INFO-F-422: Statistical foundations of machine learning

Classification

Gianluca Bontempi

Machine Learning Group
Computer Science Department
mlg.ulb.ac.be

Classification problem

- Let $\mathbf{x} \in \mathbb{R}^n$ be the input and $\mathbf{y}$ a categorical output variable that takes values in $\{c_1, \ldots, c_K\}$.
- E.g, let $\mathbf{x}$ be the month and $\mathbf{y}$ a categorical variable taking $K = 2$ possible values $\{RAIN, NO.RAIN\}$ or $\{SPAM, NO.SPAM\}$.

Separability and noise

- **Separable classes:** given an input x, the output $\mathbf{y}$ takes always the same value. In other terms

$$\forall x \in \mathbb{R}^n \quad \exists c_k : \text{Prob}\{\mathbf{y} = c_k | x\} = 1$$

  This is also known as *noiseless* or *degenerate* situation.

- **Non separable classes:** given an input x, we can have realizations of $\mathbf{y}$ with different values. In other terms

$$\exists x \in \mathbb{R}^n : \quad \forall c_k \quad \text{Prob}\{\mathbf{y} = c_k | x\} < 1$$

Stochastic non separable setting

- We consider a stochastic setting to model non separable tasks.
- This means that data are noisy and follow a probabilistic distribution.
- Given an input x, $\mathbf{y}$ does not always take the same value.
- $\mathbf{y}$ follows the conditional probabilistic distribution

$$\sum_{k=1}^{K} \text{Prob}\left\{\mathbf{y} = c_k | x\right\} = 1$$

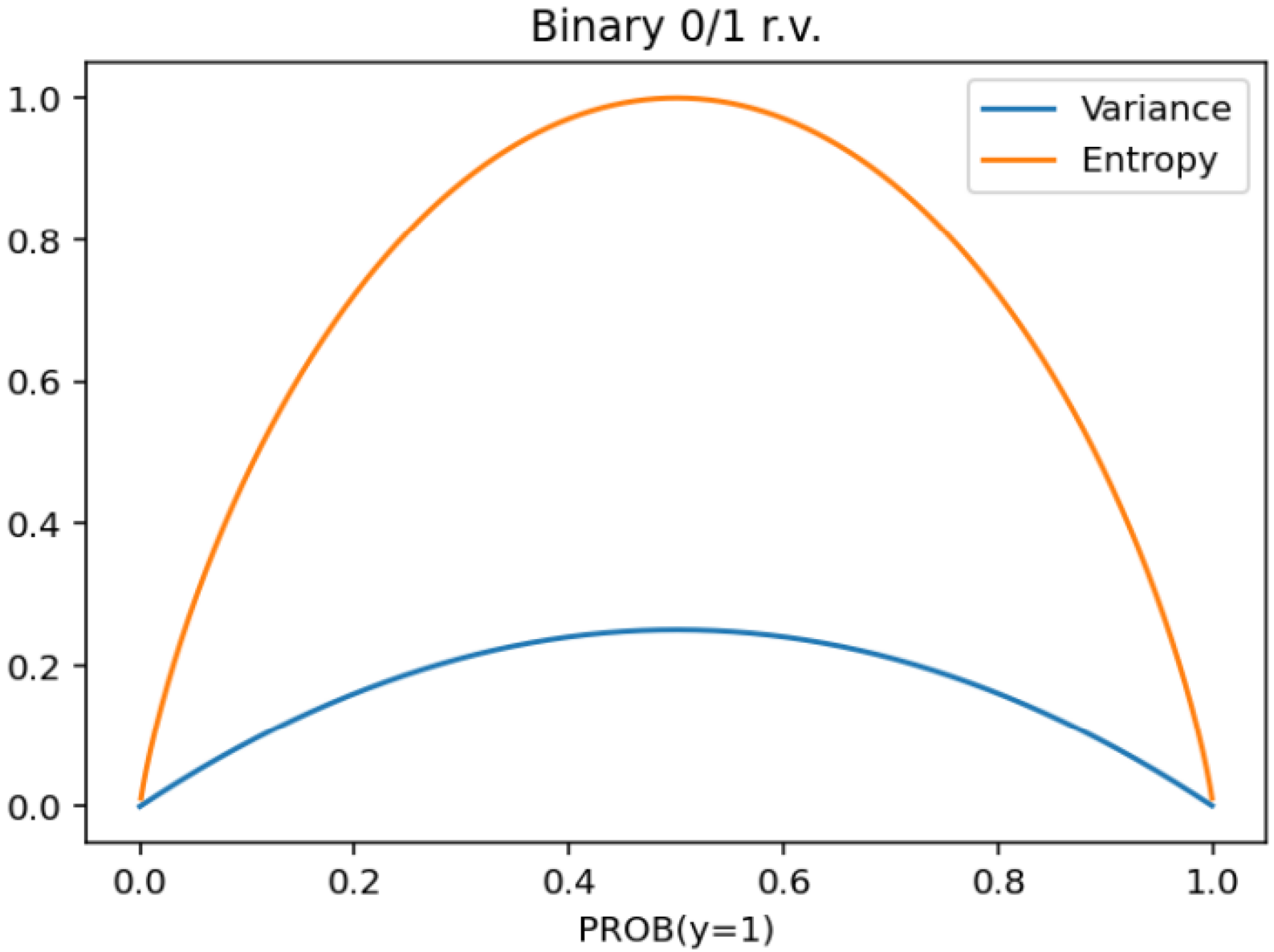
Binary classification problem

- It is a problem where the output class $\mathbf{y}$ can take only $K = 2$ values.
- Suppose for simplicity that $\mathbf{y} \in \{c_1 = 0, c_2 = 1\}$.
- Let us denote

$$p_0 = \text{Prob}\{\mathbf{y} = 0\}$$
$$p_1 = \text{Prob}\{\mathbf{y} = 1\}$$

  where $p_1 + p_0 = 1$.
- Note that for a binary variable

$$E[\mathbf{y}] = 0 \cdot p_0 + 1 \cdot p_1 = p_1$$
$$\text{Var}[\mathbf{y}] = E[(\mathbf{y} - E[\mathbf{y}])^2] = p_0(0 - p_1)^2 + p_1(1 - p_1)^2 = p_1(1 - p_1)$$
$$H[\mathbf{y}] = E[-\log p(\mathbf{y})] = -p_0 \log p_0 - p_1 \log p_1$$


Binary 0/1 r.v.
Variance
Entropy
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PROB(y=1)

Degree of non separability

In the case of non separable classes we can have classification problems with different degrees of separability.
Let

$$\text{Prob}\{\mathbf{y} = 1|x\} = p_1(x), \qquad \text{Prob}\{\mathbf{y} = 0|x\} = p_0(x)$$

The degree of non separability for an input x may be quantified by some quantitative measures like

