditional density or likelihood density

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}^{-1}(x-\mu_k)\right)$$

Linear Discriminant Analysis

$$
\begin{aligned}
\hat{G}(x) &= \arg\max_k Pr(G = k \mid X = x) \\
&= \arg\max_k f_k(x)\pi_k \\
&= \arg\max_k \log(f_k(x)\pi_k) \\
&= \arg\max_k \Big[-\log((2\pi)^{p/2}|\Sigma|^{1/2}) \\
&\quad -\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big] \\
&= \arg\max_k \left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \right] \\
&\quad -\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) \\
&= x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x
\end{aligned}
$$

Linear Discriminant Analysis

- To sum up

$$\hat{G}(x) = \arg\max_k \left[x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k) \right]$$

- Define the **linear discriminant function**

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

Then

$$\hat{G}(x) = \arg\max_k \delta_k(x)$$

- The decision boundary between class *k* and *l* is:

$$\{x : \delta_k(x) = \delta_l(x)\}$$

- Or equivalently the following holds

$$\log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l) = 0$$

Linear Discriminant Analysis

Consider the classification through decision boundary

$$\log \frac{\Pr(G=k \mid X=x)}{\Pr(G=l \mid X=x)} = \log \frac{\pi_k}{\pi_l} + \log \frac{f_k}{f_l}$$

$$= \underbrace{(\log \pi_k + x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k)}_{\delta_k(x)} - \underbrace{(\log \pi_l + x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l)}_{\delta_l(x)}$$

Classification rule: $\hat{G}(x) = \arg\max_k \delta_k(x)$

is equivalent to: $\hat{G}(x) = \arg\max_k \Pr(G=k \mid X=x)$

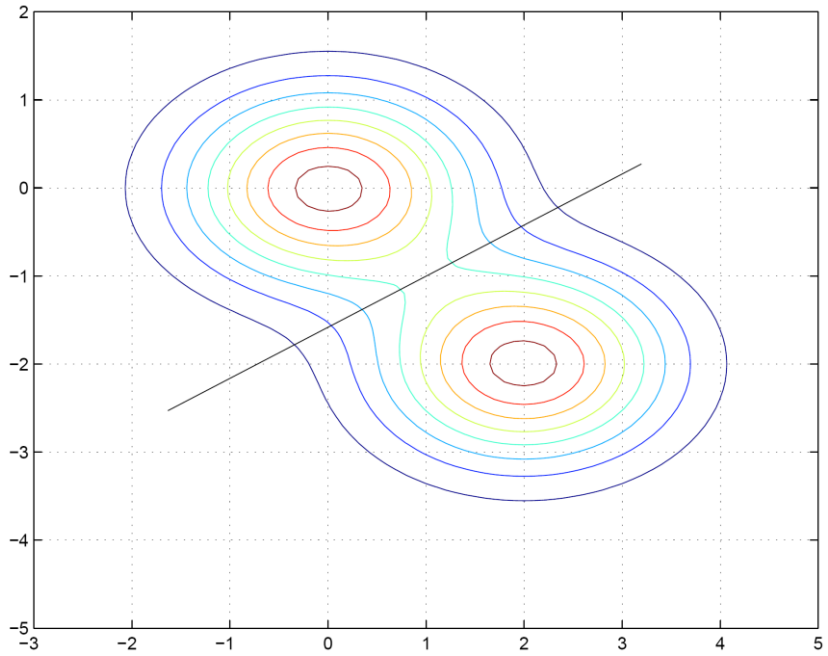
The good old Bayes classifier!

Binary Classification Example

Binary classification ($k = 1, l = 2$):

- Define $a_0 = \log \frac{\pi_1}{\pi_2} - \frac{1}{2}(\mu_1 + \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2)$.
- Define $(a_1, a_2, ..., a_p)^T = \Sigma^{-1}(\mu_1 - \mu_2)$.
- Classify to class 1 if $a_0 + \sum_{j=1}^{p} a_j x_j > 0$; to class 2 otherwise.
- An example:
  - $\pi_1 = \pi_2 = 0.5$.
  - $\mu_1 = (0, 0)^T$, $\mu_2 = (2, -2)^T$.
  - $\Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.5625 \end{pmatrix}$.
  - Decision boundary:

$$5.56 - 2.00x_1 + 3.56x_2 = 0.0 \ .$$



Estimate Gaussian Distributions

In practice, we need to estimate the Gaussian distribution.

Total N input-output pairs
N_k number of pairs in class k
Total number of classes: K
Training data utilized to estimate

$$(g_i, x_i),\ i = 1:N$$

Prior probabilities: $\hat{\pi}_k = N_k / N$

Means: $\hat{\mu}_k = \sum_{g_i = k} x_i / N_k$

Covariance matrix: $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N - K)$

Diabetes Data Example

- **Diabetes data**

  The diabetes data set is taken from the UCI machine learning database repository at:
  http://www.ics.uci.edu/~mlearn/Machine-Learning.html .

  The original source of the data is the National Institute of Diabetes and Digestive and Kidney Diseases. There are 768 cases in the data set, of which 268 show signs of diabetes according to World Health organization criteria. Each case contains 8 quantitative variables, including diastolic blood pressure, triceps skin fold thickness, a body mass index, etc.

  – Two classes: with or without signs of diabetes.
  – Denote the 8 original variables by $\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_8$
  – Remove the mean of $\tilde{X}_j$ and normalize it to unit variance.

Diabetes Data Set

Two input variables computed from the principal components of the original 8 variables.

$$X_1 = 0.1284\tilde{X}_1 + 0.3931\tilde{X}_2 + 0.3600\tilde{X}_3 + 0.4398\tilde{X}_4 + 0.4350\tilde{X}_5 + 0.4519\tilde{X}_6 + 0.2706\tilde{X}_7 + 0.1980\tilde{X}_8$$

$$X_2 = 0.5938\tilde{X}_1 + 0.1740\tilde{X}_2 + 0.1839\tilde{X}_3 - 0.3320\tilde{X}_4 - 0.2508\tilde{X}_5 - 0.1010\tilde{X}_6 - 0.1221\tilde{X}_7 + 0.6206\tilde{X}_8$$

Prior probabilities: $\hat{\pi}_1 = 0.651$, $\hat{\pi}_2 = 0.349$.

