

CS 2750 Machine Learning Lecture 10

Generative classification model. GLIMS.

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

CS 2750 Machine Learning

Binary classification

- **Two classes** $Y = \{0,1\}$
- Our goal is to learn to classify correctly two types of examples
 - Class 0 – labeled as 0,
 - Class 1 – labeled as 1
- We would like to learn $f : X \rightarrow \{0,1\}$
- **Zero-one error (loss) function**

$$Error_1(\mathbf{x}_i, y_i) = \begin{cases} 1 & f(\mathbf{x}_i, \mathbf{w}) \neq y_i \\ 0 & f(\mathbf{x}_i, \mathbf{w}) = y_i \end{cases}$$

- Error we would like to minimize: $E_{(x,y)}(Error_1(\mathbf{x}, y))$
- **First step:** we need to devise a model of the function

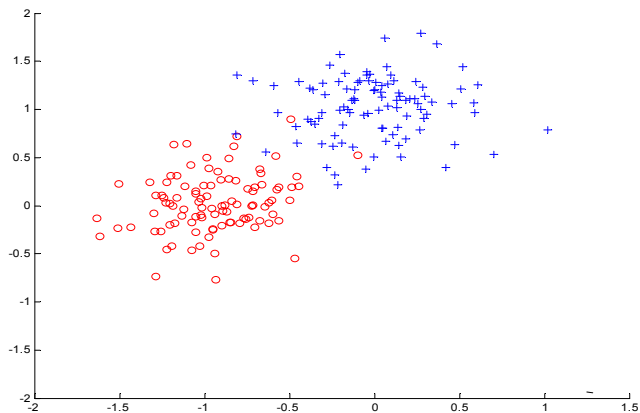
CS 2750 Machine Learning

Discriminant functions

- One convenient way to represent **classifiers** is through
 - **Discriminant functions**
- **Works for binary and multi-way classification**
- **Idea:**
 - For every class $i = 0, 1, \dots, k$ define a function $g_i(\mathbf{x})$ mapping $X \rightarrow \mathcal{R}$
 - When the decision on input \mathbf{x} should be made choose the class with the highest value of $g_i(\mathbf{x})$
- So what happens with the input space? Assume a binary case.

