第四章 基本分类方法

吴洪
电子科技大学计算机学院

*With thanks to Bishop, Hauskrecht, Urtasun and Zemel for use of their figures and slides

2020年春季

Outline

- Basic of Linear Classification Models
- Logistic Regresion
- Generative Model and Naïve Bayes
- KNN
- Evaluation of Classification

Regression vs Classification

- In *Regression* we assign input vector x to one or more continuous target variables t
  - Linear regression has simple analytical and computational properties
- In *Classification* we assign input vector x to one of K discrete classes C_k, $k = 1, \ldots, K$.
  - We discuss here linear models for Classification
  - Ordinal Regression is a form of classification where discrete classes have an ordering
    - E.g., relevance score regression

Different Approaches to Classification

We can divide the large variety of classification approaches into roughly three main types:

1. Discriminative function

   - directly estimate a decision rule/boundary - e.g., logistic regression, decision tree, svm

2. Probabilistic Models: Generative/Discriminative

   - build a generative statistical model - e.g., Naïve Bayes

3. Instance based classifiers (Nonparametric Method)

   - Use observation directly (no models) - e.g. K nearest neighbors

From Regression to Classification

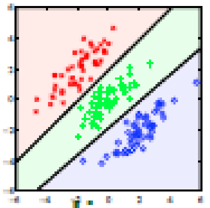
- Linear Regression model $y(\boldsymbol{x},\boldsymbol{w})$ is a linear function of parameters $\boldsymbol{w}$
  - In simple case model is also a linear function of $\boldsymbol{x}$
    - Thus has the form $y(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0$ where y is a real no.
- For classification we need need to predict class labels or posterior probabilities in range $(0,1)$
  - For this, we use a generalization where we transform the linear function of w using a nonlinear function $f(.)$, so that

$$y(\boldsymbol{x})=f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0\right)$$

- $f(.)$ is known as an *activation function*

Linear Classification Models

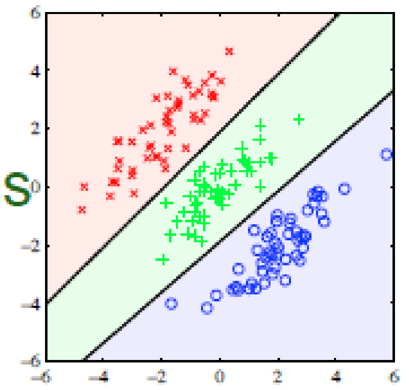
- Decision surfaces of $y(\boldsymbol{x}) = f(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0)$ correspond to $y(\boldsymbol{x}) = \text{constant}$ or $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = \text{constant}$



  - Surfaces are linear in $\boldsymbol{x}$ even if $f(.)$ is nonlinear
    - For this reason they are called *generalized linear models*
  - However no longer linear in parameters $\boldsymbol{w}$ due to presence of $f(.)$, therefore:
    - More complex models for classification than regression
- Linear classification algorithms we discuss are applicable even if we transform $\boldsymbol{x}$ using a vector of basis functions $\phi(\boldsymbol{x})$

Linear Classification Models

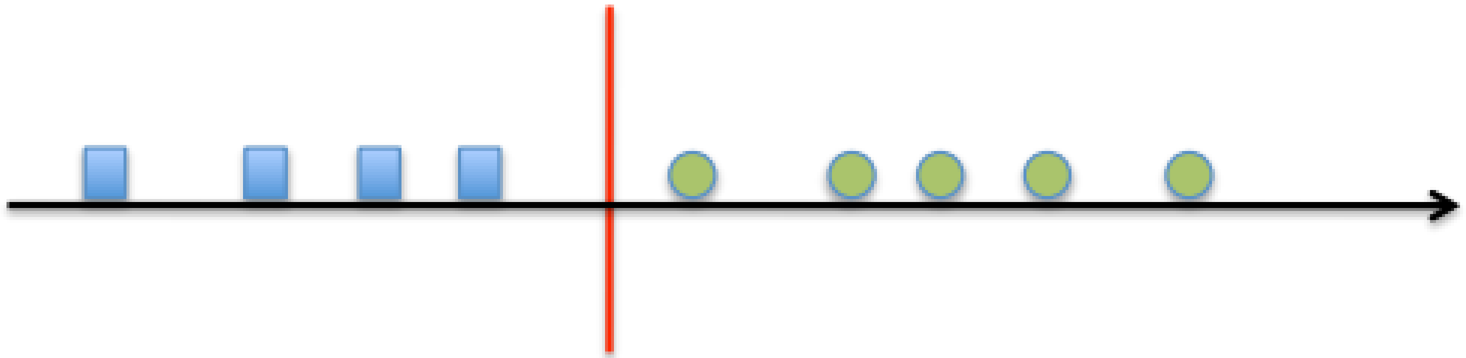
- Common classification scenario: classes considered disjoint
  - Each input assigned to only one class
- Input space divided into decision regions
- Decision surfaces are linear functions of input x
  - Defined by $(D-1)$ dimensional hyperplanes within D dim. input space



Straight line is1-D in 2-D
A plane is 2-D in 3-D

Data sets whose classes can be separated exactly by linear decision surfaces are said to be Linearly separable

Example in 1D

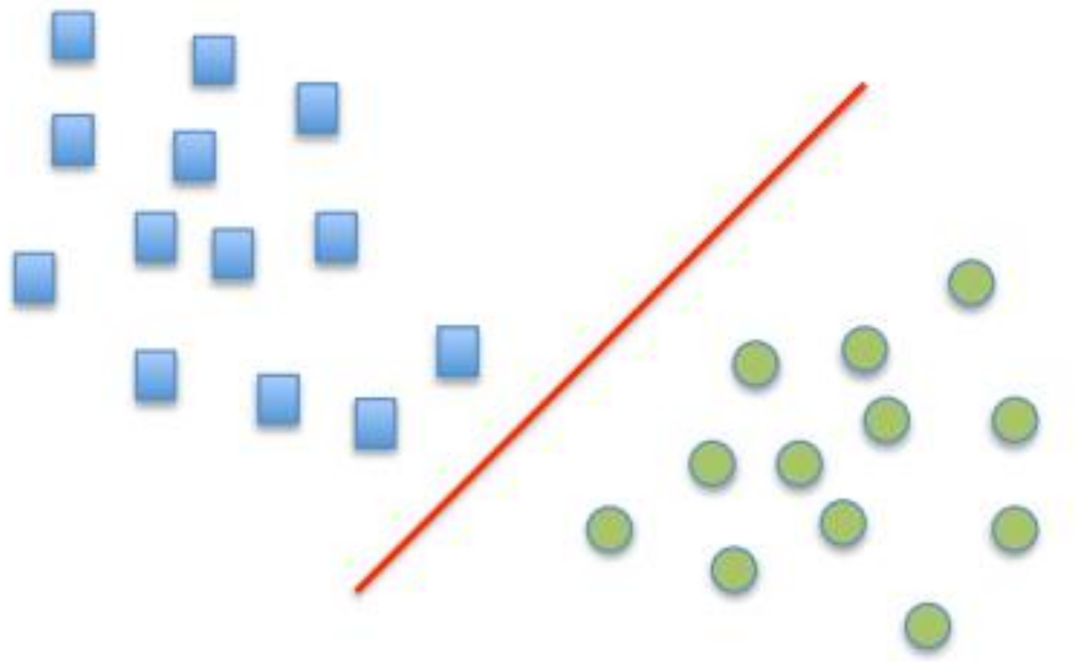


- The **linear classifier** has a **linear boundary (hyperplane)**

$$w_0 + \mathbf{w}^T\mathbf{x} = 0$$

which separates the space into two "half-spaces"

Example in 2D



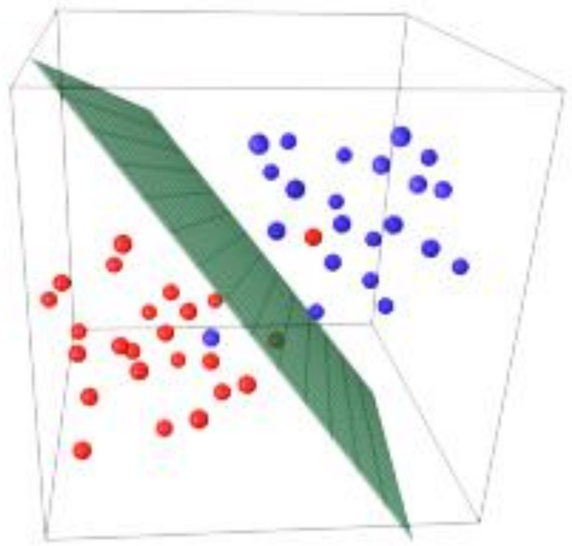
- The **linear classifier** has a **linear boundary (hyperplane)**

$$w_0 + \mathbf{w}^T\mathbf{x} = 0$$

which separates the space into two "half-spaces"