$\hat{\mu}_1 = (-0.4035, -0.1935)^T$, $\hat{\mu}_2 = (0.7528, 0.3611)^T$.

$$\hat{\Sigma} = \begin{pmatrix} 1.7925 & -0.1461 \\ -0.1461 & 1.6634 \end{pmatrix}$$

Classification rule:

$$\hat{G}(x) = \begin{cases} 1 & 0.7748 - 0.6771x_1 - 0.3929x_2 \geq 0 \\ 2 & \text{otherwise} \end{cases}$$

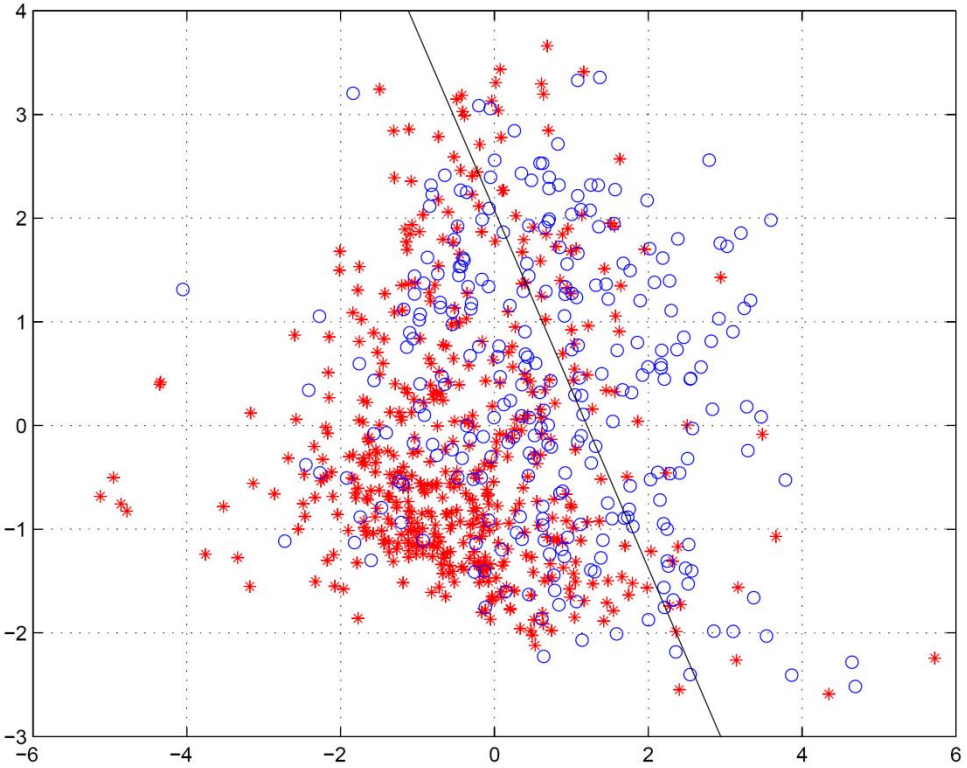
$$= \begin{cases} 1 & 1.1443 - x_1 - 0.5802x_2 \geq 0 \\ 2 & \text{otherwise} \end{cases}$$

Result of Linear Regression Based Classification

- Classification error rate: 28.52%.

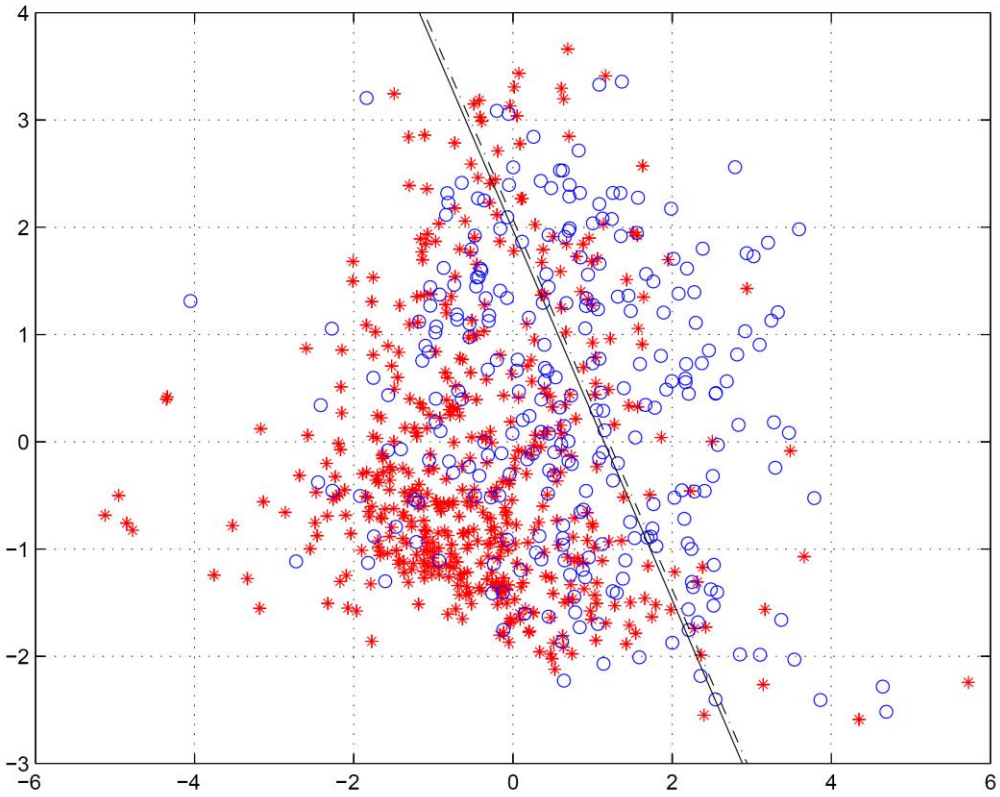
$$\hat{G}(x) = \begin{cases} 1 & \hat{Y}_1 \geq \hat{Y}_2 \\ 2 & \hat{Y}_1 < \hat{Y}_2 \end{cases}$$

$$= \begin{cases} 1 & 0.151 - 0.1256X_1 - 0.0729X_2 \geq 0 \\ 2 & otherwise \end{cases}$$