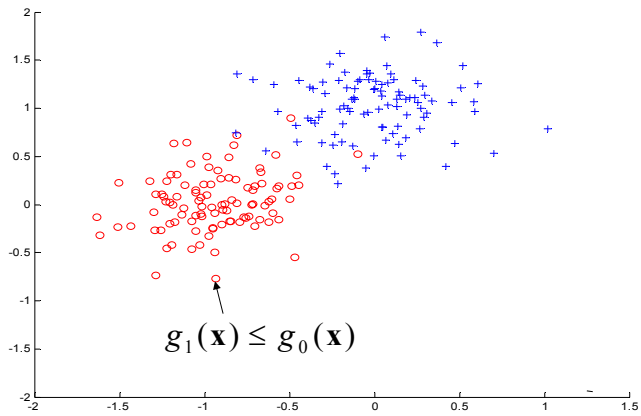
CS 2750 Machine Learning

Discriminant functions



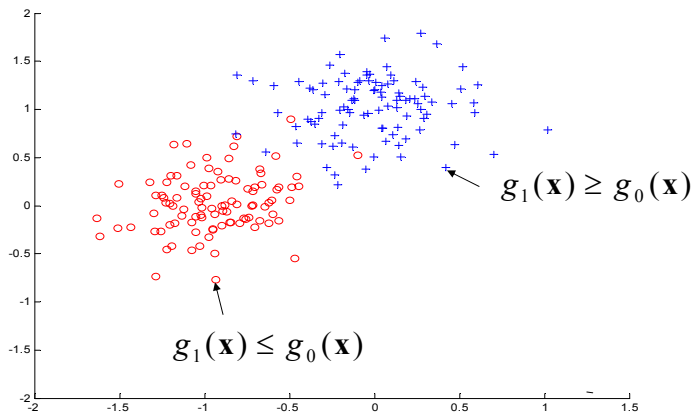
CS 2750 Machine Learning

Discriminant functions



CS 2750 Machine Learning

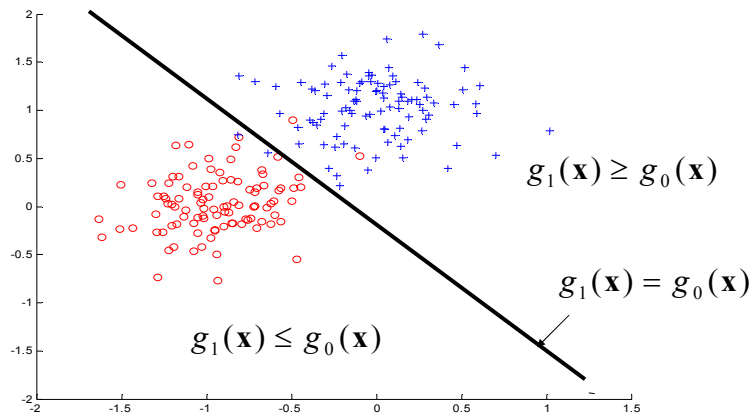
Discriminant functions



CS 2750 Machine Learning

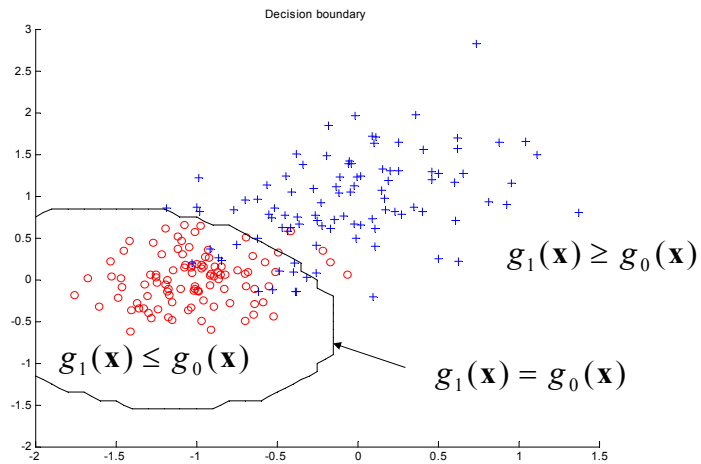
Discriminant functions

- Define **decision boundary**.



CS 2750 Machine Learning

Quadratic decision boundary



CS 2750 Machine Learning

Logistic regression model

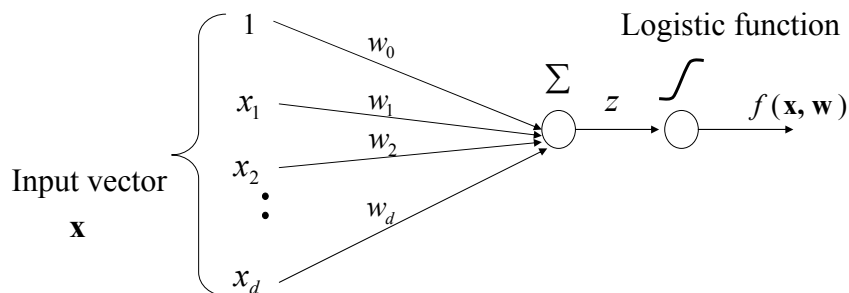
- Defines a linear decision boundary

- Discriminant functions:

$$g_1(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) \quad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x})$$

- where $g(z) = 1/(1 + e^{-z})$ - is a logistic function

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$



CS 2750 Machine Learning

Linear decision boundary

- Logistic regression model defines a **linear decision boundary**
- **Why?**
- **Answer:** Compare two **discriminant functions**.

- **Decision boundary:** $g_1(\mathbf{x}) = g_0(\mathbf{x})$

- For the boundary it must hold:

$$\log \frac{g_0(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T \mathbf{x})}{g(\mathbf{w}^T \mathbf{x})} = 0$$

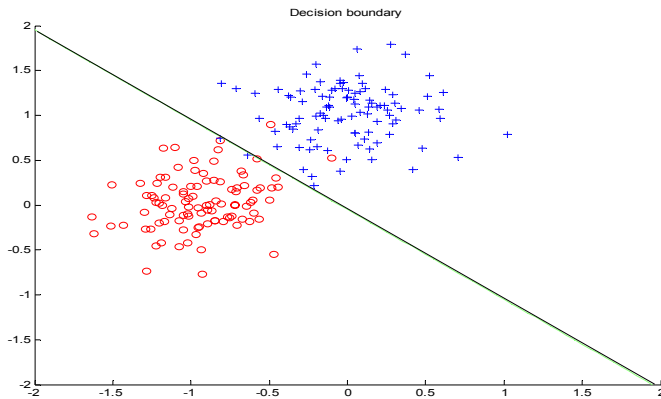
$$\log \frac{g_0(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{\frac{\exp(-(\mathbf{w}^T \mathbf{x}))}{1 + \exp(-(\mathbf{w}^T \mathbf{x}))}}{1} = \log \exp(-(\mathbf{w}^T \mathbf{x})) = \mathbf{w}^T \mathbf{x} = 0$$