- **Conditional variance**

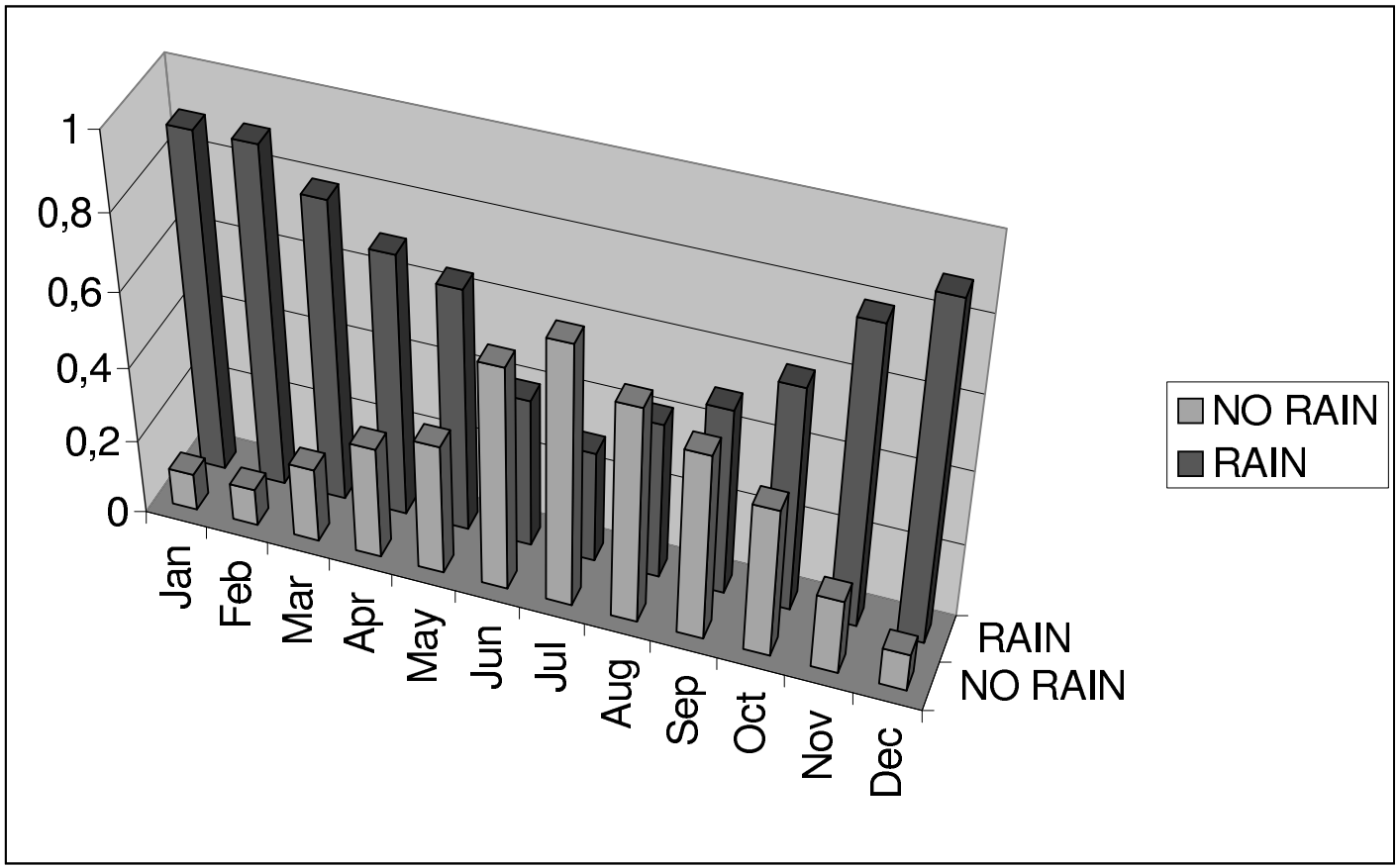
$$\text{Var}[\mathbf{y}|x] = p_1(x)(1 - p_1(x))$$

- **Conditional entropy**

$$H[\mathbf{y}|x] = -p_0(x)\log p_0(x) - p_1(x)\log p_1(x)$$

Both measures attain their maximum value when $p_0 = p_1 = 1/2$

The conditional distribution



For a fixed month

$$\text{Prob}\{\mathbf{y} = RAIN|\mathbf{x} = month\}+\text{Prob}\{\mathbf{y} = NO.RAIN|\mathbf{x} = month\} = 1$$

Classification probabilistic setting

- $\mathbf{x} \in \mathbb{R}^n$: real-valued input vector
- $\mathbf{y}$: categorical random output variable in $\{c_1, \ldots, c_K\}$.
- $\text{Prob}\{\mathbf{y} = c_k | x\}$: probability that output belongs to the kth class given x:

$$\sum_{k=1}^{K} \text{Prob}\{\mathbf{y} = c_k | x\} = 1$$

- $\hat{c}(x)$: output class prediction in $\{c_1, \ldots, c_K\}$
- $L_{(K \times K)}$: *loss matrix* null on the diagonal and non negative elsewhere.

Loss matrix

PRED \ REAL	c_1	c_2	c_3
$\widehat{c}_1$	L_{11}	L_{12}	L_{13}
$\widehat{c}_2$	L_{21}	L_{22}	L_{23}
$\widehat{c}_3$	L_{31}	L_{32}	L_{33}

- $L_{jk} = L_{(c_j, c_k)}$: misclassification cost when predicted class is $\hat{c}(x) = c_j$ and correct class is c_k.
- Suppose that for a given x the classifier returns $\hat{c}(x)$. The average cost of this classification is

$$\sum_{k=1}^{K} L_{(\hat{c}(x), c_k)} \text{Prob}\left\{\mathbf{y} = c_k | x\right\}$$

Ethical costs (self-driving car)

Does the car brake or not (class **y**) given the observed image (input **x**)?

CLASSIFICATION / ACTION \ REAL	Pedestrian	Bag
Pedestrian (Action: stop)	0	? (cost of lack of comfort, delay, reduced productivity)
Bag (Action: go on)	? (value of a human being)	0

Who set those numbers in the Uber car? Those numbers determines the ethics of the safety-critical agent

Bayes classifier

- Classification $\hat{c}(x)$ is optimal if it minimises

$$\sum_{k=1}^{K} L_{(\hat{c}(x),c_k)} \text{Prob}\{\mathbf{y} = c_k|x\}$$

- Optimal classifier (also known as the *Bayes classifier*) returns for all x

$$c^*(x) = \arg \min_{c_j \in \{c_1,\ldots,c_K\}} \sum_{k=1}^{K} L_{j,k} \text{Prob}\{\mathbf{y} = c_k|x\} =$$

$$= \arg \min_{c_j \in \{c_1,\ldots,c_K\}} E_{\mathbf{y}}[L_{(c_j,\mathbf{y})}|x]$$

i.e. the class that minimises the expected cost.