- In 2D this is a line

Example in 3D



- The **linear classifier** has a **linear boundary (hyperplane)**

$$w_0 + \mathbf{w}^T\mathbf{x} = 0$$

  which separates the space into two "half-spaces"
- In 3D this is a plane

Geometry of Linear Discriminant Functions

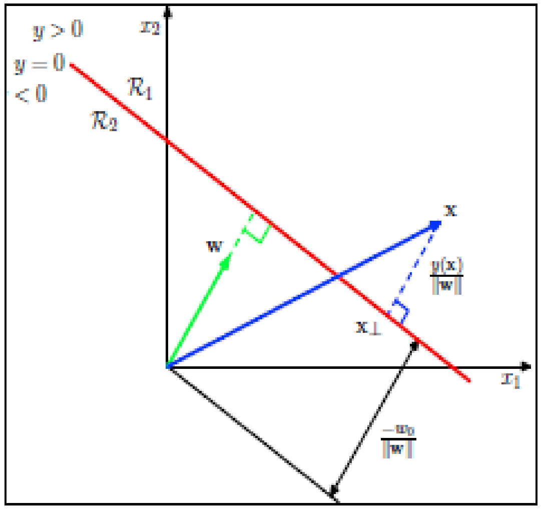
- Two-class linear discriminant function:

$$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0$$

A linear function of input vector

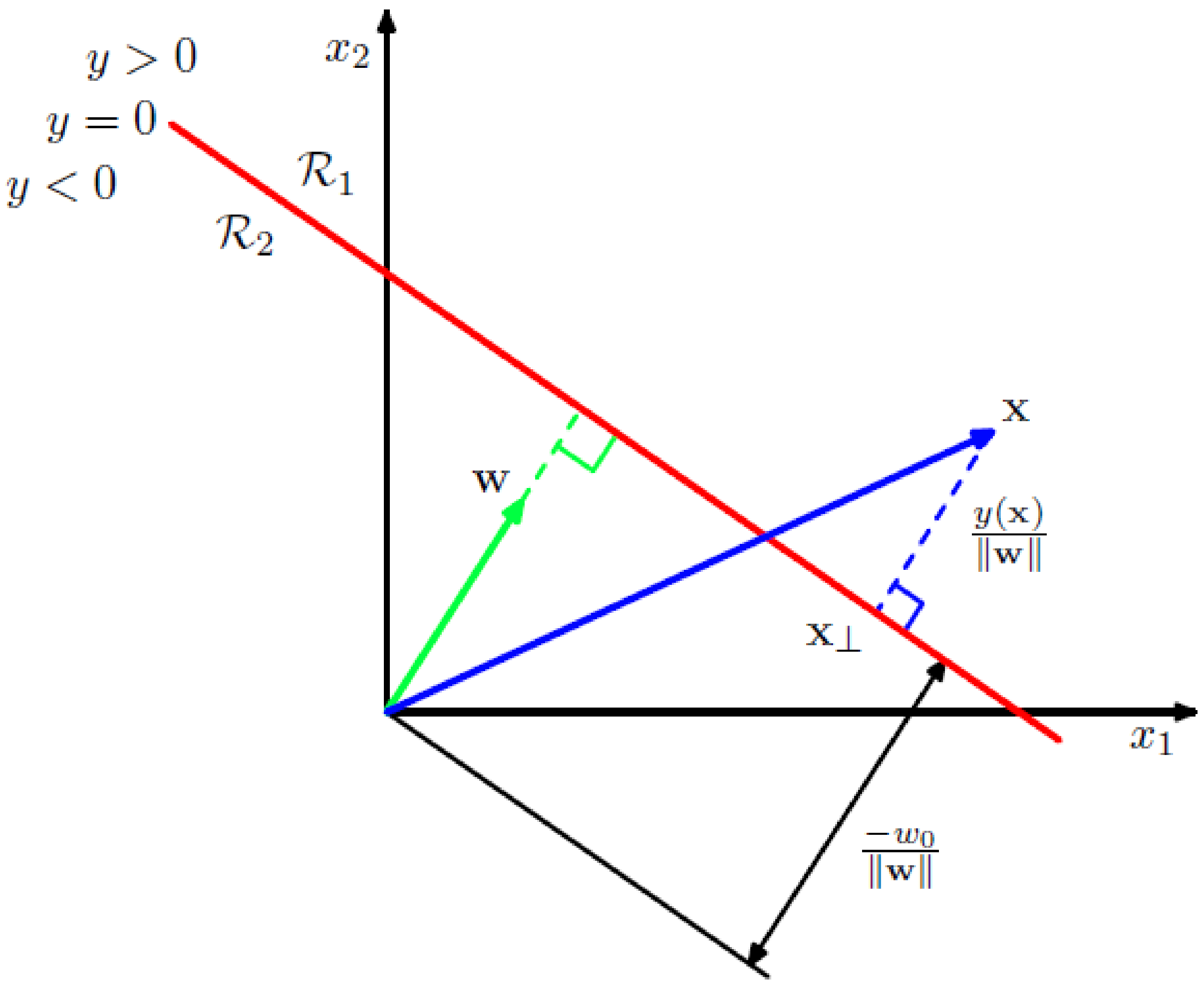
$\boldsymbol{w}$ is weight vector and w_0 is *bias*

Negative of bias sometimes called *threshold*



- Assign $\boldsymbol{x}$ to C_1 if $y(\boldsymbol{x}) \geq 0$ else C_2
  - Defines decision boundary $y(\boldsymbol{x}) = 0$
    - It corresponds to a $(D\text{-}1)$- dimensional hyperplane in a D-dimensional input space

Geometry



$y > 0$
$y = 0$
$y < 0$
x_2
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathbf{x}$
$\mathbf{w}$
$\frac{y(\mathbf{x})}{\|\mathbf{w}\|}$
$\mathbf{x}_\perp$
x_1
$\frac{-w_0}{\|\mathbf{w}\|}$

Distance of Origin to Surface is w_0

Let x_A and x_B be points on surface $y(x)=w^T x + w_0=0$

- Because $y(x_A)=y(x_B)=0$, we have $w^T(x_A-x_B)=0$,
- Thus w is orthogonal to every vector lying on decision surface
- So w determines orientation of the decision surface

– If x is a point on surface then $y(x)=0$ or $w^T x = -w_0$

- Normalized distance from origin to surface:

$$\frac{\mathrm{w}^T\mathrm{x}}{||\mathrm{w}||} = -\frac{w_0}{||\mathrm{w}||}$$

where $||\mathrm{w}||$ is the norm defined as

$$||\mathrm{w}||^2 = \mathrm{w}^T\mathrm{w} = w_1^2 + .. + w_{M-1}^2$$

  – Where elements of w are normalized by dividing by its norm $||x||$
    » By definition of a normalized vector which has length 1

– w_0 sets distance of origin to surface

Distance of arbitrary point x to surface

- Let x be an arbitrary point
  - We can show that $y(x)$ gives signed measure of perpendicular distance r from x to surface as follows:
    - If x_p is orthogonal projection of x to surface then

$x = x_p + r\dfrac{\mathrm{w}}{||\,\mathrm{w}\,||}$ by vector addition

Second term is a vector normal to surface.

This vector is parallel to w

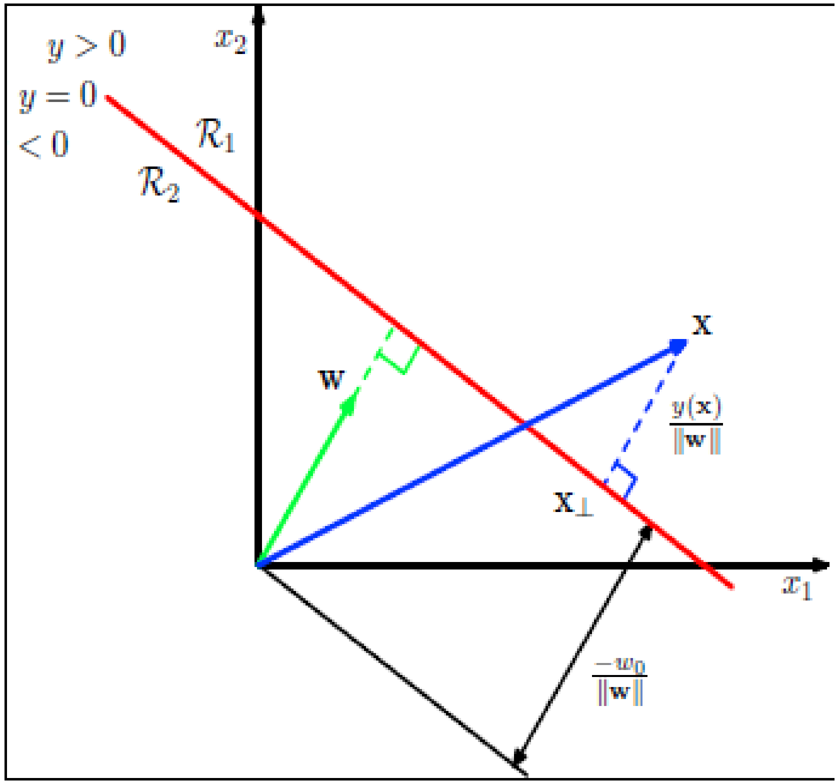
which is normalized by length ||w|| .

Since a normalized vector has length 1

we need to scale by r.

From which we can get

$r = \dfrac{y(\mathrm{x})}{||\,\mathrm{w}\,||}$



Decision rule for linear classifier

- A reasonable **decision rule** is

$$y = \begin{cases} 1 & w_o + \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- How can I mathematically write this rule?

$$y = \text{sign}(w_0 + \mathbf{w}^T\mathbf{x})$$

- How does this function look like?

step function

Outline

Classification as Regression

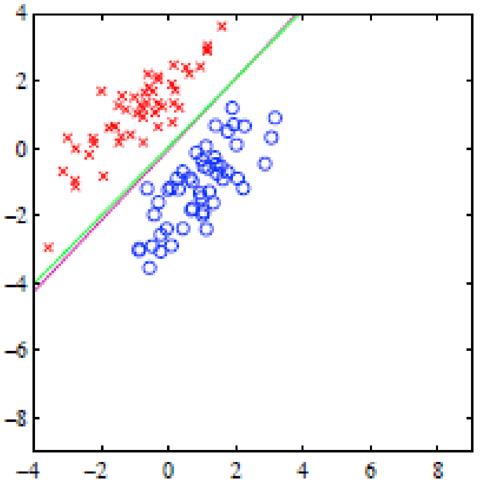
- Can we do this task using what we have learned in the previous lecture?
- Simple hack: Ignore that the input is categorical!
- Suppose we have a binary problem, $t \in \{-1, 1\}$
- Assuming the standard model used for regression

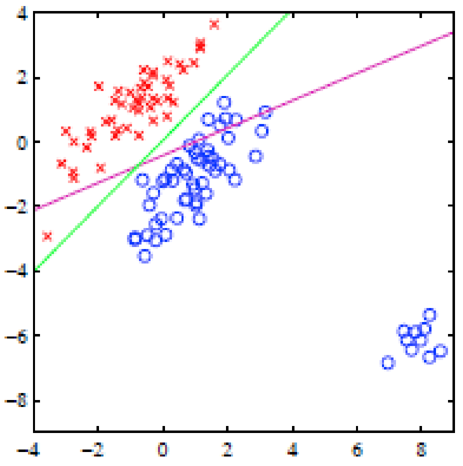
$$y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