CS 2750 Machine Learning

Logistic regression model. Decision boundary

- LR defines a linear decision boundary

Example: 2 classes (blue and red points)



CS 2750 Machine Learning

Generative approach to classification

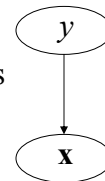
Idea:

1. Represent and learn the distribution $p(\mathbf{x}, y)$
2. Use it to define probabilistic discriminant functions

E.g. $g_0(\mathbf{x}) = p(y = 0 | \mathbf{x})$ $g_1(\mathbf{x}) = p(y = 1 | \mathbf{x})$

Typical model $p(\mathbf{x}, y) = p(\mathbf{x} | y)p(y)$

- $p(\mathbf{x} | y) =$ **Class-conditional distributions (densities)**
binary classification: two class-conditional distributions
 $p(\mathbf{x} | y = 0)$ $p(\mathbf{x} | y = 1)$
- $p(y) =$ **Priors on classes** - probability of class y
binary classification: Bernoulli distribution



$$p(y = 0) + p(y = 1) = 1$$

CS 2750 Machine Learning

Generative approach to classification

Example:

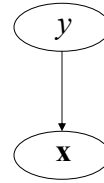
- **Class-conditional distributions**
 - **multivariate normal distributions**

$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for } y = 0$$

$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for } y = 1$$

Multivariate normal $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$



- **Priors on classes (class 0,1)** $y \sim \text{Bernoulli}$

- **Bernoulli distribution**

$$p(y, \theta) = \theta^y (1 - \theta)^{1-y} \quad y \in \{0, 1\}$$

CS 2750 Machine Learning

Learning of parameters of the model

Density estimation in statistics

- We see examples – we do not know the parameters of Gaussians (class-conditional densities)

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- **ML estimate of parameters** of a multivariate normal $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a set of n examples of \mathbf{x}

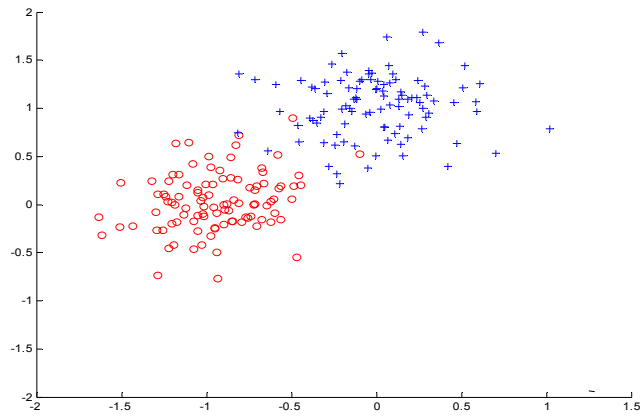
Optimize log-likelihood: $l(D, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^n p(\mathbf{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

- How about **class priors**?

CS 2750 Machine Learning

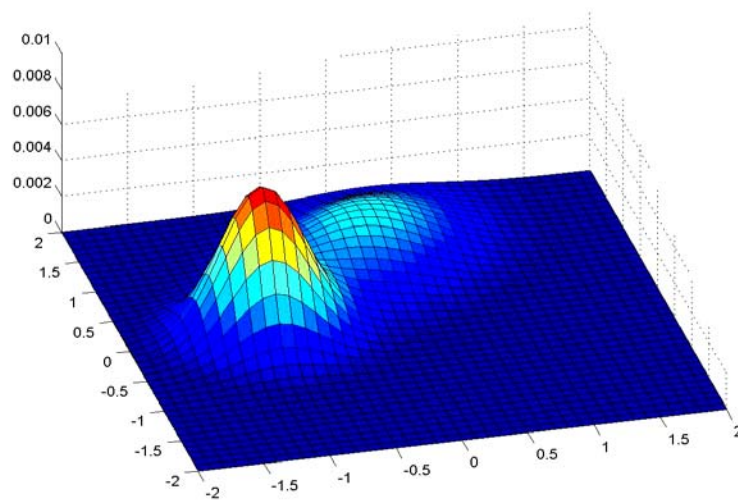
Generative model



CS 2750 Machine Learning

2 Gaussian class-conditional densities

Class conditional densities



CS 2750 Machine Learning

Making class decision

Basically we need to design discriminant functions

Two possible choices:

- **Likelihood of data** – choose the class (Gaussian) that explains the input data (\mathbf{x}) better (likelihood of the data)

$$\underbrace{p(\mathbf{x} | \mu_1, \Sigma_1)}_{g_1(\mathbf{x})} > \underbrace{p(\mathbf{x} | \mu_0, \Sigma_0)}_{g_0(\mathbf{x})} \quad \longrightarrow \quad \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

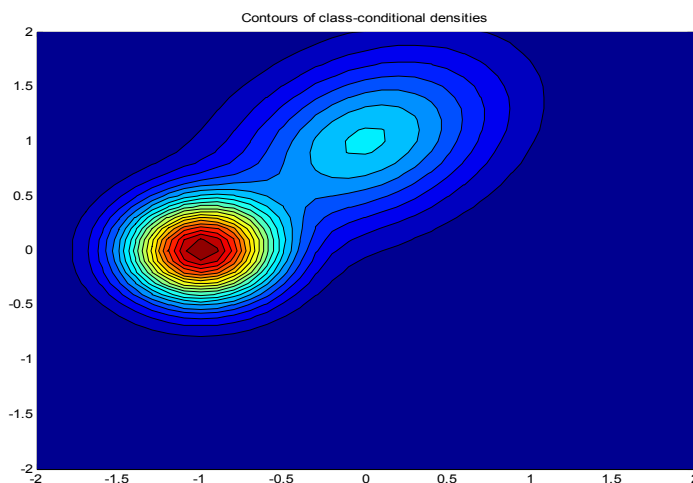
- **Posterior of a class** – choose the class with better posterior probability

$$p(y = 1 | \mathbf{x}) > p(y = 0 | \mathbf{x}) \quad \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

$$p(y = 1 | \mathbf{x}) = \frac{p(\mathbf{x} | \mu_1, \Sigma_1) p(y = 1)}{p(\mathbf{x} | \mu_0, \Sigma_0) p(y = 0) + p(\mathbf{x} | \mu_1, \Sigma_1) p(y = 1)}$$

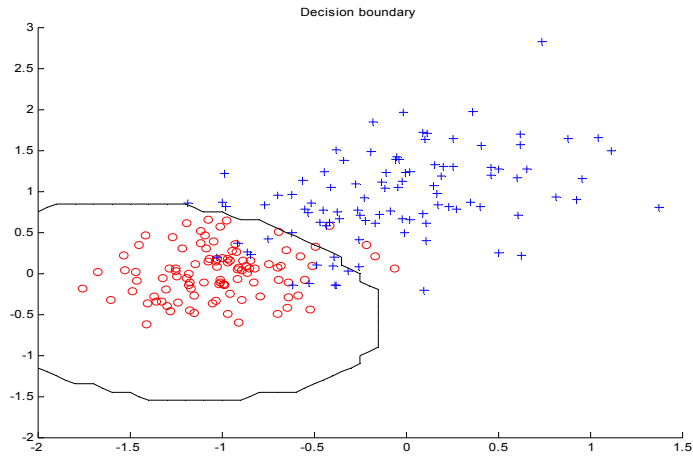
CS 2750 Machine Learning

2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

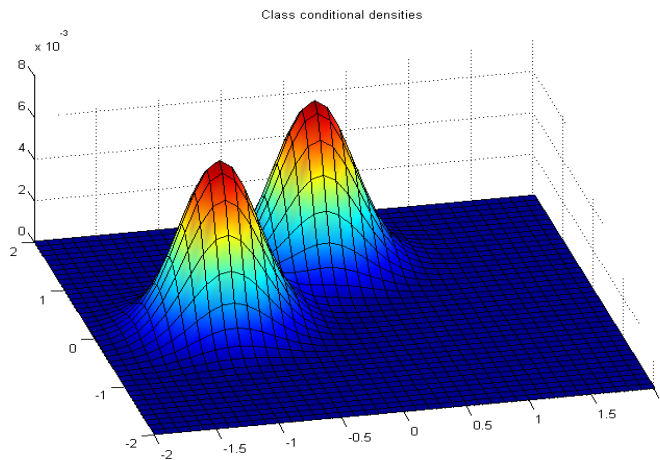
2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