The 0-1 case

- If 0-1 loss function, the optimal classifier

$$c^*(x) = \arg \min_{c_j \in \{c_1,\ldots,c_K\}} \sum_{k=1:K, k\neq j} \text{Prob}\{\mathbf{y} = c_k|x\} =$$

$$= \arg \min_{c_j \in \{c_1,\ldots,c_K\}} (1 - \text{Prob}\{\mathbf{y} = c_j|x\}) =$$

$$= \arg \max_{c_j \in \{c_1,\ldots,c_K\}} \text{Prob}\{\mathbf{y} = c_j|x\}$$

- Bayes classifier selects the *maximum a posteriori* class c_j, $j = 1, \ldots, K$

Discrete input example

PRED \ REAL	C1	C2	C3
C1	0	1	5
C2	20	0	10
C3	2	1	0

x	C1	C2	C3
1	0.1	0.6	0.3
2	0.2	0.8	0.0
3	0.9	0.04	0.06
4	0.5	0.25	0.25
5	0.3	0.1	0.6

Bayes classification in $x = 2$ is given by

$$c^*(2) = \arg \min_{k=1,2,3} \{0.2 * 0 + 0.8 * 1 + 0.0 * 5, \quad \textit{(avg loss if } \hat{c} = 1\textit{)}$$
$$0.2 * 20 + 0.8 * 0 + 0.0 * 10, \quad \textit{(avg loss if } \hat{c} = 2\textit{)}$$
$$0.2 * 2 + 0.8 * 1 + 0.0 * 0 \quad \textit{(avg loss if } \hat{c} = 3\textit{)}$$
$$\} = \arg \min_{k=1,2,3} \{0.8, 4, 1.2\} = 1$$

What would have been Bayes classification in the 0-1 case?

Bayes' theorem for continuous inputs

For categorical $\mathbf{y}$ and continuous $\mathbf{x}$

$$\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} = \frac{p_{\mathbf{x}}(x|\mathbf{y} = c_k)\text{Prob}\{\mathbf{y} = c_k\}}{\sum_{k=1}^{K} p_{\mathbf{x}}(x|\mathbf{y} = c_k)\text{Prob}\{\mathbf{y} = c_k\}}$$

$$p_{\mathbf{x}}(x|\mathbf{y} = c_k) = \frac{\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}\, p_{\mathbf{x}}(x)}{\text{Prob}\{\mathbf{y} = c_k\}}$$

- Knowing the conditional distribution $\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}$ and the apriori density $p_{\mathbf{x}}(x)$, we can derive the conditional density $p_{\mathbf{x}}(x|\mathbf{y} = c_k)$.
- Note that for a multivariate $\mathbf{x} \in \mathbb{R}^n$
  1. $\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}$ is a multi-input single-output function which for each values of x returns a value in $[0, 1]$.
  2. for a given class $p_{\mathbf{x}}(x|\mathbf{y} = c_k)$ is a multivariate n-dimensional density.

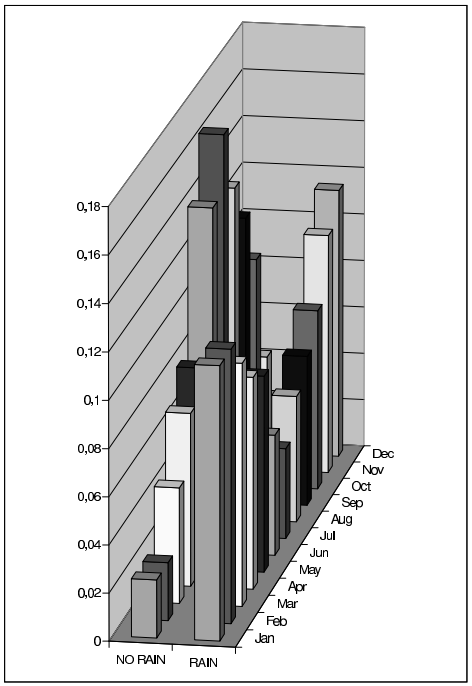
Bayes' theorem for discrete inputs

For categorical $\mathbf{y}$ and discrete $\mathbf{x}$

$$\text{Prob}\{\mathbf{y} = c_k|\mathbf{x} = x\} = \frac{\text{Prob}\{\mathbf{x} = x|\mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}}{\sum_{k=1}^{K}\text{Prob}\{\mathbf{x} = x|\mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}}$$

$$\text{Prob}\{\mathbf{x} = x|\mathbf{y} = c_k\} = \frac{\text{Prob}\{\mathbf{y} = c_k|\mathbf{x} = x\}\,\text{Prob}\{\mathbf{x} = x\}}{\text{Prob}\{\mathbf{y} = c_k\}}$$