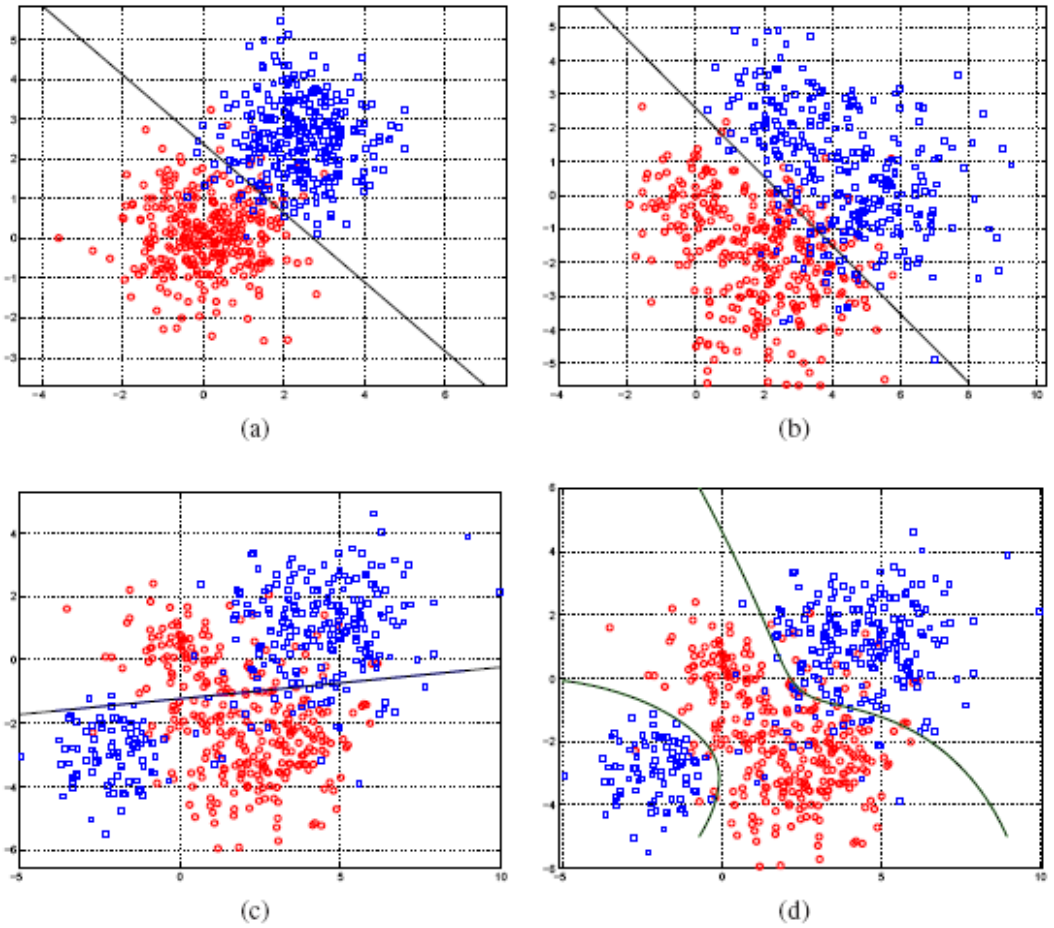
LDA Classification Result

- The scatter plot follows. Without diabetes: stars (class 1), with diabetes: circles (class 2). Solid line: classification boundary obtained by LDA. Dash dot line: boundary obtained by linear regression of indicator matrix.
- Within training data classification error rate: 28.26%.



Simulated Examples

- LDA applied to three simulated data sets in (a)-(c). (a): The true within class densities are Gaussian with identical covariance matrices across classes. (b) and (c): The true within class densities are mixtures of two Gaussians.
- (d): The data set is the same as that in (c). Decision boundaries are obtained by modeling each class by a mixture of two Gaussians.



Quadratic Discriminant Analysis

- Relaxes the same covariance assumption – class conditional probability densities (still multivariate Gaussians) are allowed to have different covariant matrices
- The class decision boundaries are not linear rather quadratic

$$\log \frac{\Pr(G=k \mid X=x)}{\Pr(G=l \mid X=x)} = \log \frac{\pi_k}{\pi_l} + \log \frac{f_k}{f_l} =$$

$$\underbrace{(\log \pi_k - \frac{1}{2}(x-\mu_k)^T \boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k|)}_{\delta_k(x)} - \underbrace{(\log \pi_l - \frac{1}{2}(x-\mu_l)^T \boldsymbol{\Sigma}_l^{-1}(x-\mu_l) - \frac{1}{2}\log|\boldsymbol{\Sigma}_l|)}_{\delta_l(x)}$$

Classification rule: $\hat{G}(x) = \arg\max_k \delta_k(x)$

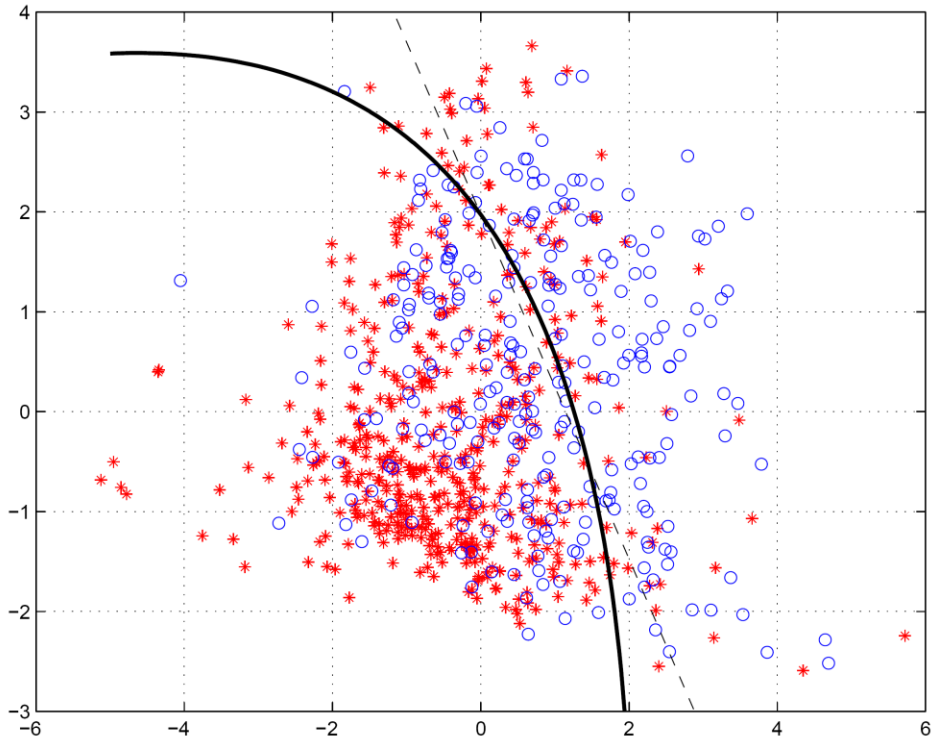
Diabetes Data Set

Prior probabilities: $\hat{\pi}_1 = 0.651, \hat{\pi}_2 = 0.349.$

$$\hat{\mu}_1 = (-0.4035, -0.1935)^T, \hat{\mu}_2 = (0.7528, 0.3611)^T$$

$$\hat{\Sigma}_1 = \begin{pmatrix} 1.6769 & -0.0461 \\ -0.0461 & 1.5964 \end{pmatrix}$$

$$\hat{\Sigma}_2 = \begin{pmatrix} 2.0087 & -0.3330 \\ -0.3330 & 1.7887 \end{pmatrix}$$



Within training data
classification error rate: 27.04%.