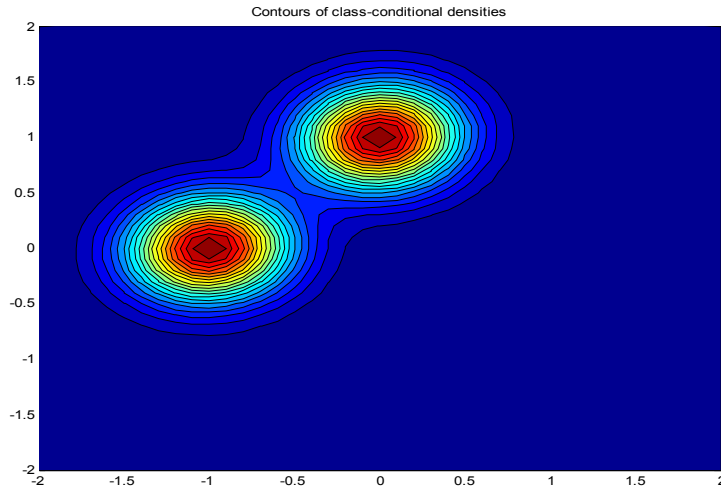
2 Gaussians: Linear decision boundary

- When covariances are the same $\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}), y = 0$
 $\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}), y = 1$



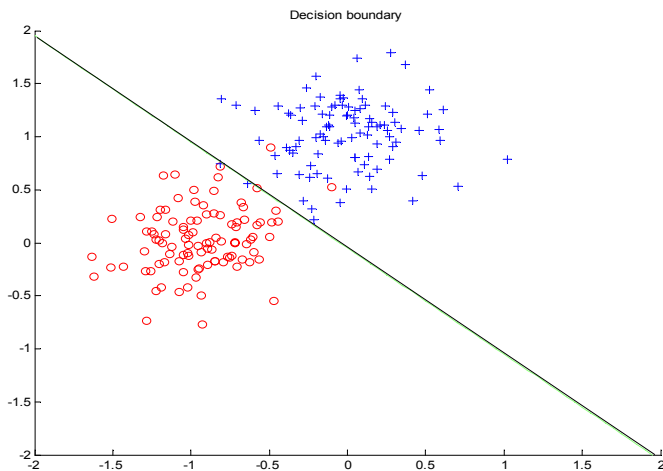
CS 2750 Machine Learning

2 Gaussians: Linear decision boundary



CS 2750 Machine Learning

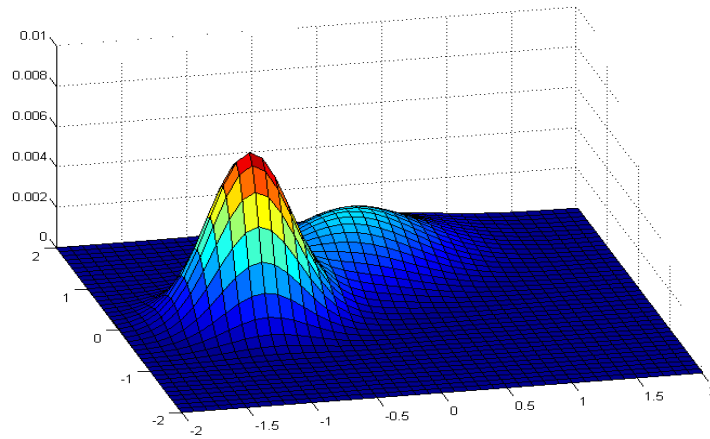
2 Gaussians: linear decision boundary



CS 2750 Machine Learning

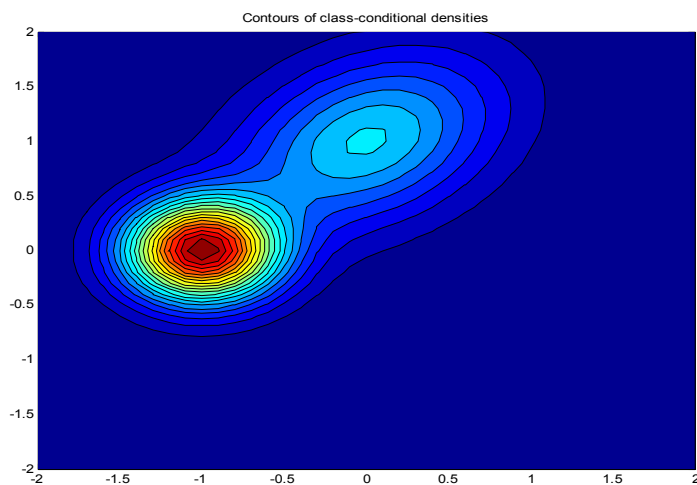
2 Gaussians: Quadratic decision boundary