Inverse (or class-) conditional distribution

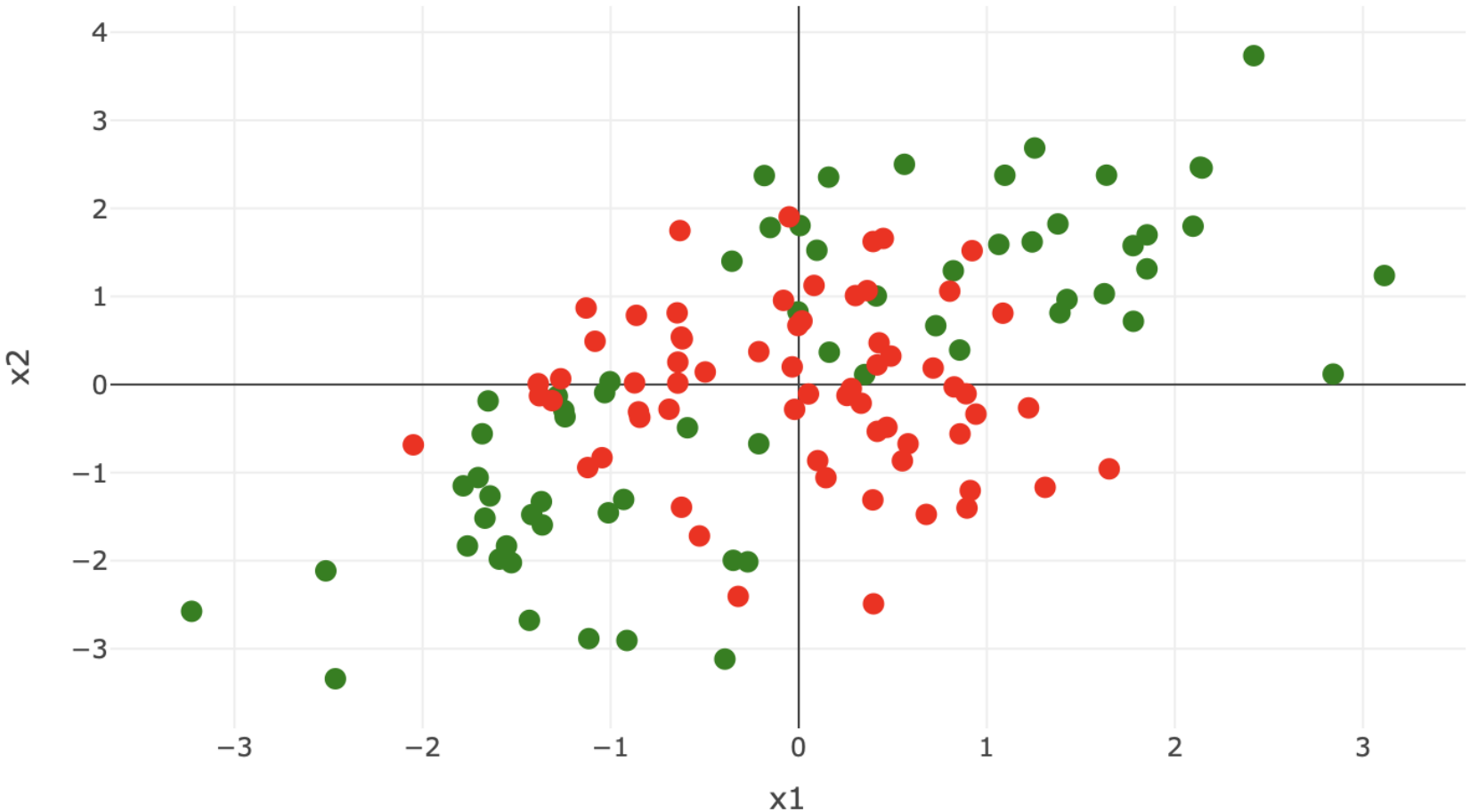


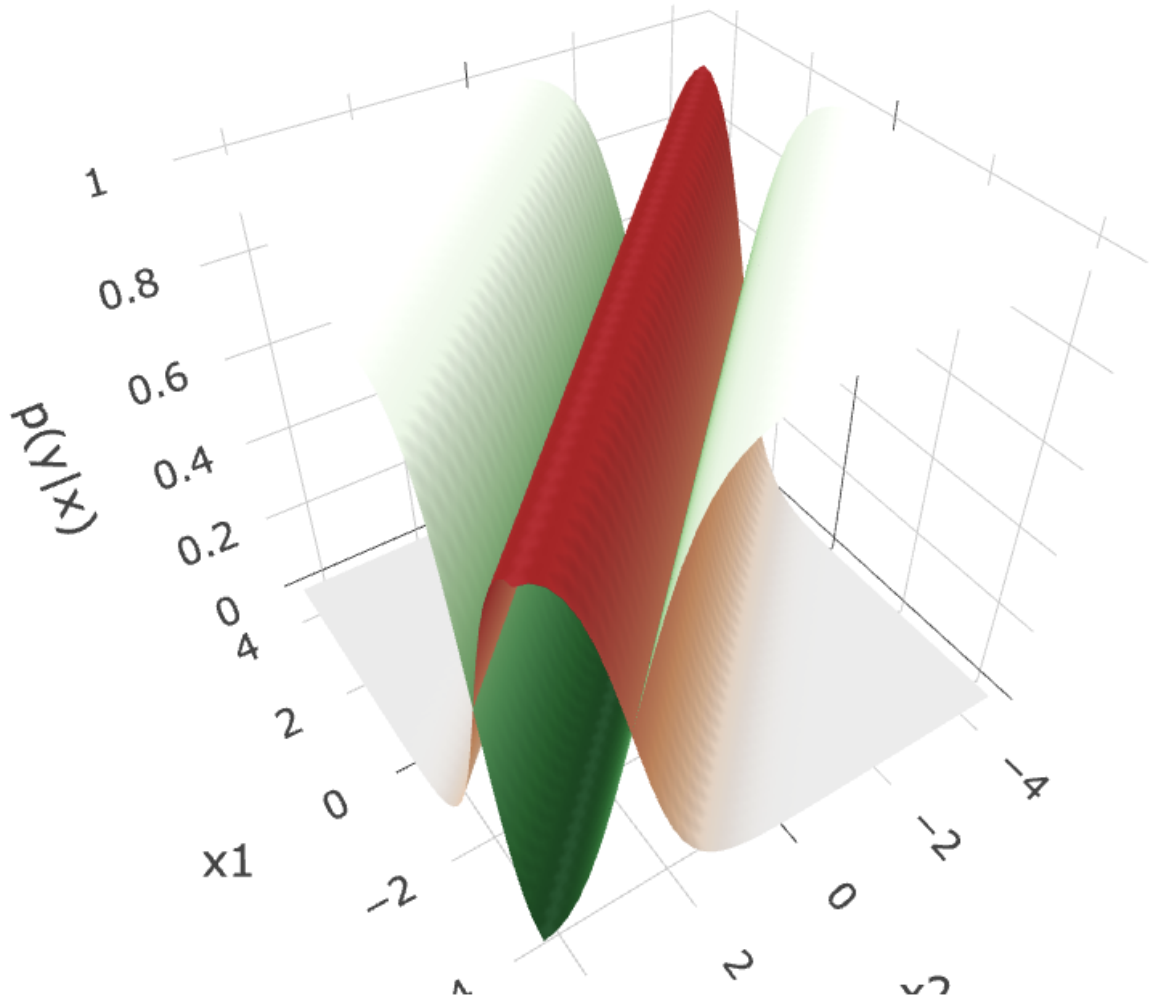
The figure assumes that the apriori distribution is uniform.