LDA on Expanded Basis

Expand input space to include X_1X_2, X_1^2, and X_2^2.
Input is five dimensional: $X = (X_1, X_2, X_1X_2, X_1^2, X_2^2)$.

$$\hat{\mu}_1 = \begin{pmatrix} -0.4035 \\ -0.1935 \\ 0.0321 \\ 1.8363 \\ 1.6306 \end{pmatrix} \qquad \hat{\mu}_2 = \begin{pmatrix} 0.7528 \\ 0.3611 \\ -0.0599 \\ 2.5680 \\ 1.9124 \end{pmatrix}$$

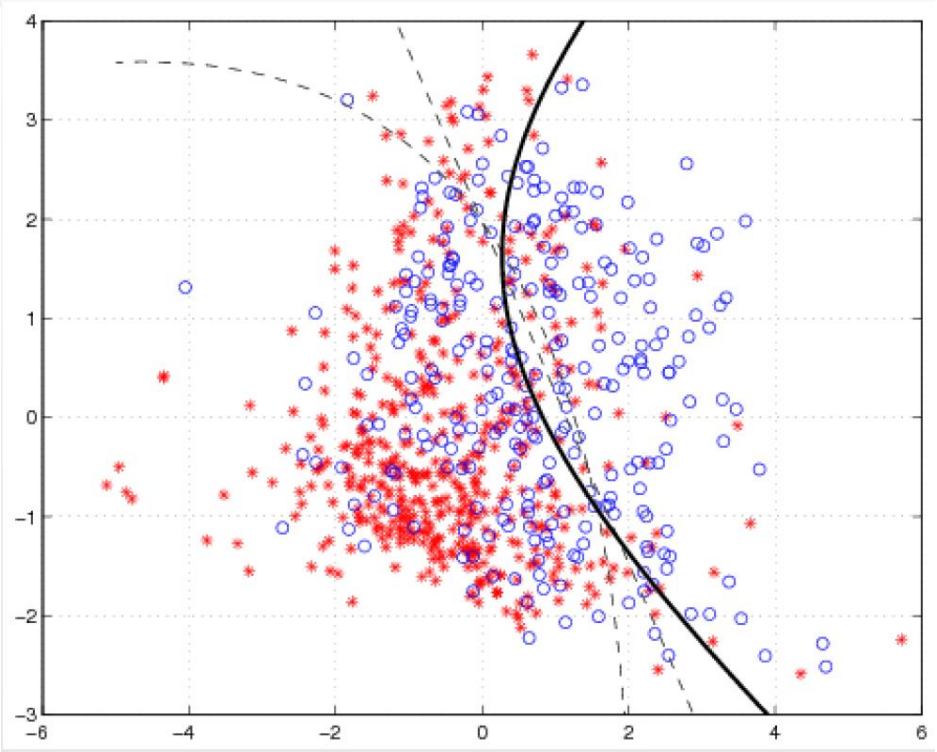
$$\hat{\Sigma} = \begin{pmatrix} 1.7925 & -0.1461 & -0.6254 & 0.3548 & 0.5215 \\ -0.1461 & 1.6634 & 0.6073 & -0.7421 & 1.2193 \\ -0.6254 & 0.6073 & 3.5751 & -1.1118 & -0.5044 \\ 0.3548 & -0.7421 & -1.1118 & 12.3355 & -0.0957 \\ 0.5215 & 1.2193 & -0.5044 & -0.0957 & 4.4650 \end{pmatrix}$$

LDA on Expanded Basis

Classification boundary:

$$0.651 - 0.728x_1 - 0.552x_2 - 0.006x_1x_2 - 0.071x_1^2 + 0.170x_2^2 = 0$$

If the linear function on the right hand side is non-negative, classify as 1; otherwise 2.



Within training data
classification error rate: 26.82%.

The Classification Problem

The **linear regression model** discussed in the previous lesson assumes that the response variable y is **quantitative (metrical)**

- in many situations, the response variable is instead **qualitative (categorical)**

Qualitative variables take values in an **unordered set** $\mathcal{C} = \{\text{"cat}_1\text{"}, \dots, \text{"cat}_C\text{"}\}$, such as:

- `eye color` $\in \{\text{"brown"}, \text{"blue"}, \text{"green"}\}$
- `email` $\in \{\text{"spam"}, \text{"not spam"}\}$

Metric data

- Describe a quantity
- An ordering is defined
- A distance is defined

Categorical data

- Describe membership categories
- It is not meaningful to apply an ordering
- It is not meaningful to compute distances

Why Not Linear Regression?

Suppose that we are trying to **estimate the medical condition** of a patient in the emergency room based on her **symptoms**

There are three possibilities: `stroke`, `drug overdose` and `epileptic seizure`

We could consider encoding these values as a **quantitative** response variable, y, as

$$y = \begin{cases} 1 & \text{if } \texttt{stroke} \\ 2 & \text{if } \texttt{drug overdose} \\ 3 & \text{if } \texttt{epileptic seizure} \end{cases}$$

However, we are implicitly saying that the «difference» between `drug overdose` and `stroke` is the same as the «difference» between `epileptic seizure` and `drug overdose`, which **does not make much sense**

Why Not Linear Regression?

We can also change the encoding to

$$y = \begin{cases} 1 & \text{if } \texttt{epileptic seizure} \\ 2 & \text{if } \texttt{stroke} \\ 3 & \text{if } \texttt{drug overdose} \end{cases}$$

This would imply a **totally different relationship** among the three conditions

- each of these codings would produce fundamentally **different linear models**…
- …that would ultimately lead to **different sets of estimates** on test observations

In general, there is no natural way to convert a qualitative response variable with more than two levels into a quantitative response that is ready for linear regression

Why Not Linear Regression?

With two levels, the situation is better. For instance, perhaps there are only two possibilities for the patient's medical condition: **`stroke`** and **`drug overdose`**

$$y = \begin{cases} 0 & \text{if } \texttt{stroke} \\ 1 & \text{if } \texttt{drug overdose} \end{cases}$$

We can fit a linear regression to this binary response, and classify as **`drug overdose`** if $\hat{y} > 0.5$ and **`stroke`** otherwise, interpreting $\hat{y}$ **as a probability of drug overdose**

However, if we use linear regression, some of our estimates might be **outside the [0, 1] interval,** which does not make sense as a probability. There is nothing that "saturates" the output between 0 and 1. ➡ **Logistic function (Sigmoid)**

Logistic Regression

Purpose: Estimate the probability that a set of input regressors $\boldsymbol{\varphi} \in \mathbb{R}^{d\times 1}$ belong to **one of two classes** $y \in \{0, 1\}$