- How can we obtain $\mathbf{w}$?
- Use least squares, $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$. How is $\mathbf{X}$ computed? and $\mathbf{t}$?
- Which loss are we minimizing? Does it make sense?

$$\ell_{square}(\mathbf{w}, t) = \frac{1}{N}\sum_{i=1}^{N}(t_i - \mathbf{w}^T\mathbf{x}_i)^2$$

Least Squares is Sensitive to Outliers





Magenta: Least Squares
Green: Logistic Regression (more robust)

Sum of squared errors penalizes predictions that are "too correct"
Or long way from decision boundary

SVMs have an alternate error function (hinge function) that does not have this limitation

Linear Classifier

- The classifier we have looked at is

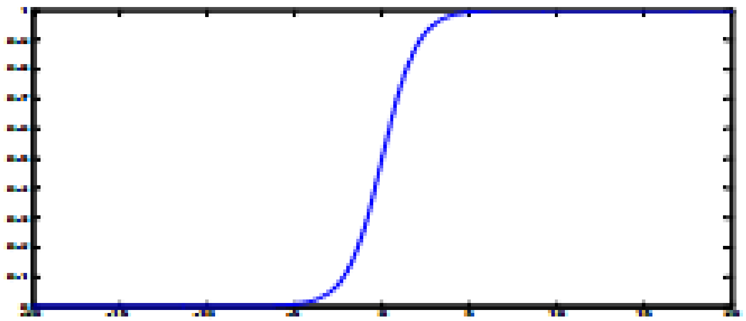
$$y(\mathbf{x}) = \text{sign}(w_0 + \mathbf{w}^T\mathbf{x})$$

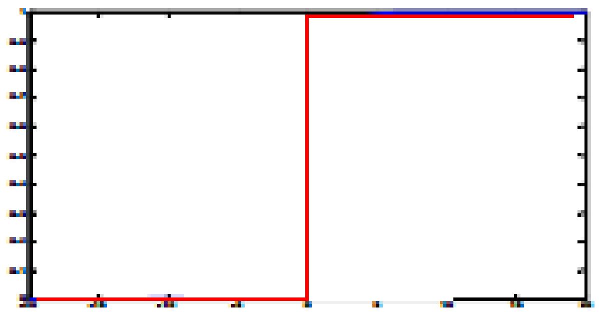
- It was difficult to optimize any loss on $\ell(y, t)$ due to the form of $y(\mathbf{x})$
- Can we have a smoother function such that things become easier to optimize?

Logistic function

Function: $$g(z) = \frac{1}{(1+e^{-z})}$$

- Is also referred to as a **sigmoid function**
- takes a real number and outputs the number in the interval [0,1]
- Models a smooth switching function; replaces hard threshold function


Logistic (smooth) switching


Threshold (hard) switching

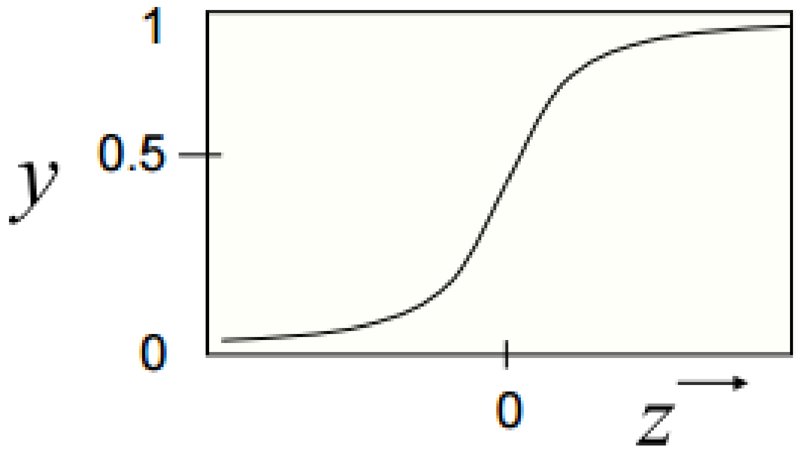
Logistic Regression

- An alternative: replace the *sign*$(\cdot)$ with the **sigmoid** or **logistic function**
- We assumed a particular functional form: sigmoid applied to a linear function of the data

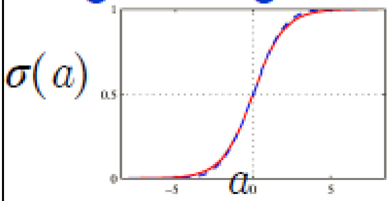
$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

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



Logistic Sigmoid



Properties:

A. Symmetry

$\sigma(-a)=1-\sigma(a)$

B. Inverse

$a=\ln(\sigma/1-\sigma)$

known as *logit*. Also known as *log odds* since it is the ratio

$\ln[p(C_1|\phi)/p(C_2|\phi)]$

C. Derivative

$d\sigma/da = \sigma(1-\sigma)$

- The output is a smooth function of the inputs and the weights

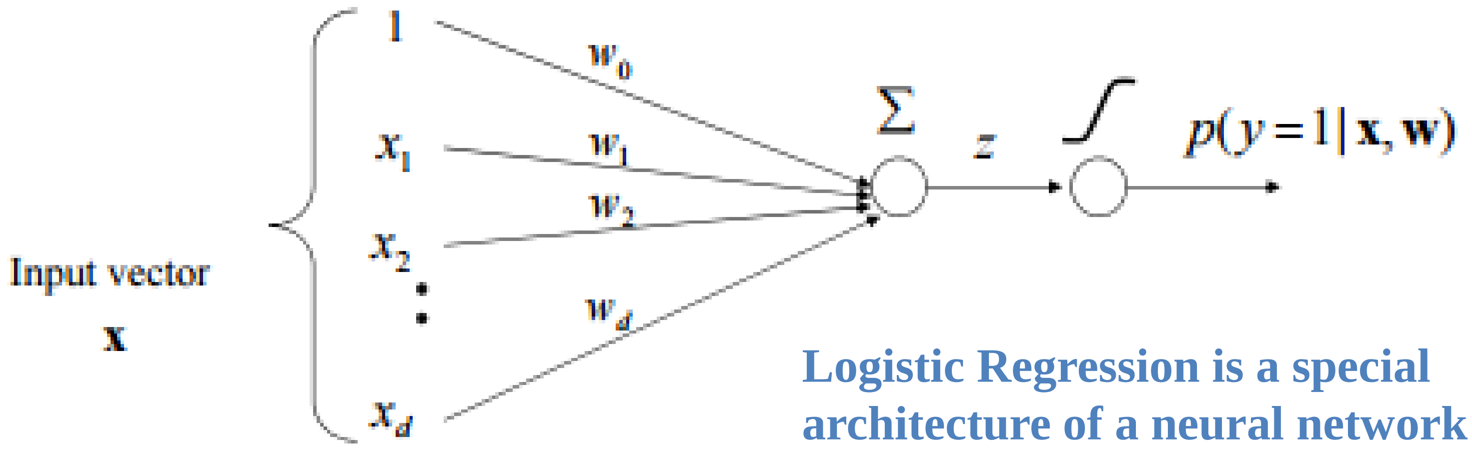
Logistic regression model

- **Discriminant functions:**

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

- **Values of discriminant functions vary in interval [0,1]**
  - **Probabilistic interpretation**

$$f(\mathbf{x},\mathbf{w}) = p(y=1 \mid \mathbf{w},\mathbf{x}) = g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Logistic Regression is a special architecture of a neural network

Logistic regression

- We learn **a probabilistic function**

$$f : X \rightarrow [0,1]$$

  - where f describes the probability of class 1 given $\mathbf{x}$

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = p(y = 1 \mid \mathbf{x}, \mathbf{w})$$

Note that: $$p(y = 0 \mid \mathbf{x}, \mathbf{w}) = 1 - p(y = 1 \mid \mathbf{x}, \mathbf{w})$$

- Making decisions with the logistic regression model:

If $p(y = 1 \mid \mathbf{x}) \geq 1/2$ then choose **1**
Else choose **0**

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:

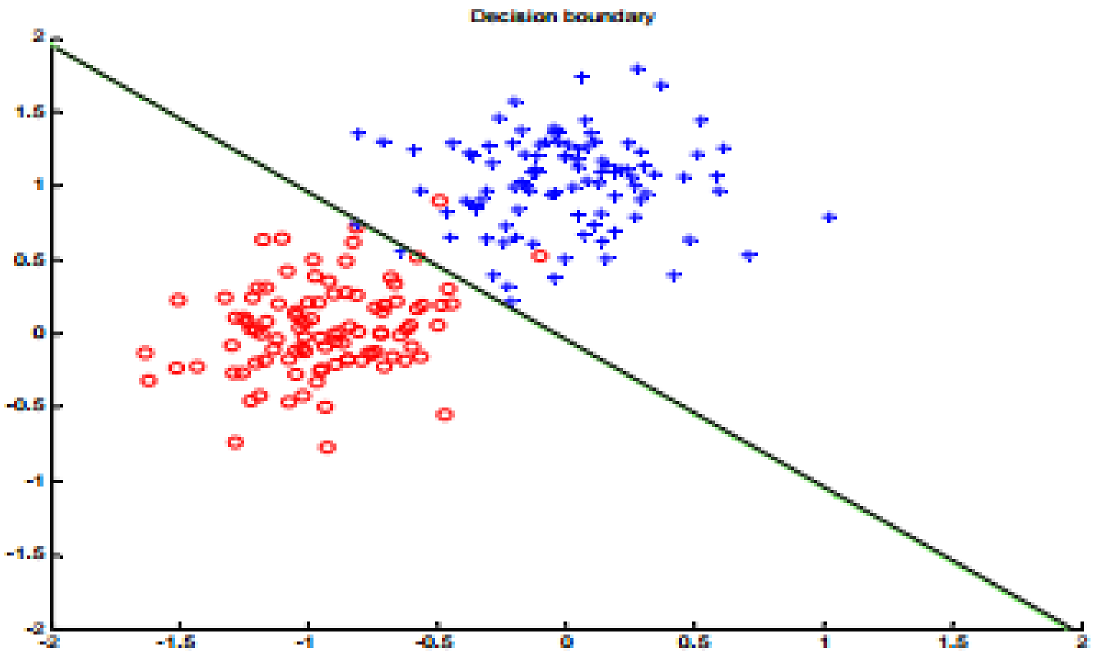
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T\mathbf{x})}{g(\mathbf{w}^T\mathbf{x})} = 0$$