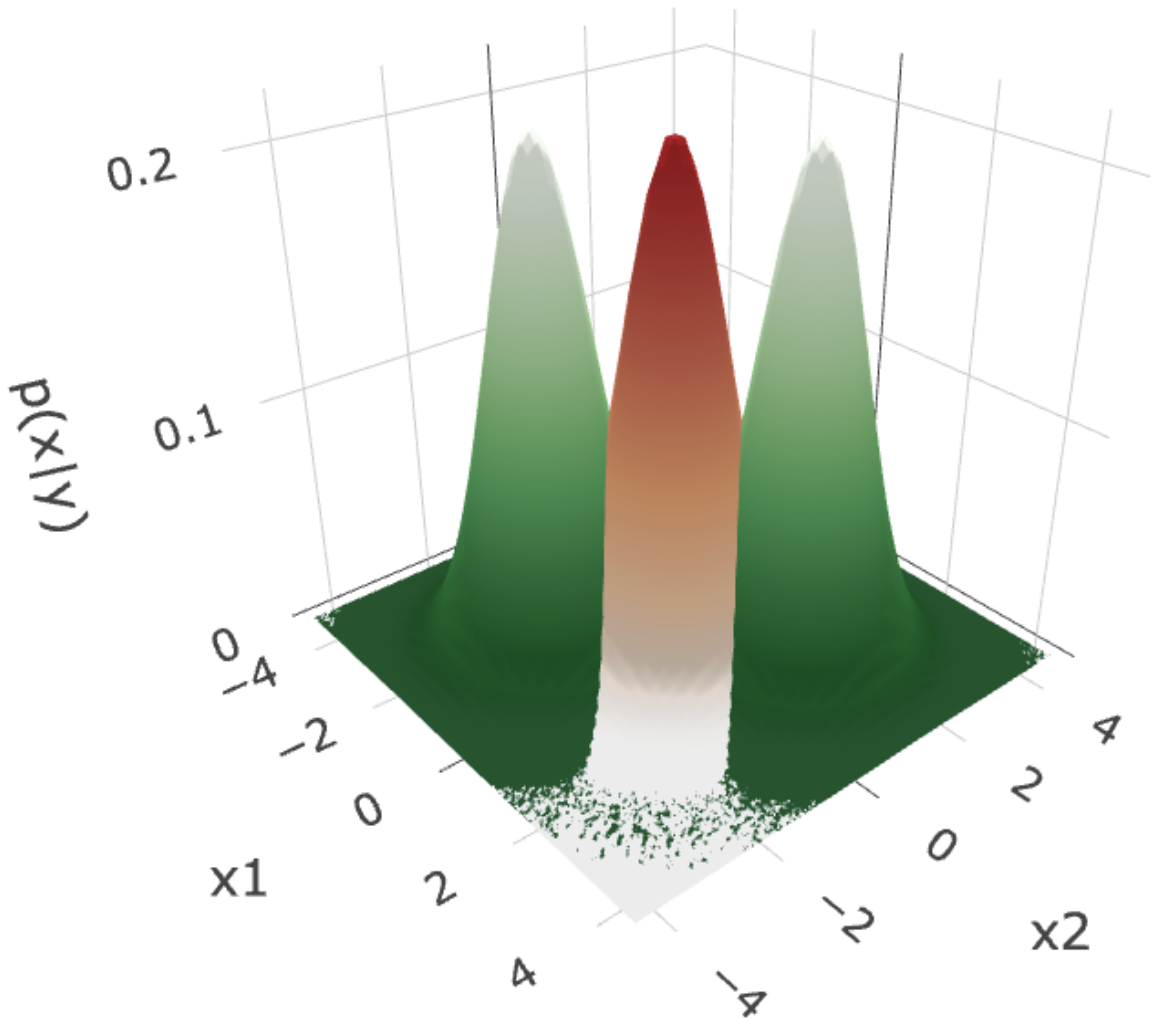
$$\sum_{month} \text{Prob}\{\mathbf{x} = month | \mathbf{y} = NO.RAIN\} =$$

$$= \sum_{month} \text{Prob}\{\mathbf{x} = month | \mathbf{y} = RAIN\} = 1$$

Two dimensional input




p(y|x)
1
0.8
0.6
0.4
0.2
0
x1
4
2
0
−2
−4
2
0
−2


0.2
0.1
0
p(x|y)
−4
−2
0
2
4
x1
4
2
0
−2
−4
x2

Classification strategies

Optimal classification is possible only if the quantities $\text{Prob}\{\mathbf{y} = c_k|x\}$, $k = 1, \ldots, K$ are known. What happens if this is not the case?

Four strategies are generally used.

- **1. Direct estimation via regression techniques.** If the classification problem has $K = 2$ classes and if we denote them by $y = 0$ and $y = 1$

  $$E[\mathbf{y}|x] = 1 \cdot \text{Prob}\{\mathbf{y} = 1|x\} + 0 \cdot \text{Prob}\{\mathbf{y} = 0|x\} = \text{Prob}\{\mathbf{y} = 1|x\}$$

  Then the classification problem can be put in the form of a regression problem where the output takes value in $\{0, 1\}$.

Limitations of the regression approach

- Regression typically relying on least-squares approaches which makes implicitly the hypothesis of conditional Gaussian distributions.
- Predictions out of the $[0, 1]$ interval.
- Predictions hard to interpret in probabilistic terms.
- Solutions highly sensitive to inputs distributions: sum-of-squares minimisation give too weight to
  - input points far from the separating boundaries,
  - input outliers

- **2. Direct estimation via cross-entropy.** It models the conditional distribution $\text{Prob}\{\mathbf{y} = c_k|x\}, k = 1, \ldots, K$ with a set of K functions $\hat{P}_k(x, \alpha), k = 1, \ldots, K$ satisfying the constraints $\hat{P}_k(x, \alpha) > 0$ and $\sum_{k=1}^{K} \hat{P}_k(x, \alpha) = 1$.

Parametric estimation by maximum likelihood or minimization of cross-entropy

$$J(\alpha) = -\log \prod_{i=1}^{N} \text{Prob}\{\mathbf{y} = y_i|x_i\} = -\sum_{i=1}^{N} \log \hat{P}_{y_i}(x_i, \alpha)$$

Cross-entropy approaches

Typical approaches rely on logistic regression and neural networks. In logistic regression for a two-class tasks we have

$$\hat{P}_1(x, \alpha) = \frac{\exp^{x^T \alpha}}{1 + \exp^{x^T \alpha}}, \qquad \hat{P}_2(x, \alpha) = \frac{1}{1 + \exp^{x^T \alpha}}$$

which implies

$$\log \frac{\hat{P}_1(x, \alpha)}{\hat{P}_2(x, \alpha)} = x^T \alpha$$

where $\alpha \in \mathbb{R}^n$ and the transformation $\log \frac{p}{1-p}$ is also known as the *logit* transformation.

Logistic regression

In a binary case ($c_1 = 1, c_2 = 0$) the cross-entropy function becomes

$$J(\alpha) = -\sum_{i=1}^{N} \log \hat{P}_{y_i}(x_i, \alpha) =$$

$$-\sum_{i=1}^{N} \{y_i \log \hat{P}_1(x_i, \alpha) + (1 - y_i) \log(1 - \hat{P}_1(x_i, \alpha))\} =$$

$$-\sum_{i=1}^{N} \{y_i x_i^T \alpha - \log(1 + \exp^{x_i^T \alpha})\}$$

Given the nonlinear nature of the task an iterative solution (iteratively reweighted least squares) is required.

- **3. Discriminant functions:** classifier as a set of K discriminant functions

$$g_k(x) = \text{Prob}\{\mathbf{y} = k|x\}, \quad k = 1, \ldots, K$$

associated to the K a posteriori probabilities such that $\hat{c}(x) = c_k$ if

$$g_k(x) > g_j(x) \quad \text{for all } j \neq k$$

Discriminant functions partition the input space into K *decision regions* separated by *decision boundaries*

- **4. Density estimation via Bayes theorem.** Since

$$\text{Prob}\{\mathbf{y} = c_k|x\} = \frac{p(x|\mathbf{y} = c_k)\text{Prob}\{\mathbf{y} = c_k\}}{p(x)}$$

an estimation of $\text{Prob}\{x|\mathbf{y} = c_k\}$ allows an estimation of $\text{Prob}\{\mathbf{y} = c_k|x\}$.