- When different covariances $\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_1), y = 0$
 $\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_2), y = 1$



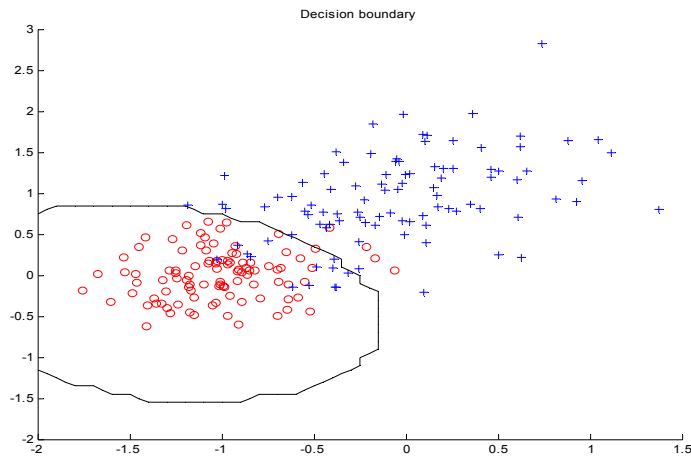
CS 2750 Machine Learning

2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

2 Gaussians: Quadratic decision boundary



CS 2750 Machine Learning

Generative approach to classification

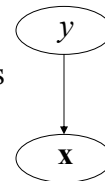
Idea:

1. Represent and learn the distribution $p(\mathbf{x}, y)$
2. Use it to define probabilistic discriminant functions

E.g. $g_0(\mathbf{x}) = p(y = 0 | \mathbf{x})$ $g_1(\mathbf{x}) = p(y = 1 | \mathbf{x})$

Typical model $p(\mathbf{x}, y) = p(\mathbf{x} | y)p(y)$

- $p(\mathbf{x} | y) =$ **Class-conditional distributions (densities)**
binary classification: two class-conditional distributions
 $p(\mathbf{x} | y = 0)$ $p(\mathbf{x} | y = 1)$
- $p(y) =$ **Priors on classes** - probability of class y
binary classification: Bernoulli distribution

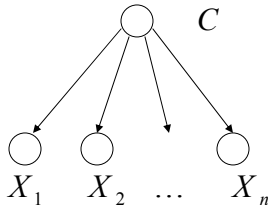


$$p(y = 0) + p(y = 1) = 1$$

CS 2750 Machine Learning

Naïve Bayes classifier

- a generative classifier model with an additional simplifying assumption:
 - All input attributes are conditionally independent of each other given the class. So we have:



$$p(\mathbf{x}, y) = p(\mathbf{x} | y) p(y)$$

$$p(\mathbf{x} | y) = \prod_{i=1}^N p(x_i | y)$$

Learning of parameters of the model

Much simpler density estimation problems

- We need to learn:
$$p(\mathbf{x} | y = 0) \quad \text{and} \quad p(\mathbf{x} | y = 1) \quad \text{and} \quad p(y)$$
- Because of the assumption of the conditional independence we need to learn:
$$\text{for every variable } i: \quad p(x_i | y = 0) \quad \text{and} \quad p(x_i | y = 1)$$
- **If the number of input attributes is large this much easier**
- **Also, the model gives us a flexibility to represent input attributes different of different forms !!!**
- E.g. one attribute can be modeled using the Bernoulli, the other as Gaussian density, or as a Poisson distribution

Making a class decision for the Naïve Bayes

Discriminant functions.

- **Likelihood of data** – choose the class that explains the input data (\mathbf{x}) better (likelihood of the data)

$$\underbrace{\prod_{i=1}^N p(x_i | \Theta_{1,i})}_{g_1(\mathbf{x})} > \underbrace{\prod_{i=1}^N p(x_i | \Theta_{2,i})}_{g_0(\mathbf{x})} \implies \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$

- **Posterior of a class** – choose the class with better posterior probability $p(y = 1 | \mathbf{x}) > p(y = 0 | \mathbf{x})$ then $y=1$
else $y=0$

$$p(y = 1 | \mathbf{x}) = \frac{\left(\prod_{i=1}^N p(x_i | \Theta_{1,i}) \right) p(y = 1)}{\left(\prod_{i=1}^N p(x_i | \Theta_{1,i}) \right) p(y = 0) + \left(\prod_{i=1}^N p(x_i | \Theta_{2,i}) \right) p(y = 1)}$$

CS 2750 Machine Learning

Back to logistic regression

- **Two models with linear decision boundaries:**
 - Logistic regression
 - Generative model with 2 Gaussians with the same covariance matrices

$$x \sim N(\mu_0, \Sigma) \quad \text{for } y = 0$$

$$x \sim N(\mu_1, \Sigma) \quad \text{for } y = 1$$

- **Two models are related !!!**
 - When we have **2 Gaussians with the same covariance matrices** the probability of y given \mathbf{x} has the form of a logistic regression model !!!

$$p(y = 1 | \mathbf{x}, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = g(\mathbf{w}^T \mathbf{x})$$

CS 2750 Machine Learning

When is the logistic regression model correct?