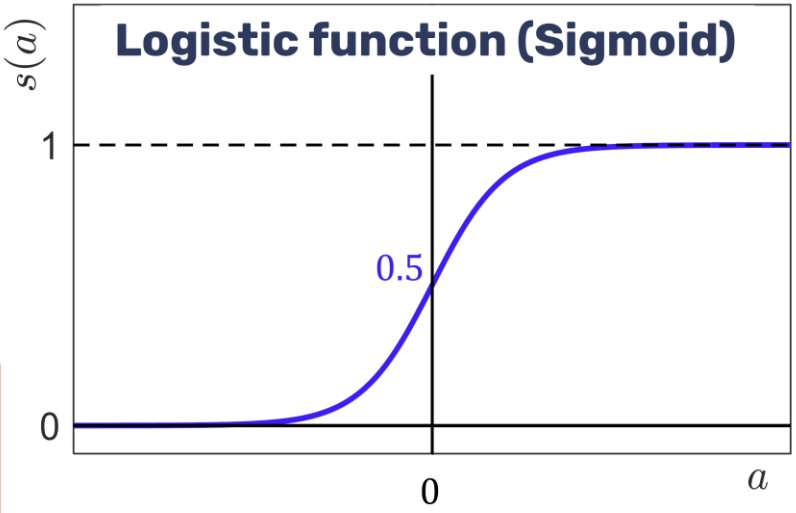
Define the linear combination quantity

$$a = \sum_{i=0}^{d-1} \varphi_i \cdot \theta_i = \boldsymbol{\varphi}^\top \cdot \boldsymbol{\theta}$$

The formula $s(a)$ is the **logistic function**

$$s(a) = \frac{1}{1+e^{-a}} = \frac{e^a}{1+e^a}$$

- $a \gg 0 \Rightarrow s(a) = 1$
- $a \ll 0 \Rightarrow s(a) = 0$



Logistic Regression

Purpose: **Estimate the probability** that a set of input regressors $\boldsymbol{\varphi} \in \mathbb{R}^{d\times 1}$ belong to **one of two classes** $y \in \{0, 1\}$

$$P(y = 1|\boldsymbol{\varphi}) = s(a) = s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{\varphi}^\top \boldsymbol{\theta}}}$$

The output of $s(\boldsymbol{\varphi}^\top \boldsymbol{\theta})$ is **interpreted as a probability**

- $\boldsymbol{\varphi}^\top \boldsymbol{\theta} \gg 0 \Rightarrow s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) \gg 0.5 \Rightarrow P(y = 1|\boldsymbol{\varphi}) \approx 1$ ➡ **φ is classified to class 1**
- $\boldsymbol{\varphi}^\top \boldsymbol{\theta} \ll 0 \Rightarrow s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) \ll 0.5 \Rightarrow P(y = 1|\boldsymbol{\varphi}) \approx 0$ ➡ **φ is classified to class 0**

Logistic Regression Cost Function

Suppose to have at disposal a dataset $\mathcal{D} = \{(\boldsymbol{\varphi}(1), y(1)), \dots, (\boldsymbol{\varphi}(N), y(N))\}$ where $\boldsymbol{\varphi}(i) \in \mathbb{R}^{d\times 1}$ and $y(i) \in \{0, 1\},\ i = 1, \dots, N,\ i.i.d.$

Estimate a logistic regression model $P(y(i) = 1|\boldsymbol{\varphi}(i)) = \dfrac{1}{1 + e^{-\boldsymbol{\varphi}(i)^\top \boldsymbol{\theta}}} \equiv \pi(i)$

The **logistic regression cost function** $J(\boldsymbol{\theta})$ is defined as:

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \big(y(i) \cdot \ln \pi(i) + \big(1 - y(i)\big) \cdot \ln[1 - \pi(i)] \big)$$

Logistic Regression Cost Function

QUIZ: In the logistic regression cost function, where are the parameters $\boldsymbol{\theta}$ that we want to estimate?

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \Big(y(i) \cdot \ln \pi(i) + \big(1 - y(i)\big) \cdot \ln[1 - \pi(i)] \Big)$$

- ❑ In the $y(i)$ terms
- ❑ In the $\ln$ terms
- ❑ In the $\pi(i)$ terms

Logistic Regression Cost Function

Cost function interpretation

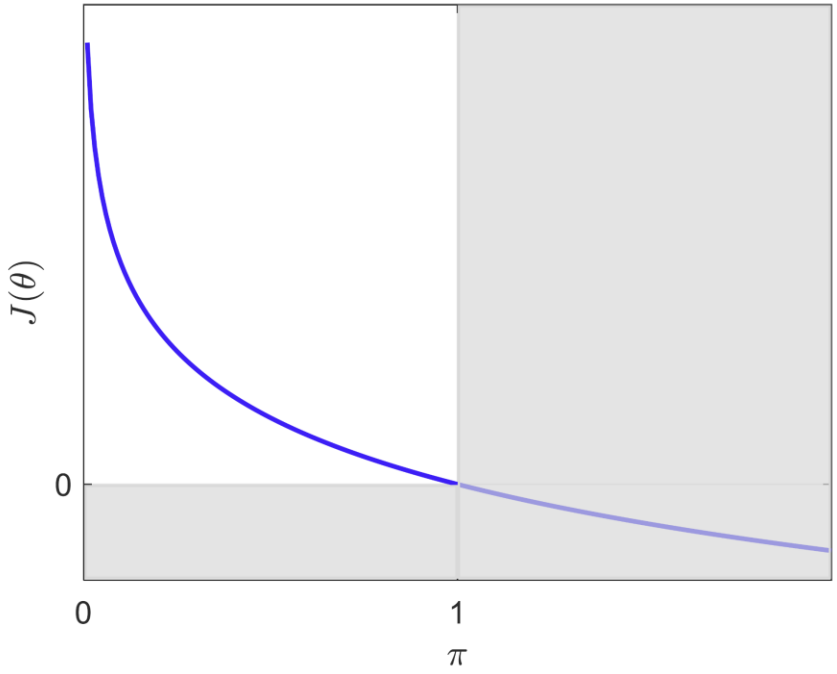
Suppose there is only one datum $\mathcal{D} = \{(\boldsymbol{\varphi}, y)\}$

$$\Rightarrow J(\boldsymbol{\theta}) = \begin{cases} -\ln \pi & \text{if} \quad y = 1 \\ -\ln[1-\pi] & \text{if} \quad y = 0 \end{cases}$$

Case $y = 1$

$J(\boldsymbol{\theta}) = -\ln \pi$

- $J(\boldsymbol{\theta}) \approx 0$ if $y = 1$ and $\pi \approx 1$
- $J(\boldsymbol{\theta}) \approx +\infty$ if $y = 1$ and $\pi \approx 0$