This is also called the *generative approach* in alternative to the *discriminative approach* of the previous strategies.

Discriminant functions

- Classifier as set of K discriminant functions $g_k(x)$, $k = 1, \ldots, K$ such that $\hat{c}(x) = c_k$ if

$$k = \arg\max_j g_j(x)$$

- Zero-one loss function: optimal classifier is the *maximum a posteriori* discriminant function $g_k(x) = \text{Prob}\{\mathbf{y} = k|x\}$.
- The discriminant functions divide the feature space into K *decision regions* separated by *decision boundaries*

Discriminant functions and Bayes rule

- We can multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision.
- If we replace every $g_k(z)$ by $f(g_k(z))$, where $f(\cdot)$ is a monotonically increasing function, the classification is unchanged:

$$\arg\max_j g_j(x) = \arg\max_j f(g_j(x))$$

- Any of the following choices gives identical classification result:

$$g_k(x) = \text{Prob}\{\mathbf{y} = k|x\} = \frac{p(x|\mathbf{y} = k)P(\mathbf{y} = k)}{p(x)}$$

$$f(g_k(x)) = p(x|\mathbf{y} = k)P(\mathbf{y} = k)$$

$$f(g_k(x)) = \ln p(x|\mathbf{y} = k) + \ln P(\mathbf{y} = k)$$

Discriminant functions in the gaussian case

- Multivariate normal densities $p(x|\mathbf{y}=k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ where $x \in \mathbb{R}^n$, μ_k is a $[n,1]$ vector and Σ_k is a $[n,n]$ covariance matrix.
- Since

$$p(x|\mathbf{y}=k) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det(\Sigma_k)}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right\}$$

the discriminant function is then

$$g_k(x) = \ln p(x|\mathbf{y}=k) + \ln P(\mathbf{y}=k) =$$
$$= -\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) - \frac{n}{2}\ln 2\pi - \frac{1}{2}\ln\det(\Sigma_k) + \ln P(\mathbf{y}=k)$$

QDA vs LDA

- No assumptions about Σ_k: discriminant functions are quadratic (QDA).
- Assumption of identical covariance $\Sigma_k = \Sigma \; \forall k$: linear discriminant functions (LDA). The matrix Σ is also called the *pooled covariance matrix*.
- In the following we will consider the simplest LDA case where the covariance matrices are not only identical but also spherical:

$$\Sigma_k = \sigma^2 I$$

where I is the $[n, n]$ identity matrix,

Gaussian case: $\Sigma_k = \sigma^2 I$

- All classes have a Gaussian distribution of $\mathbf{x}$ where the covariance matrix is identical and diagonal.
- The terms

$$\det(\Sigma_k) = \sigma^{2n}, \qquad \Sigma_k^{-1} = (1/\sigma^2)I$$

  are independent of k, then they are unimportant additive constants that can be ignored.
- Simpler discriminant function

$$g_k(x) = -\frac{\|x - \mu_k\|^2}{2\sigma^2} + \ln P(\mathbf{y} = k) =$$
$$-\frac{(x - \mu_k)^T (x - \mu_k)}{2\sigma^2} + \ln P(\mathbf{y} = k) =$$
$$= -\frac{1}{2\sigma^2}[x^T x - 2\mu_k^T x + \mu_k^T \mu_k] + \ln P(\mathbf{y} = k)$$

- Since the quadratic term $x^T x$ is the same for all k (ignorable additive constant) this is equivalent to a linear discriminant function

$$g_k(x) = w_k^T x + w_{k0}$$

where w_k is a $[n, 1]$ vector

$$w_k = \frac{1}{\sigma^2}\mu_k$$

and

$$w_{k0} = -\frac{1}{2\sigma^2}\mu_k^T \mu_k + \ln P(\mathbf{y} = k)$$

is the *bias* or threshold.

Decision boundary

In the two-classes problem, the decision boundary (i.e. the set of points where $g_1(x) = g_2(x)$) is given by the hyperplane having equation

$$w^T(x - x_0) = 0$$

where

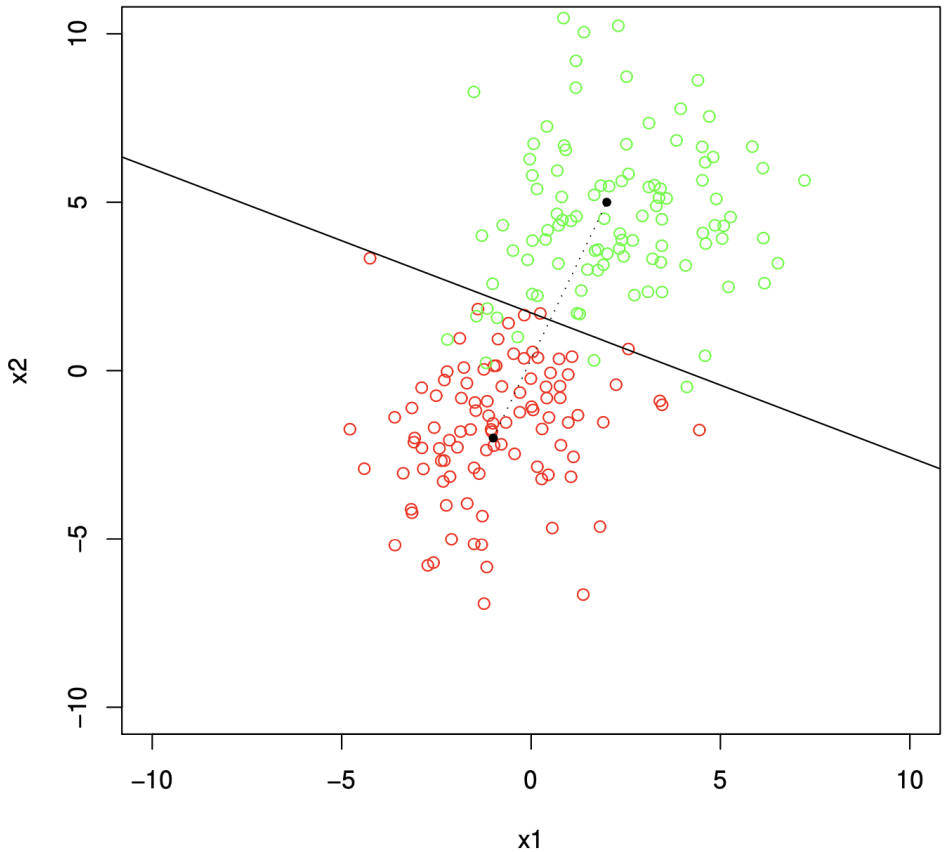
$$w = \mu_1 - \mu_2$$

and

$$x_0 = \frac{1}{2}(\mu_1 + \mu_2) - \frac{\sigma^2}{\|\mu_1 - \mu_2\|^2} \ln \frac{\text{Prob}\{\mathbf{y} = 1\}}{\text{Prob}\{\mathbf{y} = 2\}}(\mu_1 - \mu_2)$$

This equation defines a hyperplane through the point x_0 and orthogonal to the vector w.
See Python script `Linear/discri.py`.