- Members of the exponential family can be often more naturally described as

$$f(\mathbf{x} | \boldsymbol{\theta}, \boldsymbol{\varphi}) = h(x, \boldsymbol{\varphi}) \exp \left\{ \frac{\boldsymbol{\theta}^T \mathbf{x} - A(\boldsymbol{\theta})}{a(\boldsymbol{\varphi})} \right\}$$

$\boldsymbol{\theta}$ - A location parameter $\boldsymbol{\varphi}$ - A scale parameter

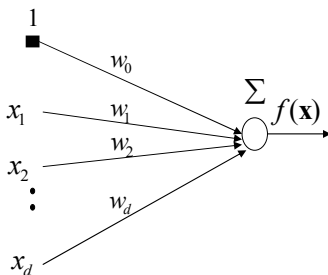
- Claim:** A logistic regression is a correct model when class conditional densities are from the same distribution in the exponential family and have **the same scale factor** $\boldsymbol{\varphi}$
- Very powerful result !!!!**
 - We can represent posteriors of many distributions with the same small network

CS 2750 Machine Learning

Linear units

Linear regression

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$



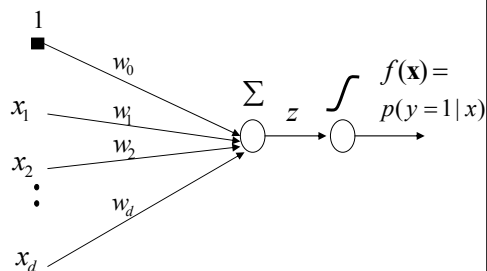
Gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^n (y_i - f(\mathbf{x}_i)) \mathbf{x}_i$$

$$\text{Online: } \mathbf{w} \leftarrow \mathbf{w} + \alpha (y - f(\mathbf{x})) \mathbf{x}$$

Logistic regression

$$f(\mathbf{x}) = p(y=1 | \mathbf{x}, \mathbf{w}) = g(\mathbf{w}^T \mathbf{x})$$



Gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^n (y_i - f(\mathbf{x}_i)) \mathbf{x}_i$$

$$\text{Online: } \mathbf{w} \leftarrow \mathbf{w} + \alpha (y - f(\mathbf{x})) \mathbf{x}$$

The same

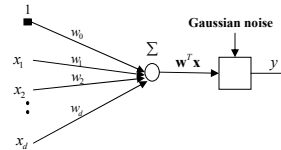
CS 2750 Machine Learning

Gradient-based learning

- The **same simple gradient update rule** derived for both the linear and logistic regression models
- Where the magic comes from?
- Under the **log-likelihood** measure the function models and the models for the output selection fit together:

– **Linear model + Gaussian noise**

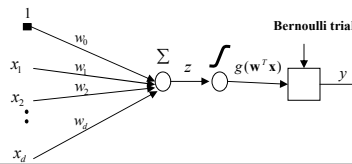
$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$$



– **Logistic + Bernoulli**

$$y = \text{Bernoulli}(\theta)$$

$$\theta = p(y = 1 | \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$



CS 2750 Machine Learning

Generalized linear models (GLIM)

Assumptions:

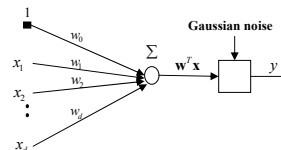
- The conditional mean (expectation) is:

$$\mu = f(\mathbf{w}^T \mathbf{x})$$
 - Where $f(\cdot)$ is a **response function**
- Output y is characterized by an exponential family distribution with a conditional mean μ

Examples:

– **Linear model + Gaussian noise**

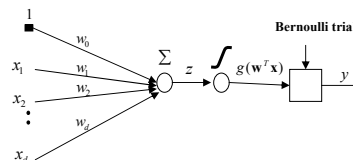
$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$$