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

Logistic regression model. Decision boundary

- **LR defines a linear decision boundary**

Example: 2 classes (blue and red points)



Logistic regression: parameter learning

Likelihood of outputs

- **Let**

$$D_i = < \mathbf{x}_i, y_i > \qquad \mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$$

- **Then**

$$L(D, \mathbf{w}) = \prod_{i=1}^{n} P(y = y_i \mid \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i}$$

- **Find weights w that maximize the likelihood of outputs**
  - Apply the log-likelihood trick. The optimal weights are the same for both the likelihood and the log-likelihood

$$l(D, \mathbf{w}) = \log \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1-y_i} = \sum_{i=1}^{n} \log \mu_i^{y_i} (1 - \mu_i)^{1-y_i} =$$

$$= \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

Good news: $l(W)$ is concave function of W

Bad news: no closed-form solution to maximize $l(W)$

Derivation of the gradient

- **Log likelihood** $l(D,\mathbf{w}) = \sum_{i=1}^{n} y_i \log \mu_i + (1-y_i)\log(1-\mu_i)$

- **Derivatives of the loglikelihood**

$$\frac{\partial}{\partial w_j} l(D,\mathbf{w}) = \sum_{i=1}^{n} \frac{\partial}{\partial z_i}\left[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right]\frac{\partial z_i}{\partial w_j}$$

$$\frac{\partial z_i}{\partial w_j} = x_{i,j}$$

Derivative of a logistic function

$$\frac{\partial g(z_i)}{\partial z_i} = g(z_i)(1-g(z_i))$$

$$\frac{\partial}{\partial z_i}\left[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right] = y_i \frac{1}{g(z_i)}\frac{\partial g(z_i)}{\partial z_i} + (1-y_i)\frac{-1}{1-g(z_i)}\frac{\partial g(z_i)}{\partial z_i}$$

$$= y_i(1-g(z_i)) + (1-y_i)(-g(z_i)) = \boxed{y_i - g(z_i)}$$

$$\boxed{\nabla_{\mathbf{w}} l(D,\mathbf{w}) = \sum_{i=1}^{n} \mathbf{x}_i(y_i - g(\mathbf{w}^T\mathbf{x}_i)) = \sum_{i=1}^{n} \mathbf{x}_i(y_i - f(\mathbf{w},\mathbf{x}_i))}$$

Logistic regression: parameter learning

- **Notation:** $\mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$
- **Log likelihood**

$$l(D, \mathbf{w}) = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

- **Derivatives of the loglikelihood**

Nonlinear in weights !!

$$-\frac{\partial}{\partial w_j} l(D, \mathbf{w}) = \sum_{i=1}^{n} -x_{i,j}(y_i - g(z_i))$$

$$\nabla_{\mathbf{w}} - l(D, \mathbf{w}) = \sum_{i=1}^{n} -\mathbf{x}_i(y_i - g(\mathbf{w}^T \mathbf{x}_i)) = \sum_{i=1}^{n} -\mathbf{x}_i(y_i - f(\mathbf{w}, \mathbf{x}_i))$$

- **Gradient descent:**

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [-l(D, \mathbf{w})] \big|_{\mathbf{w}^{(k-1)}}$$

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

Logistic regression. Online gradient descent

- **On-line component of the loglikelihood**

$$-J_{\text{online}}(D_i, \mathbf{w}) = y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

- **On-line learning update for weight w** $J_{online}(D_k, \mathbf{w})$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [J_{online}(D_k, \mathbf{w})] |_{\mathbf{w}^{(k-1)}}$$

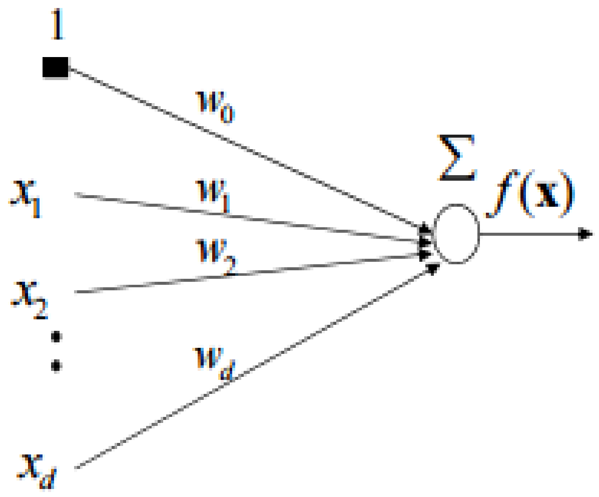
- **ith update for the logistic regression** and $D_k = < \mathbf{x}_k, y_k >$

$$\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k)[y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_k)]\mathbf{x}_k$$

Linear units

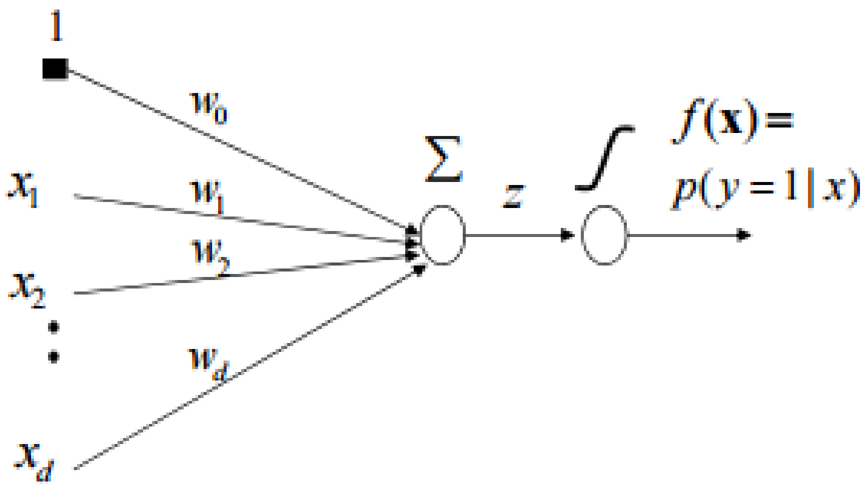
Linear regression

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



Logistic regression

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



Gradient update:

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

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

The same

$$\leftrightarrow$$

Gradient update:

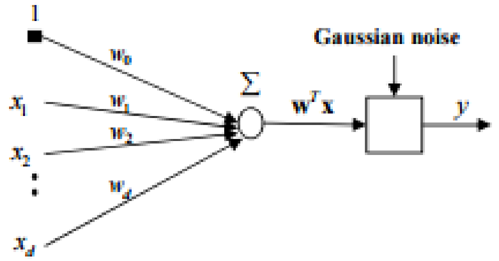
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))\mathbf{x}_i$$

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

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**

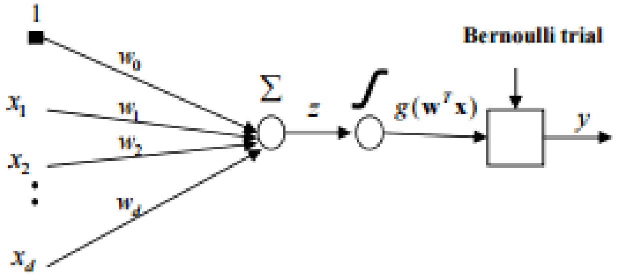
$$y = \mathbf{w}^T\mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2)$$



  - **Logistic + Bernoulli**

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

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



Outline

- Basic of Linear Classification Models
- Logistic Regresion
- Generative Model and Naïve Bayes
- KNN
- Evaluation of Classification

Generative approach to classification

Logistic regression: **model** $p(y \mid \mathbf{x})$

Generative approach:

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

   E.g. $g_o(\mathbf{x}) = p(y = 0 \mid \mathbf{x})$ $\quad g_1(\mathbf{x}) = p(y = 1 \mid \mathbf{x})$

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

- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

  $p(\mathbf{x} \mid y = 0)$ $\quad p(\mathbf{x} \mid y = 1)$

- $p(y)$ = **Priors on classes** - probability of class y

  binary classification: Bernoulli distribution

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

The posterior probability

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

sigmoid function

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

Log odds

Softmax: Generalization of logistic sigmoid

- For $K > 2$, we can use its generalization

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)p(C_j)}$$

$$\boxed{= \frac{\exp(a_k)}{\sum_j \exp(a_j)}}$$

  – Quantities a_k are defined by $a_k = \ln p(\boldsymbol{x} \mid C_k)p(C_k)$

- Known as the *soft-max* function
  – Since it is a smoothed max function
    - If $a_k >> a_j$ for all $j \neq k$ then $p(C_k \mid \boldsymbol{x}) = 1$ and 0 for rest
    - A general technique for finding max of several a_k

Class-conditional densities：Gaussian

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}} \underset{\nwarrow}{\mathbf{\Sigma}^{-1}} (\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

shared covariance matrix

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

Linear decision boundary: logistic regression

ML parameter estimation

a data set $\{\mathbf{x}_n, t_n\}$

the prior class probability $p(\mathcal{C}_1) = \pi$, so that $p(\mathcal{C}_2) = 1 - \pi$. For a data point $\mathbf{x}_n$ from class $\mathcal{C}_1$, we have $t_n = 1$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

Similarly for class $\mathcal{C}_2$, we have $t_n = 0$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}).$$