Logistic Regression Cost Function

Cost function interpretation

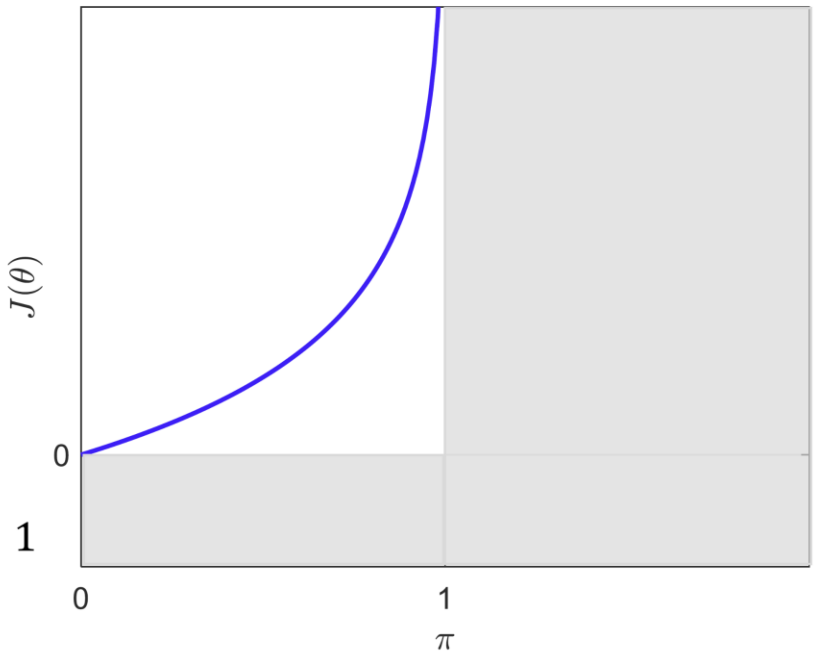
Suppose there is only one datum $\mathcal{D} = \{(\boldsymbol{\varphi}, y)\}$

$$\Rightarrow J(\boldsymbol{\theta}) = \begin{cases} -\ln \pi & \text{if} \quad y = 1 \\ -\ln[1-\pi] & \text{if} \quad y = 0 \end{cases}$$

Case $y = 0$

$J(\boldsymbol{\theta}) = -\ln[1-\pi]$

- $J(\boldsymbol{\theta}) \approx 0$ if $y = 0$ and $\pi \approx 0$
- $J(\boldsymbol{\theta}) \approx +\infty$ if $y = 0$ and $\pi \approx 1$



Computation of the minimum of $J(\boldsymbol{\theta})$

We have to compute the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} \in \mathbb{R}^{d\times 1}$. First, compute the derivative of $s(a) = \frac{1}{1+e^{-a}}$

$$\frac{\partial s(a)}{\partial a} = \frac{\partial}{\partial a}\left[\frac{1}{1+e^{-a}}\right] = \frac{\partial}{\partial a}[(1+e^{-a})^{-1}] = -(1+e^{-a})^{-2}(e^{-a})(-1) = -(1+e^{-a})^{-2}(-e^{-a}) = \frac{e^{-a}}{(1+e^{-a})^2}$$

$$= \frac{1}{(1+e^{-a})}\cdot\frac{e^{-a}}{(1+e^{-a})} = \frac{1}{(1+e^{-a})}\cdot\frac{(1+e^{-a})-1}{1+e^{-a}} = \frac{1}{1+e^{-a}}\cdot\left(\frac{1+e^{-a}}{1+e^{-a}} - \frac{1}{1+e^{-a}}\right) = \boxed{s(a)\cdot[1-s(a)]}$$

In the case where $a = \boldsymbol{\varphi}^\top\boldsymbol{\theta}$, we have that

$$\underset{d\times 1}{\nabla_{\boldsymbol{\theta}} s(\boldsymbol{\varphi}^\top\boldsymbol{\theta})} = \underset{d\times 1}{\boldsymbol{\varphi}}\cdot \underset{1\times 1}{s(\boldsymbol{\varphi}^\top\boldsymbol{\theta})}\cdot[1-\underset{1\times 1}{s(\boldsymbol{\varphi}^\top\boldsymbol{\theta})}] = \boxed{\boldsymbol{\varphi}\cdot\pi\cdot[1-\pi]}$$

Computation of the minimum of $J(\boldsymbol{\theta})$

We can now compute the gradient of $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\big(y(i)\ln\pi(i) + \big(1-y(i)\big)\ln[1-\pi(i)]\ \big) \qquad \pi(i) = \frac{1}{1+e^{-\boldsymbol{\varphi}(i)^T\boldsymbol{\theta}}}$$

$$\underset{d\times 1}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} = -\sum_{i=1}^{N}\left(y(i)\frac{\pi'(i)}{\pi(i)} + \big(1-y(i)\big)\frac{-\pi'(i)}{1-\pi(i)}\right) = -\sum_{i=1}^{N}\left(y(i)\frac{\boldsymbol{\varphi}(i)\pi(i)[1-\pi(i)]}{\pi(i)} + \big(1-y(i)\big)\frac{-\boldsymbol{\varphi}(i)\pi(i)[1-\pi(i)]}{1-\pi(i)}\right)$$

$$= \sum_{i=1}^{N}\big(-y(i)\boldsymbol{\varphi}(i)[1-\pi(i)] - \big(1-y(i)\big)\big(-\boldsymbol{\varphi}(i)\pi(i)\big)\ \big) = \sum_{i=1}^{N}(\boldsymbol{\varphi}(i)\cdot[-y(i)+y(i)\pi(i)] + \boldsymbol{\varphi}(i)\cdot[\pi(i)-y(i)\pi(i)]\ )$$

$$= \sum_{i=1}^{N}(\ \boldsymbol{\varphi}(i)\cdot[-y(i)+y(i)\pi(i)-y(i)\pi(i)+\pi(i)]\ ) = \sum_{i=1}^{N}\underset{d\times 1}{\boldsymbol{\varphi}(i)}\cdot\underset{1\times 1}{\big(\pi(i)-y(i)\big)}$$

Gradient Descent

It can be shown that:

- The cost function $J(\boldsymbol{\theta})$ is **convex** and admits a **unique minimum**
- The equations found by posing $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{0}$ are nonlinear in $\boldsymbol{\theta}$ and it **is not possible** to find a solution in **closed-form**
  - ✓ For this reason, we need to resort to **iterative optimization algorithms**

Use **gradient descent:**

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(k+1)} = \underset{d\times 1}{\widehat{\boldsymbol{\theta}}(k)} - \underset{1\times 1}{\alpha} \cdot \underset{d\times 1}{\nabla J(\boldsymbol{\theta})}\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}(k)}$$

$\alpha \in \mathbb{R}_{>0}$: learning rate

Gradient Descent

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \big(y(i) \cdot \ln \pi(i) + \big(1 - y(i)\big) \cdot \ln[1 - \pi(i)] \big)$$

Repeat {

$$\theta_0 = \theta_0 - \alpha \cdot \sum_{i=1}^{N} \big(\pi(i) - y(i)\big)$$

$$\theta_1 = \theta_2 - \alpha \cdot \sum_{i=1}^{N} \big(\pi(i) - y(i)\big) \cdot \big(-\varphi_1(i)\big)$$

$$\vdots$$

$$\theta_{d-1} = \theta_{d-1} - \alpha \cdot \sum_{i=1}^{N} \big(\pi(i) - y(i)\big) \cdot \big(-\varphi_{d-1}(i)\big)$$

}

Logistic Regression Recap

The logistic regression model, despite its name, is not used for regression, but for **classification**