– **Logistic + Bernoulli**

$$y \approx \text{Bernoulli}(\theta)$$

$$\theta = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$



CS 2750 Machine Learning

Generalized linear models

- A canonical response functions $f(\cdot)$:
 - encoded in the distribution

$$p(\mathbf{x} | \boldsymbol{\theta}, \boldsymbol{\varphi}) = h(x, \boldsymbol{\varphi}) \exp \left\{ \frac{\boldsymbol{\theta}^T \mathbf{x} - A(\boldsymbol{\theta})}{a(\boldsymbol{\varphi})} \right\}$$

- Leads to a simple gradient form
- Example: Bernoulli distribution

$$p(x | \mu) = \mu^x (1 - \mu)^{1-x} = \exp \left\{ \log \left(\frac{\mu}{1 - \mu} \right) x + \log(1 - \mu) \right\}$$

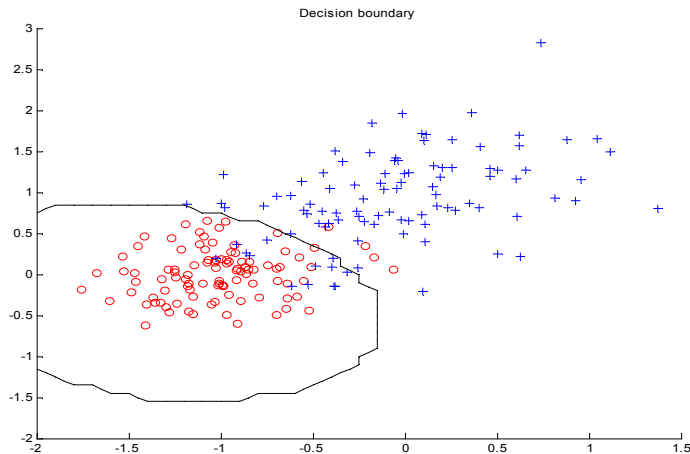
$$\theta = \log \left(\frac{\mu}{1 - \mu} \right) \quad \mu = \frac{1}{1 + e^{-\theta}}$$

- Logistic function matches the Bernoulli

CS 2750 Machine Learning

When does the logistic regression fail?

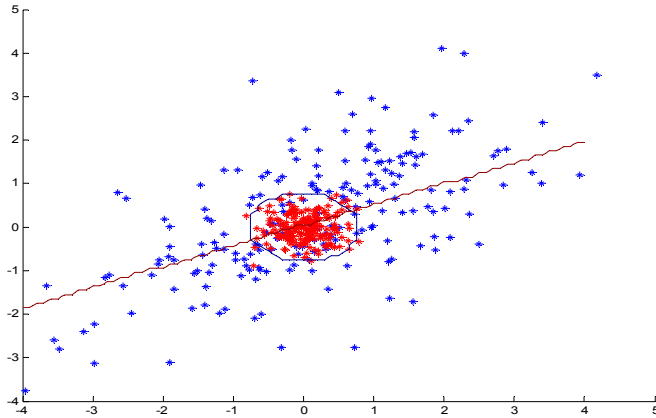
- Quadratic decision boundary is needed



CS 2750 Machine Learning

When does the logistic regression fail?

- Another example of a non-linear decision boundary



CS 2750 Machine Learning

Non-linear extension of logistic regression

- use **feature (basis) functions** to model **nonlinearities**
 - the same trick as used for the linear regression

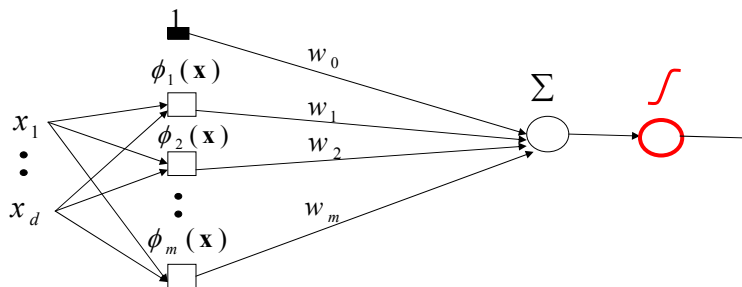
Linear regression

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^m w_j \phi_j(\mathbf{x})$$

Logistic regression

$$f(\mathbf{x}) = g(w_0 + \sum_{j=1}^m w_j \phi_j(\mathbf{x}))$$

$\phi_j(\mathbf{x})$ - an arbitrary function of \mathbf{x}



CS 2750 Machine Learning