Thus the likelihood function is given by

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$

ML parameter estimation

the prior class probability

$$\pi = \frac{1}{N}\sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

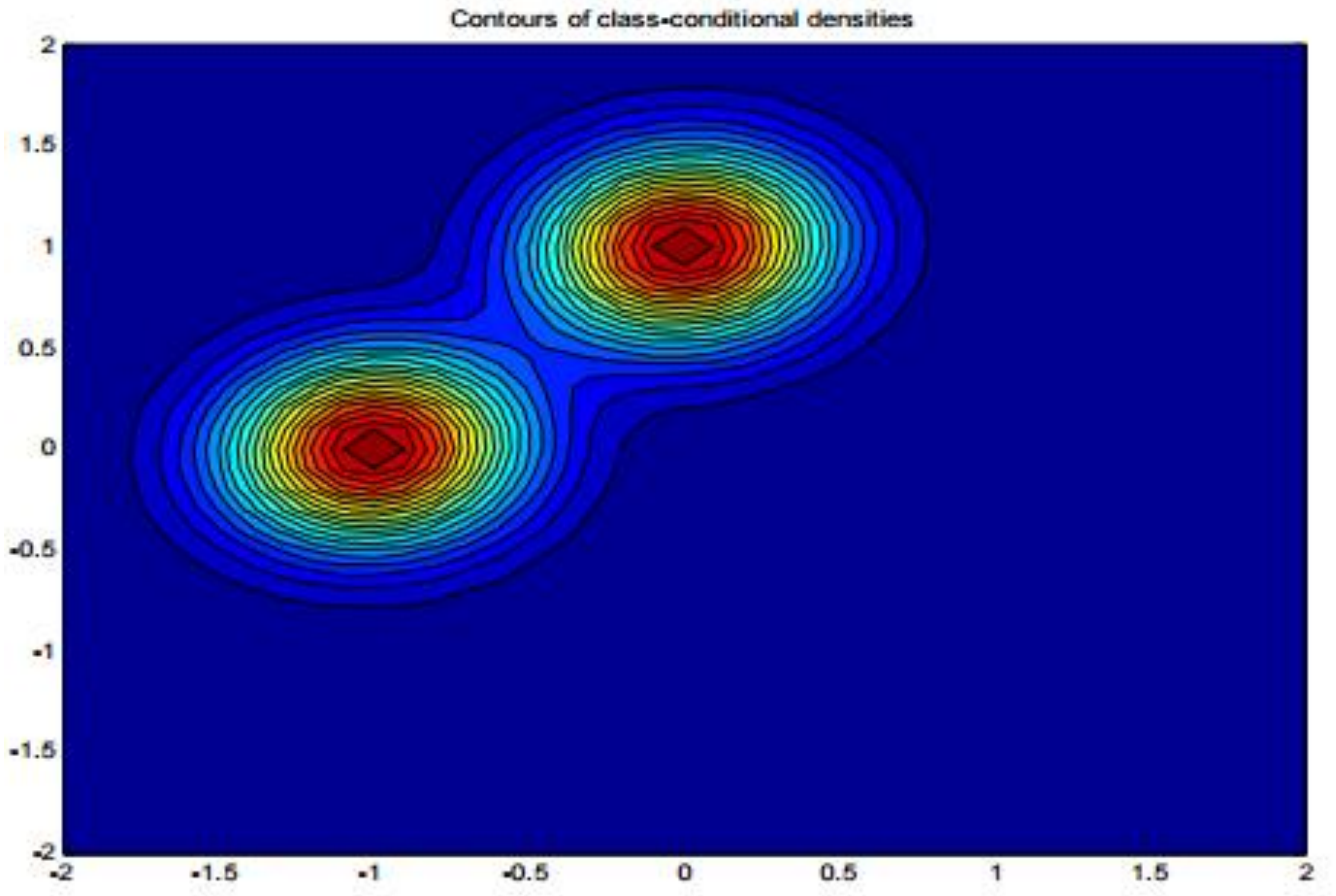
$$\mu_1 = \frac{1}{N_1}\sum^{N} t_n \mathbf{x}_n$$

$$\mu_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n$$

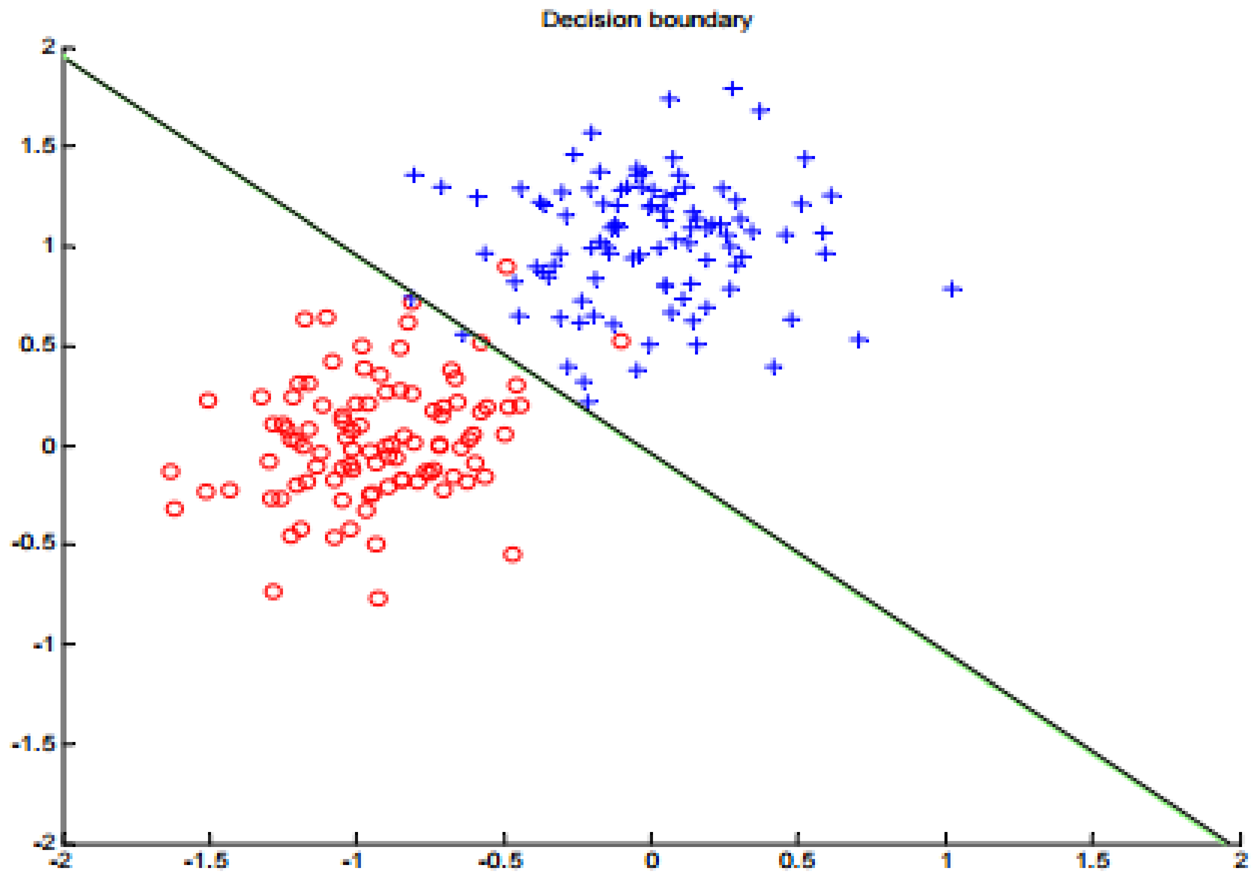
Shared covariance matrix

$$\begin{aligned} \boldsymbol{\Sigma} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \\ \mathbf{S}_1 &= \frac{1}{N_1}\sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mu_1)(\mathbf{x}_n - \mu_1)^{\mathrm{T}} \\ \mathbf{S}_2 &= \frac{1}{N_2}\sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mu_2)(\mathbf{x}_n - \mu_2)^{\mathrm{T}} \end{aligned}$$

linear decision boundary



linear decision boundary



Back to logistic regression

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

$$x \sim N(\mu_0, \Sigma) \quad \text{for} \quad y = 0$$
$$x \sim N(\mu_1, \Sigma) \quad \text{for} \quad y = 1$$

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

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

When is the logistic regression model correct?

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

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

$\boldsymbol{\theta}$ - 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 !!!!**

Quadratic discriminant analysis (QDA)

Model:

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

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

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

$$\begin{tikzcd} y \arrow[d] \\ \mathbf{x} \end{tikzcd}$$

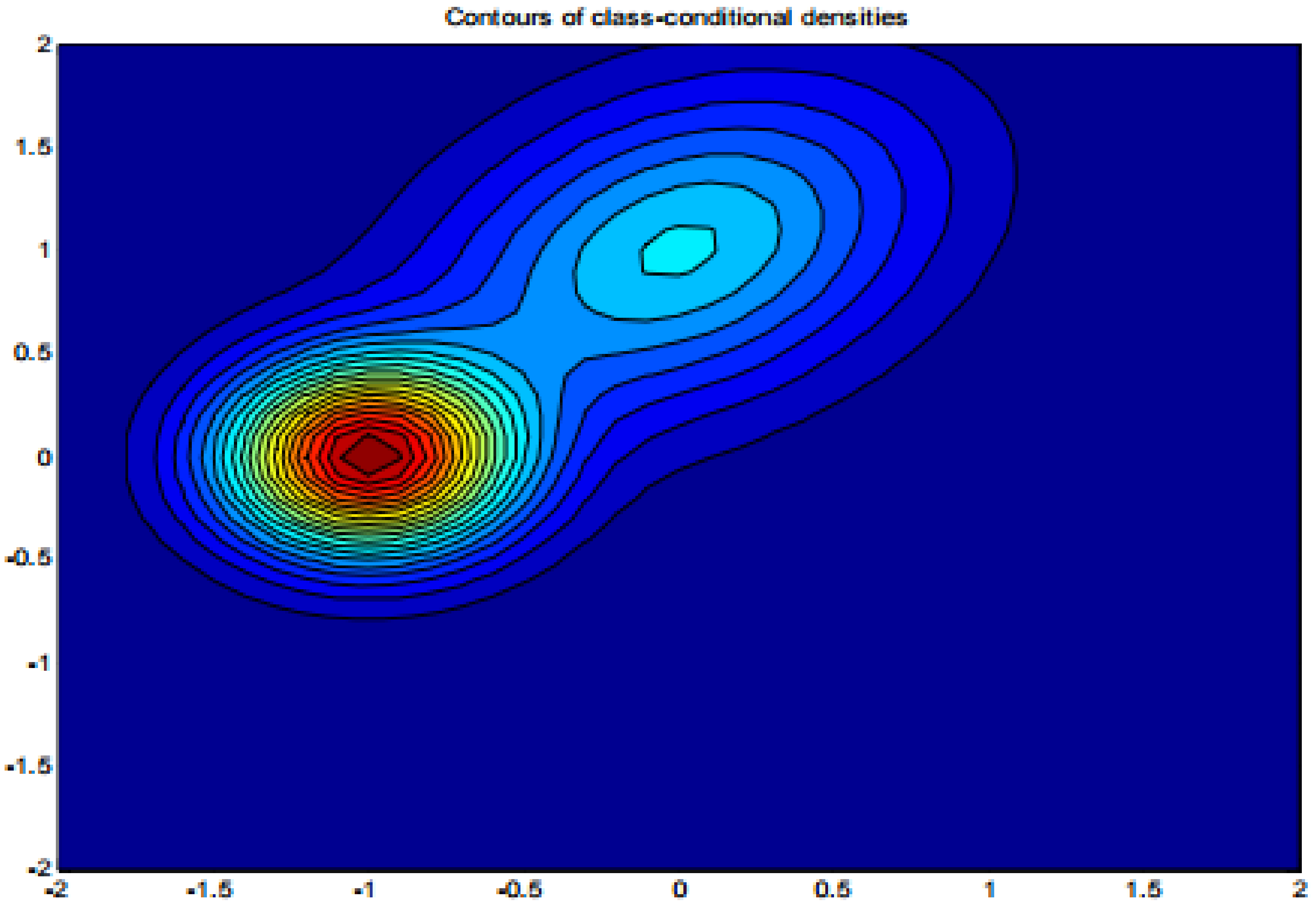
Multivariate normal $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