Python script `discri.py`



Note that the boundary line is perpendicular to the vector joining the means of the two class-conditional distributions.

Uniform prior case

- If the prior probabilities $P(\mathbf{y} = k)$ are the same for all the K classes, then the term $\ln P(\mathbf{y} = k)$ is an unimportant additive constant that can be ignored.
- In this case, it can be shown that the optimum decision rule is a *minimum distance classifier*.
- This means that in order to classify an input x, it measures the Euclidean distance $\|x - \mu_k\|^2$ from x to each of the K mean vectors, and assign x to the category of the nearest mean.
- For the more generic case $\Sigma_k = \Sigma$, the discriminant rule is based on minimizing the Mahalanobis distance:

$$\hat{c}(x) = \arg\min_k (x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$$

LDA parametric identification

$$\widehat{\text{Prob}}\{\mathbf{y} = c_k\} = \frac{N_k}{N}$$

$$\hat{\mu}_k = \frac{\sum_{i:y_i=c_k} x_i}{N_k}$$

$$\hat{\Sigma} = \frac{\sum_{k=1}^{K} \sum_{i:y_i=c_k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{N - K}$$

where N_k is the number of samples labeled with the class c_k and the covariance estimator is also known as *pooled covariance*.

From Duda-Hart book

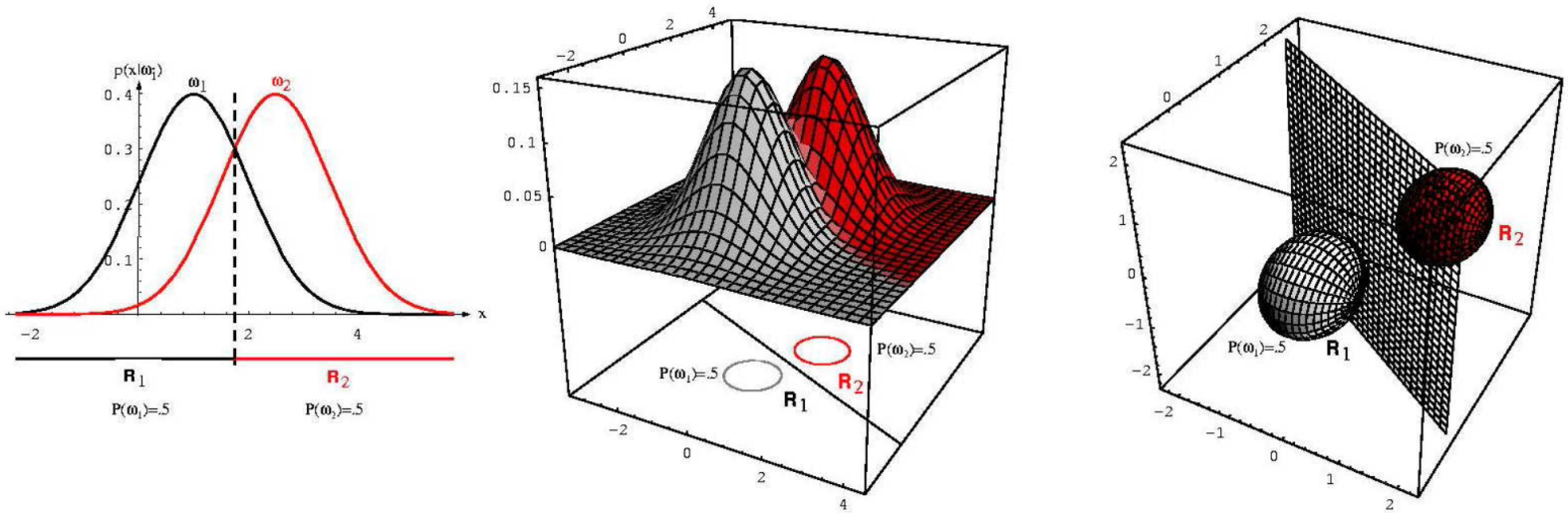


Figure 2.10: If the covariances of two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these 1-, 2-, and 3-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the 3-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$.

Hyperplanes

Consider $x \in \mathbb{R}^n$ and hyperplane

$$h(x, \beta) = \beta_0 + x^T \beta = 0, \quad \beta \in \mathbb{R}^n$$

If $n = 2$ this is a line.

- since for any x_1 and x_2 on the hyperplane we have $(x_1 - x_2)^T \beta = 0$, the normal to the hyperplane is[1]