Once the model estimates the probability of a class, we can classify a point to a particular class if the probability for that class is **above a threshold** (usually 0.5)

The function that now we are trying to estimate is: $f(\boldsymbol{\varphi}) = P(y = 1|\boldsymbol{\varphi})$

The logistic regression tries to model f by using the model:

$$s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{\varphi}^\top \boldsymbol{\theta}}}$$

The point $\boldsymbol{\varphi}$ can then be classified to class $y = +1$ if $s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) \geq 0.5$

Logistic Regression Recap

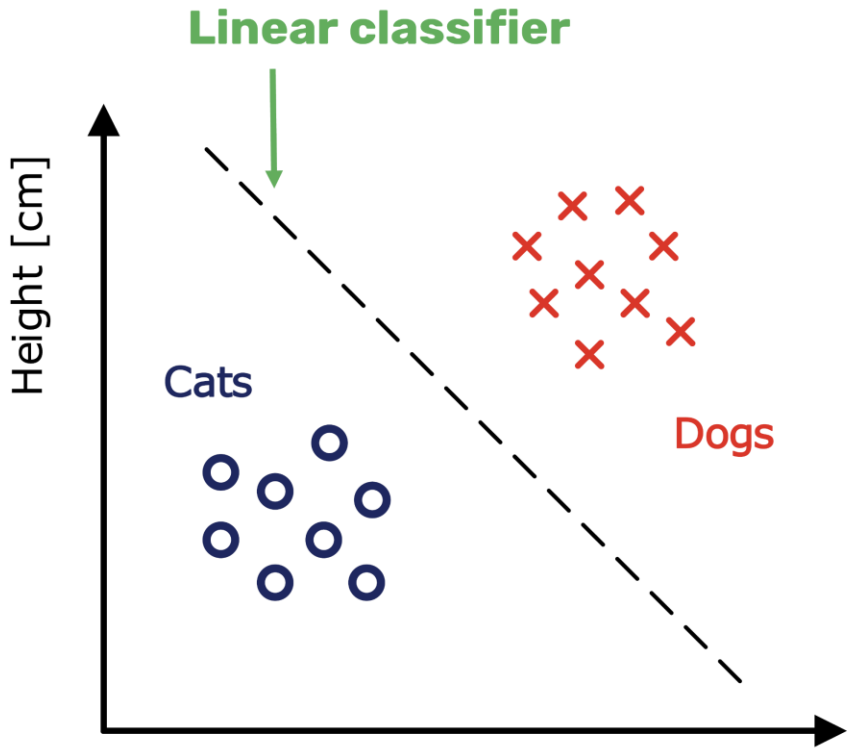
The **classification boundary** found by logistic regression is **linear**

Infact, classifying with the rule:

$$y = 1 \text{ if } \quad s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) \geq 0.5$$

is the same as saying

$$y = 1 \text{ if } \boldsymbol{\varphi}^\top \boldsymbol{\theta} \geq 0$$



Linear Classification

- All we require here is the class boundaries $\{x: \delta_k(x) = \delta_j(x)\}$ be linear for every (k, j) pair
- One can achieve this if $\delta_k(x)$ themselves are linear or any monotone transform of $\delta_k(x)$ is linear
  - An example:

$$P(G=1 \mid X=x) = \frac{\exp(\beta_0 + \beta^T x)}{1+\exp(\beta_0 + \beta^T x)}$$

$$P(G=2 \mid X=x) = \frac{1}{1+\exp(\beta_0 + \beta^T x)}$$

So that $$\log\left[\frac{P(G=1 \mid X=x)}{P(G=2 \mid X=x)}\right] = \beta_0 + \beta^T x$$

Linear

Multiclass Classification

- An input can belong to one of K classes
- Training data: Input associated with class label (a number from 1 to K)
- Prediction: Given a new input, predict the class label

Each input belongs to exactly one class. Not more, not less.

- Otherwise, the problem is not multiclass classification
- If an input can be assigned multiple labels (think tags for emails rather than folders), it is called *multi-label classification*

Binary to Multiclass

- Can we use a binary classifier to construct a multiclass classifier?
  - Decompose the prediction into multiple binary decisions

- How to decompose?
  - One-vs-all (One-vs-rest)
  - One-vs-one (All-vs-all)

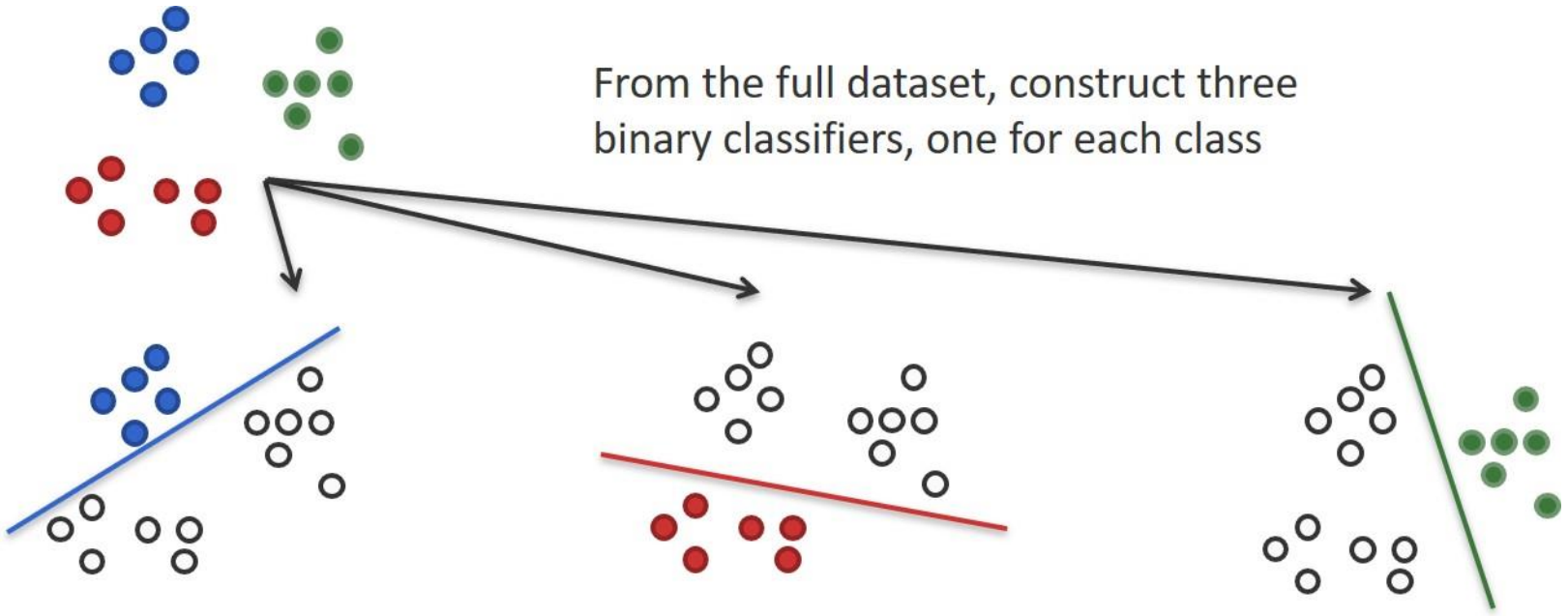
One-vs-all Classification

- Assumption: Each class individually separable from ***all*** the others
- Learning: Given a dataset $D = \{<\mathbf{x}_i, \mathbf{y}_i>\}$,
  - Decompose into K binary classification tasks
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_K$ using any learning algorithm we have seen
- Prediction: "*Winner Takes All*"