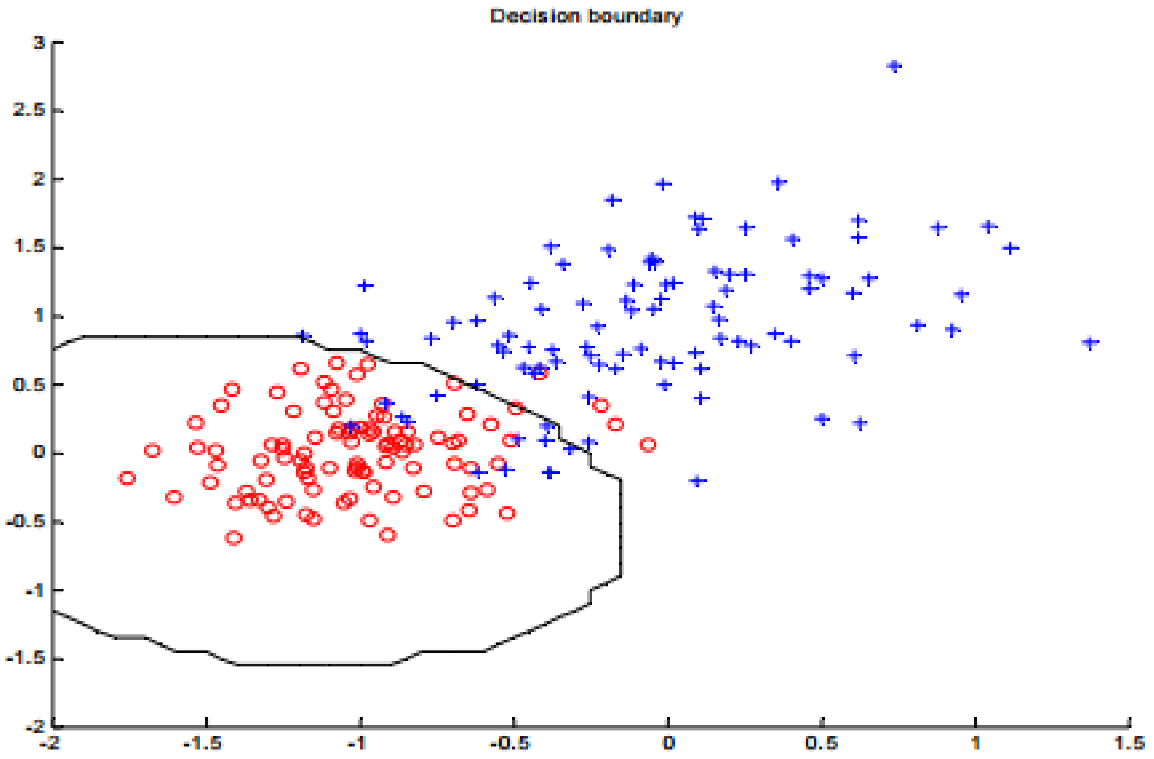
- **Priors on classes (class 0,1)** $y \sim Bernoulli$
  - **Bernoulli distribution**

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

QDA: Quadratic decision boundary



QDA: Quadratic decision boundary



Naïve Bayes classifier

- **A generative classifier model with additional simplifying assumptions:**
  - **All input attributes are conditionally independent of each other given the class.**

  So we have:

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

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

Learning parameters of the model

Much simpler density estimation problems

- We need to learn:

  $p(\mathbf{x} \mid y = 0)$ and $p(\mathbf{x} \mid y = 1)$ and $p(y)$

- Because of the assumption of the conditional independence we need to learn:

  for every variable i: $p(x_i \mid y = 0)$ and $p(x_i \mid y = 1)$

- **Much easier if the number of input attributes is large**
- **Also, the model gives us a flexibility to represent input attributes of different forms !!!**
- E.g. one attribute can be modeled using the Bernoulli, the other using Gaussian density, or as a Poisson distribution

Making a class decision for the Naïve Bayes

Discriminant functions

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

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

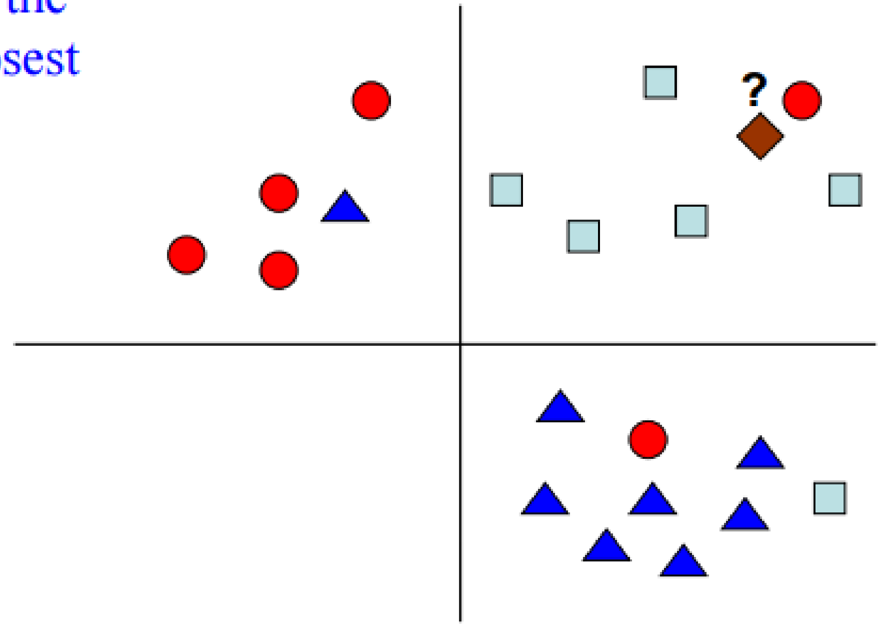
$$p(y=1\,|\,\mathbf{x}) = \frac{\left(\prod_{i=1}^{d} p(x_i\,|\,\Theta_{1,i})\right) p(y=1)}{\left(\prod_{i=1}^{d} p(x_i\,|\,\Theta_{1,i})\right) p(y=0) + \left(\prod_{i=1}^{d} p(x_i\,|\,\Theta_{2,i})\right) p(y=1)}$$

Outline

- Basic of Linear Classification Models
- Logistic Regresion
- Generative Model and Naïve Bayes
- KNN
- Evaluation of Classification

K nearest neighbors (KNN)

- A simple, yet surprisingly efficient algorithm
- Requires the definition of a distance function or similarity measures between samples
- Select the class based on the majority vote in the k closest points



Nearest-neighbor classification

learning task

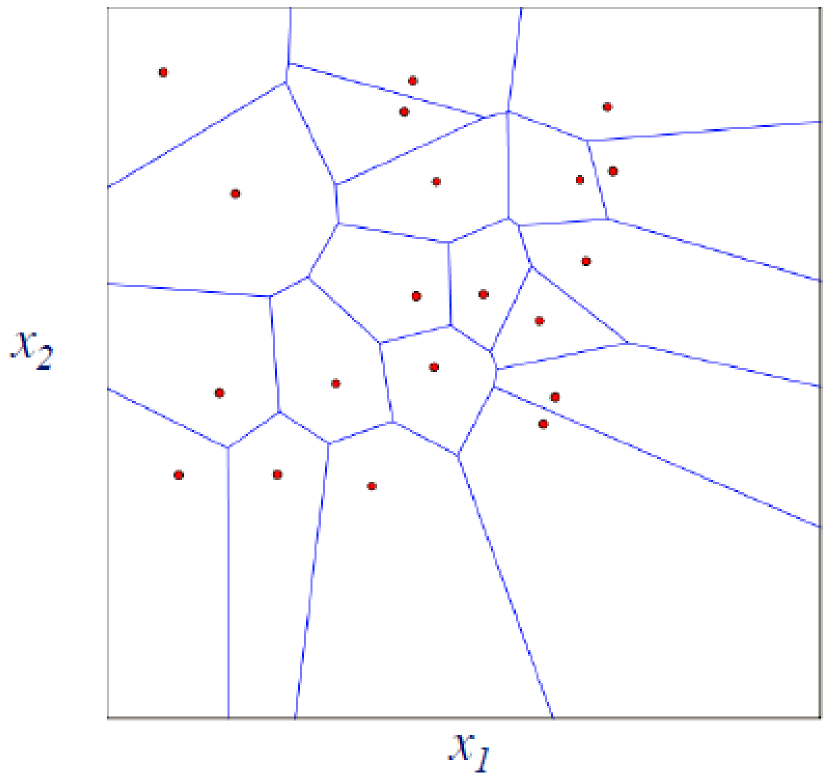
- given a training set $(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_n, y_n)$, do nothing (it's sometimes called a *lazy learner*)

classification task

- **given**: an instance $\boldsymbol{x}_q$ to classify
- find the training-set instance $\boldsymbol{x}_i$ that is most similar to $\boldsymbol{x}_q$
- return the class value y_i

The decision regions for nearest neighbor classification

Voronoi Diagram: Each polyhedron indicates the region of feature space that is in the nearest neighborhood of each training instance



k-nearest-neighbor classification

classification task

- **given**: an instance $\boldsymbol{x}_q$ to classify
- find the k training-set instances $(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_k, y_k)$ that are most similar to $\boldsymbol{x}_q$
- return the class value

$$\hat{y} \leftarrow \underset{v \in \text{values}(Y)}{\arg\max} \sum_{i=1}^{k} \delta(v, y_i) \qquad \delta(a,b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ otherwise} \end{cases}$$

(i.e. return the class that the plurality of the neighbors have)

How can we determine similarity/distance

suppose all features are nominal (discrete)

- Hamming distance: count the number of features for which two instances differ

suppose all features are continuous

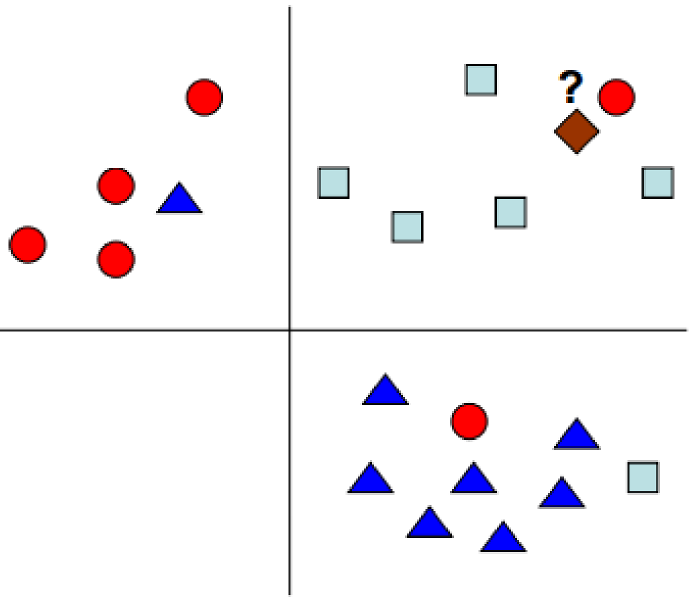
- Euclidean distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_f \left(x_{if} - x_{jf}\right)^2}$$ where x_{if} represents the f^{th} feature of $\mathbf{x}_i$

- Could also use Manhattan distance: sum the differences in feature values – continuous analog to Hamming distance

K nearest neighbors (KNN)

- Choice of k influences the 'smoothness' of the resulting classifier
- In that sense it is similar to a kernel methods (discussed later in the course)
- However, the smoothness of the function is determined by the actual distribution of the data (p(x)) and not by a predefined parameter.



A probabilistic interpretation of KNN

- The decision rule of KNN can be viewed using a probabilistic interpretation
- What KNN is trying to do is approximate the Bayes decision rule on a subset of the data
- To do that we need to compute certain properties including the conditional probability of the data given the class (p(x|y)), the prior probability of each class (p(y)) and the marginal probability of the data (p(x))
- These properties would be computed for some small region around our sample and the size of that region will be *dependent on the distribution of the test samples**

Computing probabilities for KNN

- Let V be the volume of the m dimensional ball around z containing the k nearest neighbors for z (where m is the number of features).
- Then we can write

$$p(x)V = P = \frac{K}{N} \qquad p(x) = \frac{K}{NV} \qquad p(x \mid y=1) = \frac{K_1}{N_1 V} \qquad p(y=1) = \frac{N_1}{N}$$

- Using Bayes rule we get:

$$p(y=1 \mid z) = \frac{p(z \mid y=1)\, p(y=1)}{p(z)} = \frac{K_1}{K}$$

z – new data point to classify

V - selected ball

P – probability that a random point is in V

N - total number of samples

K - number of nearest neighbors

N_1 - total number of samples from class 1

K_1 - number of samples from class 1 in K

Computing probabilities for KNN

N - total number of samples

V - Volume of selected ball

K - number of nearest neighbors

N_1 - total number of samples from class 1

K_1 - number of samples from class 1 in K

- Using Bayes rule we get:

$$p(y=1\,|\,z) = \frac{p(z\,|\,y=1)\,p(y=1)}{p(z)} = \frac{K_1}{K}$$

Using Bayes decision rule we will chose the class with the highest probability, which in this case is the class with the highest number of samples in K

Outline

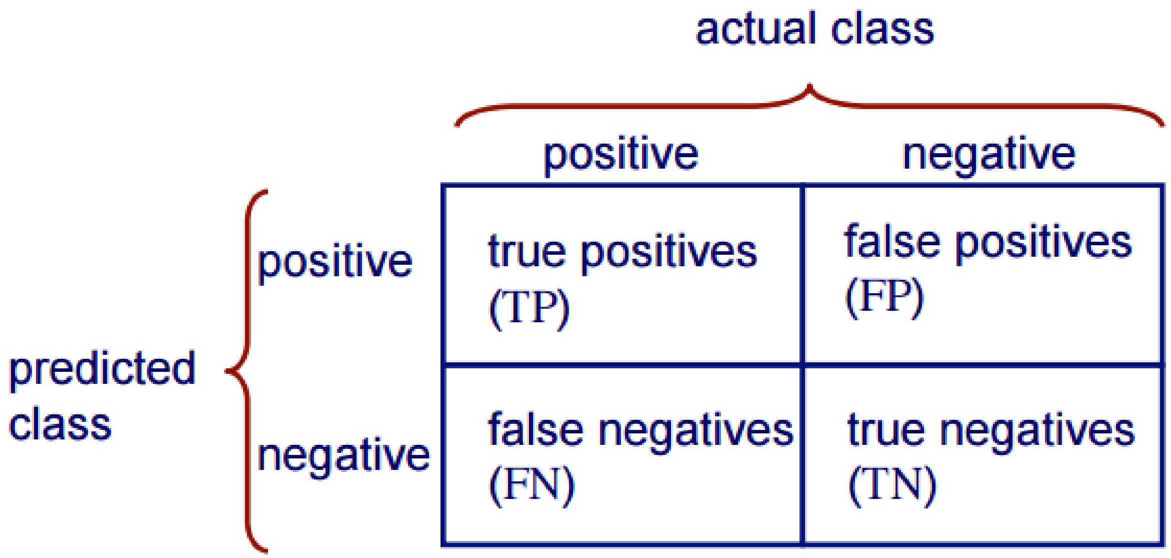
- Basic of Linear Classification Models
- Logistic Regresion
- Generative Model and Naïve Bayes
- KNN
- Evaluation of Classification