$$\text{argmax}_i \ \mathbf{w}_i^T\mathbf{x}$$

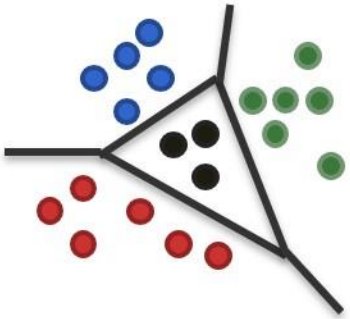
Question: What is the dimensionality of each $\mathbf{w}_i$?

Visualizing One-vs-all



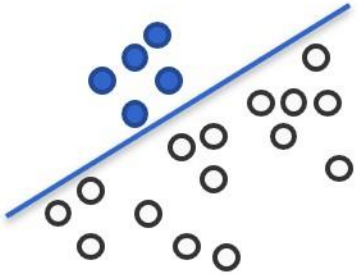
Winner Take All will predict the right answer. Only the correct label will have a positive score

One-vs-all May Not Always Work

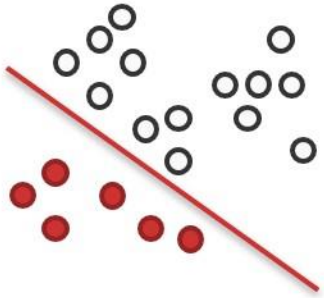


Black points are not separable with a single binary classifier

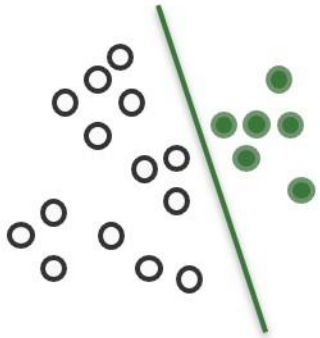
The decomposition will not work for these cases!



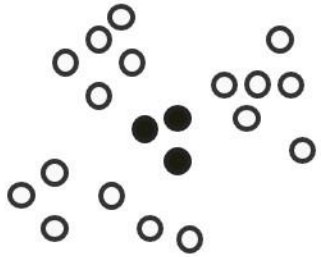
$\mathbf{w}_{blue}^T\mathbf{x} > 0$ for **blue inputs**



$\mathbf{w}_{red}^T\mathbf{x} > 0$ for **red inputs**



$\mathbf{w}_{green}^T\mathbf{x} > 0$ for **green inputs**



???

One-vs-all Classification: Summary

- Easy to learn
  - Use any binary classifier learning algorithm
- Problems
  - No theoretical justification
  - Training set of the binary classifiers may unbalanced
  - Calibration issues
    - We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  - Might not always work
    - Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized

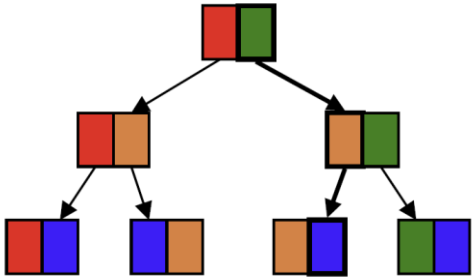
One-vs-one Classification

- Assumption: *Every* pair of classes is separable
- Learning: Given a dataset D = $\{<\mathbf{x}_i, \mathbf{y}_i>\}$,
  - For every pair of labels (j, k), create a binary classifier with:
    - Positive examples: All examples with label j
    - Negative examples: All examples with label k
  - Train $\binom{K}{2} = \frac{K(K-1)}{2}$ classifiers in all

- Prediction: More complex, each label get K-1 votes
  - How to combine the votes? Many methods
    - Majority: Pick the label with maximum votes
    - Organize a tournament between the labels

One-vs-one Classification

Tournament



Majority Vote

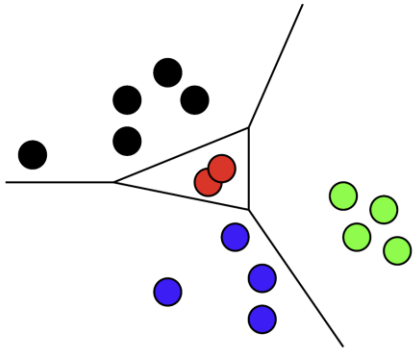
1 red, 2 yellow, 2 green

➔ ?

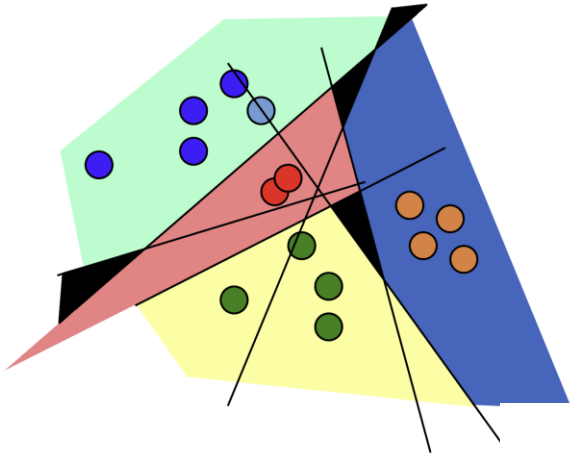
- All are post-learning and *might* cause weird stuff

One-vs-one Assumption

- Every pair of classes is separable

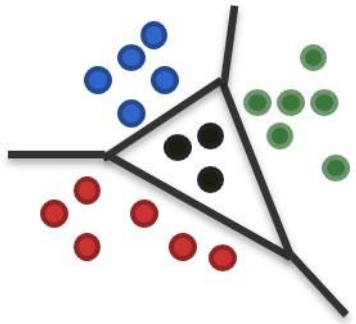


It is possible to separate all k classes with the $O(k^2)$ classifiers



Decision Regions

One-vs-one Classification



- Every pair of labels is linearly separable here
  - When a pair of labels is considered, all others are ignored

- Problems
  - $O(K^2)$ weight vectors to train and store
  - Size of training set for a pair of labels could be very small, leading to overfitting
  - Prediction is often ad-hoc and might be unstable

    Eg: What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?