Confusion matrix for 2-class problems

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Other accuracy metrics



		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

敏感度 （查全率）

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

TP rate

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

FP rate

Additional statistics

- **Sensitivity (recall)**

$$SENS = \frac{TP}{TP + FN}$$

敏感度（查全率）

- **Specificity**

$$SPEC = \frac{TN}{TN + FP}$$

特异度

- **Positive predictive value (precision)**

$$PPT = \frac{TP}{TP + FP}$$

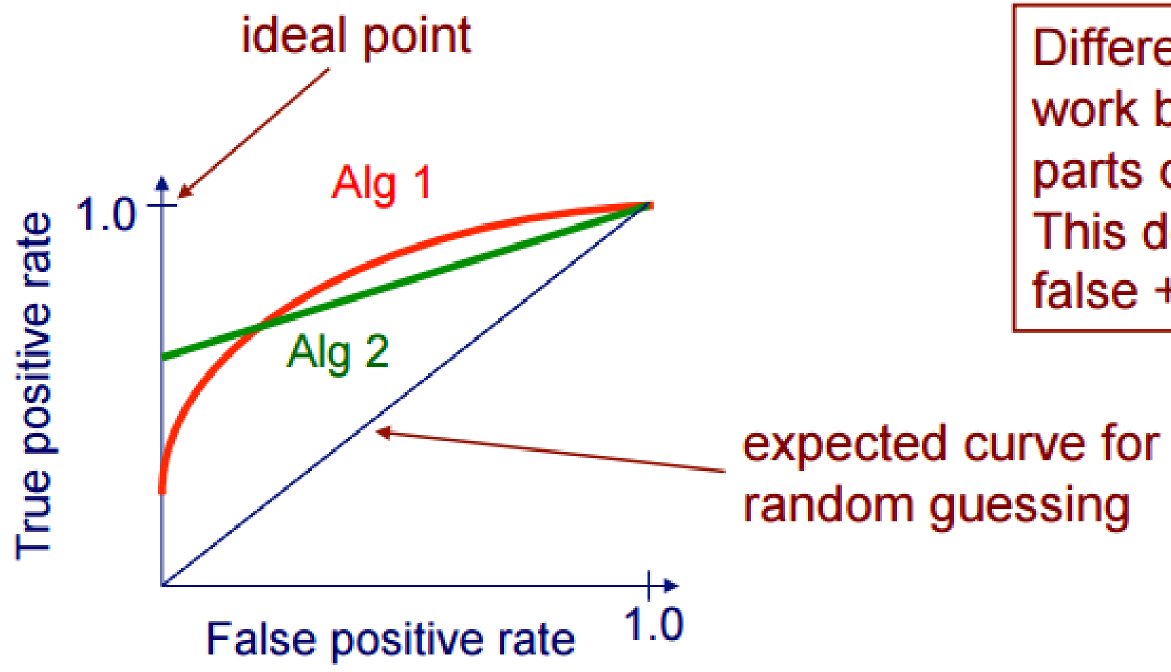
查准率

- **Negative predictive value**

$$NPV = \frac{TN}{TN + FN}$$

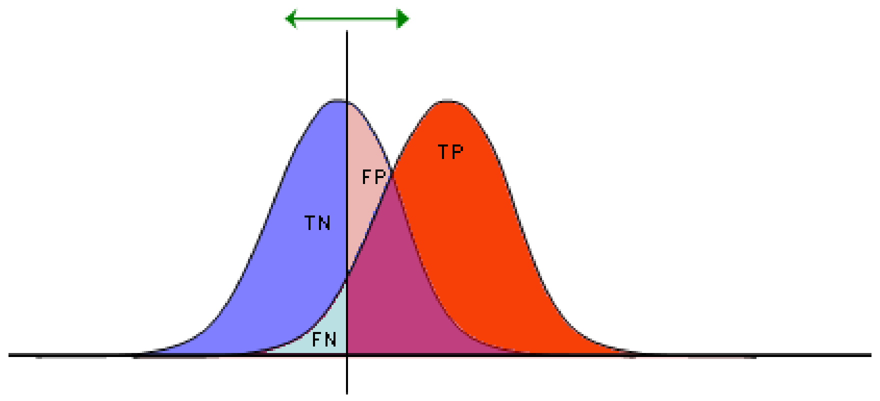
ROC curves

A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied



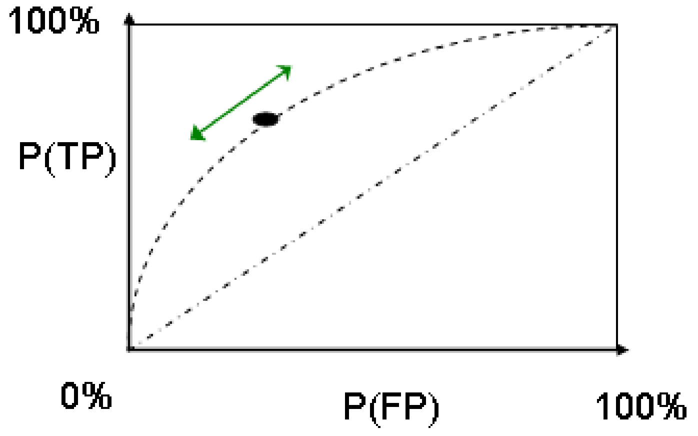
Different methods can work better in different parts of ROC space. This depends on cost of false + vs. false -

ROC curve



TP	FP
FN	TN
1	1

$$\frac{TP}{TP+FN}$$



$$\frac{FP}{TN+FP}$$

ROC curve example

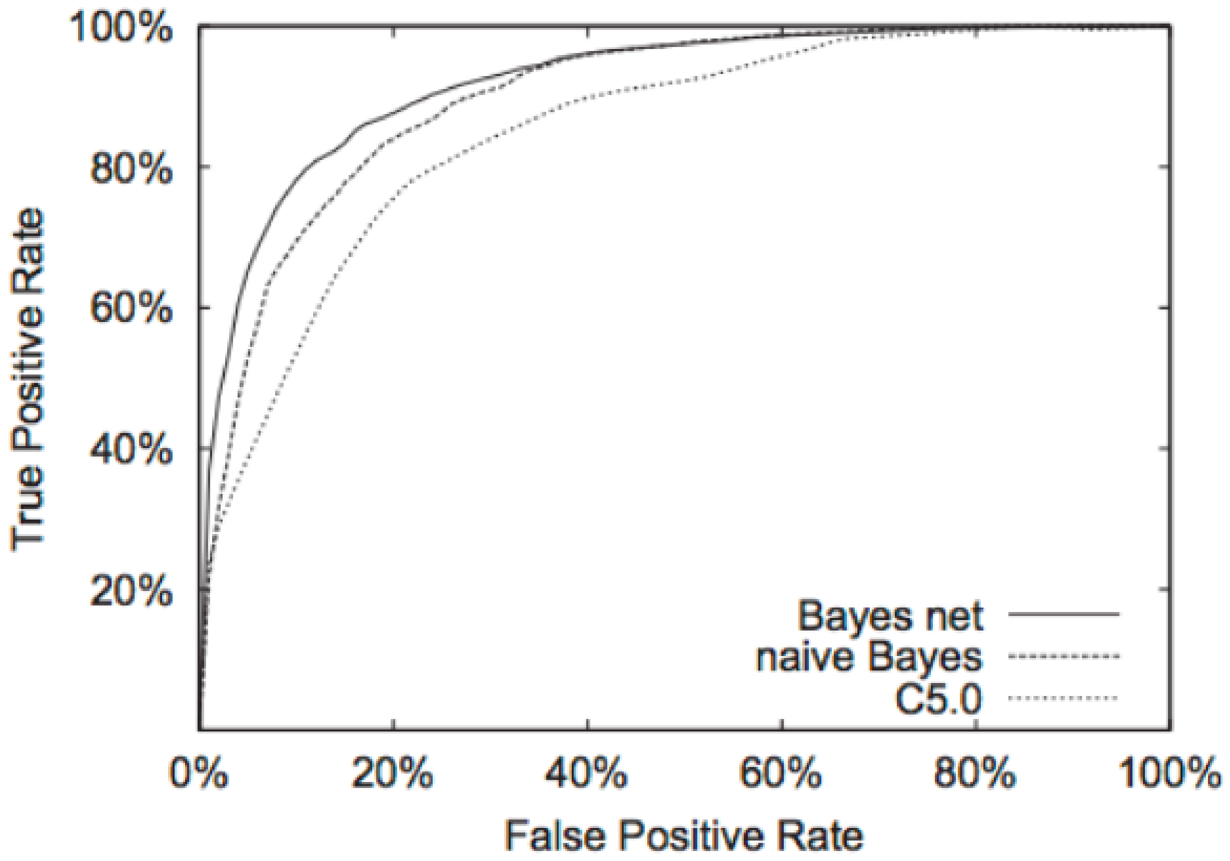
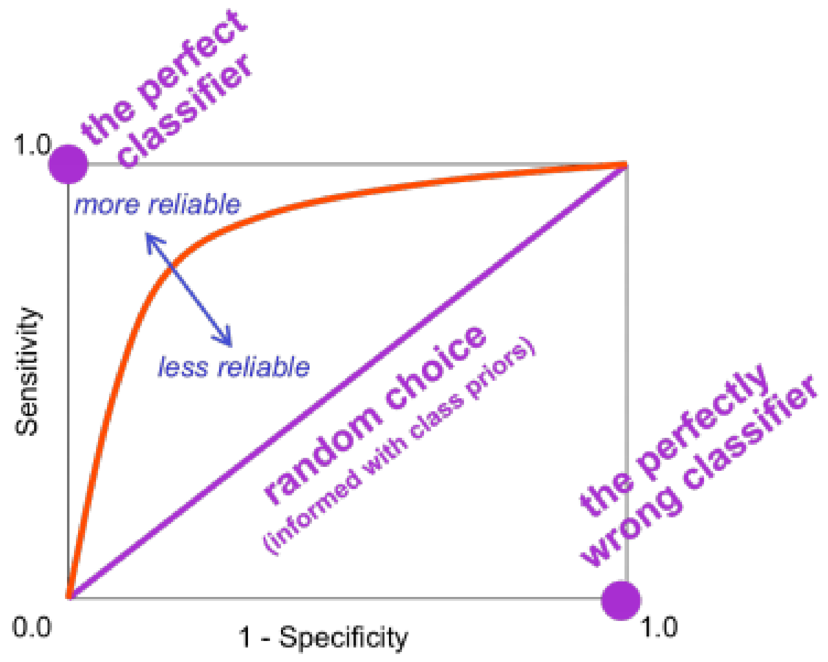


figure from Bockhorst et al., *Bioinformatics* 2003

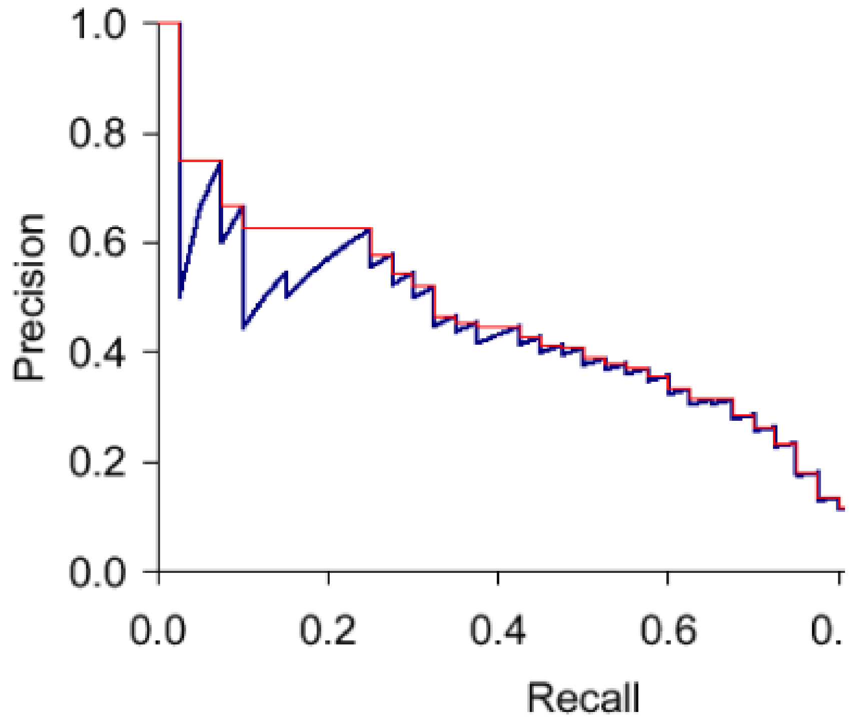
Area Under the Curve



- The AUC is the *area under the ROC curve*

- AUC has a nice interpretation: The AUC is the probability that the classifier will rank a *randomly selected* observation where $y_i = 1$ higher than a *randomly selected* observation where $y_i = 0$

Precision-Recall curves



- Precision: $TP/(TP + FP)$ (aka, PPV)
- Recall: $TP/(TP + FN)$ (aka, Sensitivity)
- Precision @50% Recall is a common performance metric

[source: Introduction to Information Retrieval, Manning et al.]

本章阅读

- Bishop Ch 4.1.1, 4.1.3, 4.2.1, 4.2.2, 4.3.2