$$\beta^* = \frac{\beta}{\|\beta\|}$$

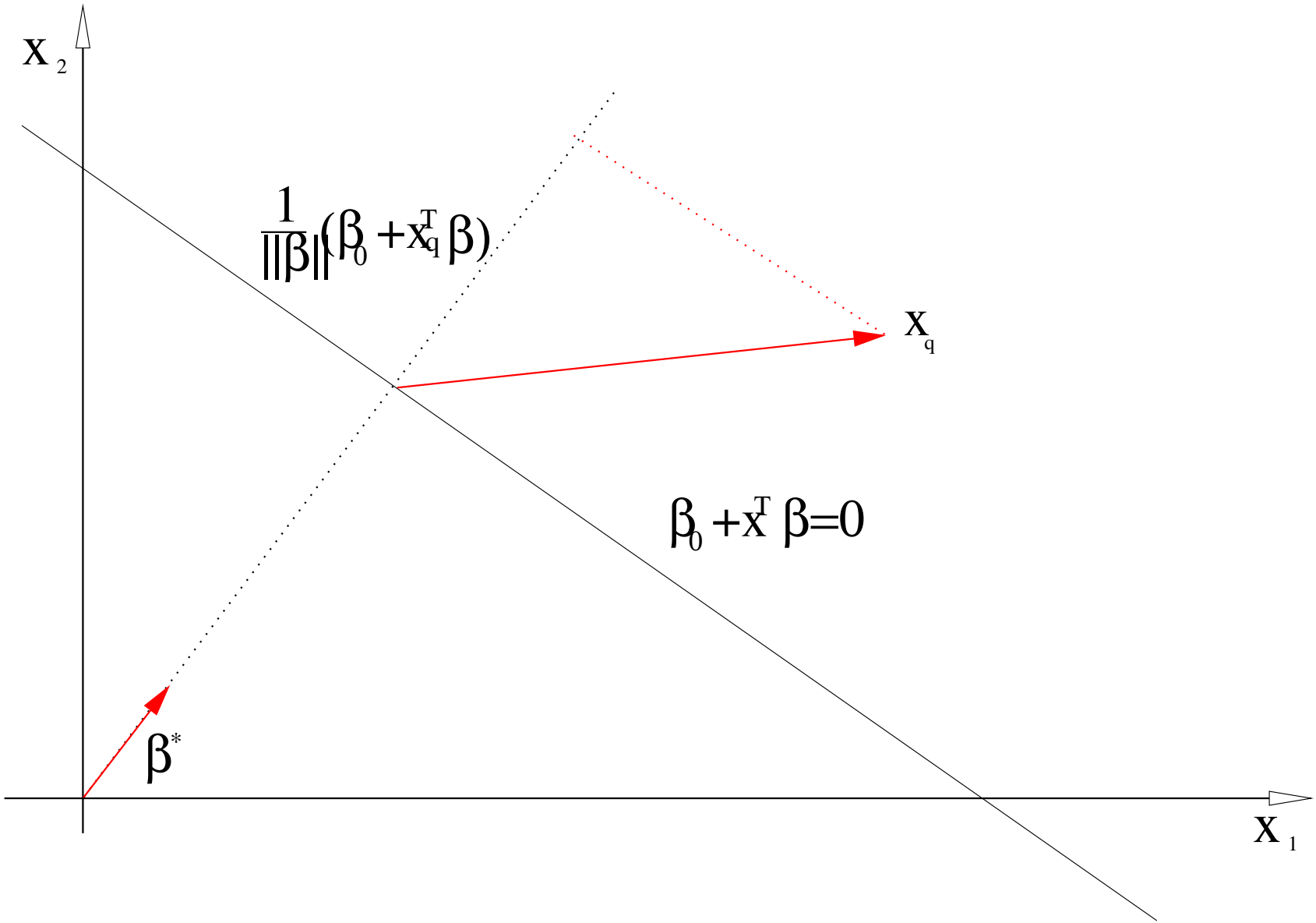
- signed Euclidean distance of any point x to the hyperplane is

$$(x - x_0)^T \beta^* = \frac{x^T \beta - x_0^T \beta}{\|\beta\|} = \frac{1}{\|\beta\|}(x^T \beta + \beta_0)$$

where x_0 is a point on the hyperplane $(\beta_0 + x_0^T \beta = 0)$

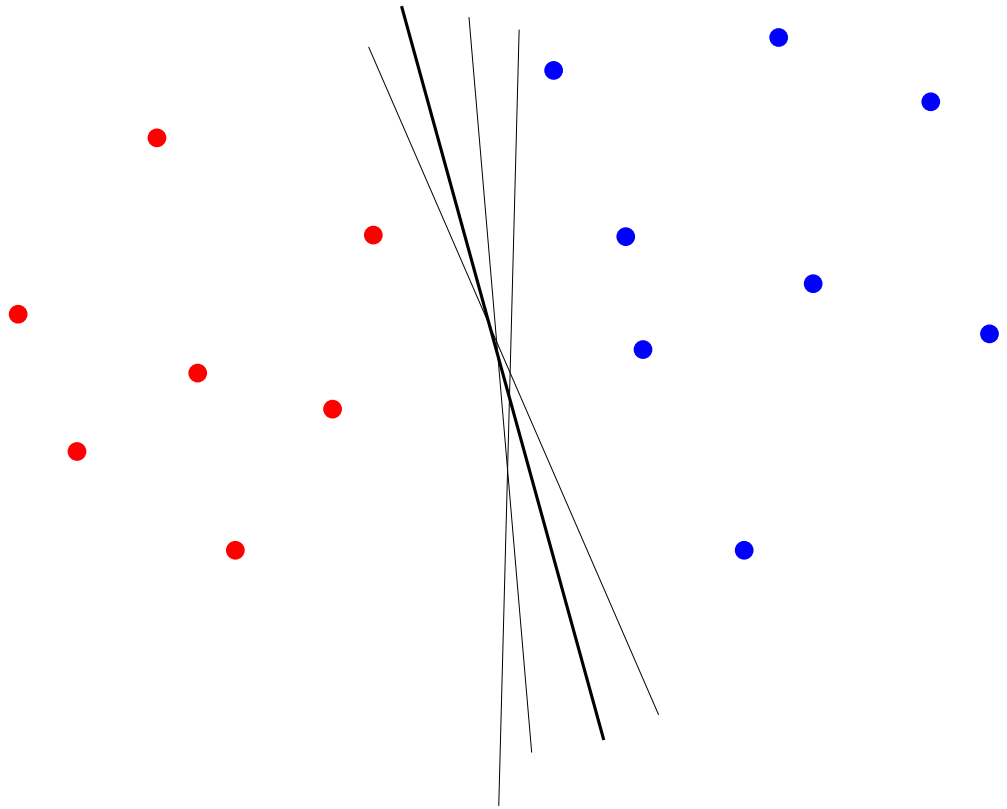
[1] 2 vectors are orthogonal if their dot product is zero.

Hyperplane



Separating hyperplane

The figure shows points in two classes in $\mathbb{R}^2$.



Data can be separated by a linear boundary but infinitely many possible *separating hyperplanes*.

Perceptron (1950)

- Sign of the linear combination $h(x, \beta) = \beta_0 + \beta^T x$ return binary classification.
- Class returned by a perceptron for x_q is

$$\text{sign}(\beta_0 + x_q^T \beta) = \begin{cases} 1 & \text{if } \beta_0 + x_q^T \beta > 0 \\ -1 & \text{if } \beta_0 + x_q^T \beta < 0 \end{cases}$$

- For all well-classified points in the training set

$$y_i(x_i^T \beta + \beta_0) > 0$$

- Misclassifications occur when

$$\begin{cases} y_i = 1 & \text{but } \beta_0 + \beta^T x_i < 0 \\ y_i = -1 & \text{but } \beta_0 + \beta^T x_i > 0 \end{cases}$$

Perceptron (II)

- Parametric identification: minimize the positive quantity

$$R_{\text{emp}}(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

  where $\mathcal{M}$ is the subset of misclassified points.
- Non-negative quantity and proportional to the distance of the misclassified points to the hyperplane.
- Since the gradients are

$$\frac{\partial R_{\text{emp}}(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i, \qquad \frac{\partial R_{\text{emp}}(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

  a gradient descent minimization procedure can be adopted.

Perceptron limitations (III)

- Separable data: many possible solutions depending on the initialization of the gradient method.
- Non separable data: algorithm will not converge.
- Also for a separable problem the convergence of the gradient minimization can be very slow.

Possible solution: optimal separating hyperplane in the SVM approach.

Optimal separating hyperplane

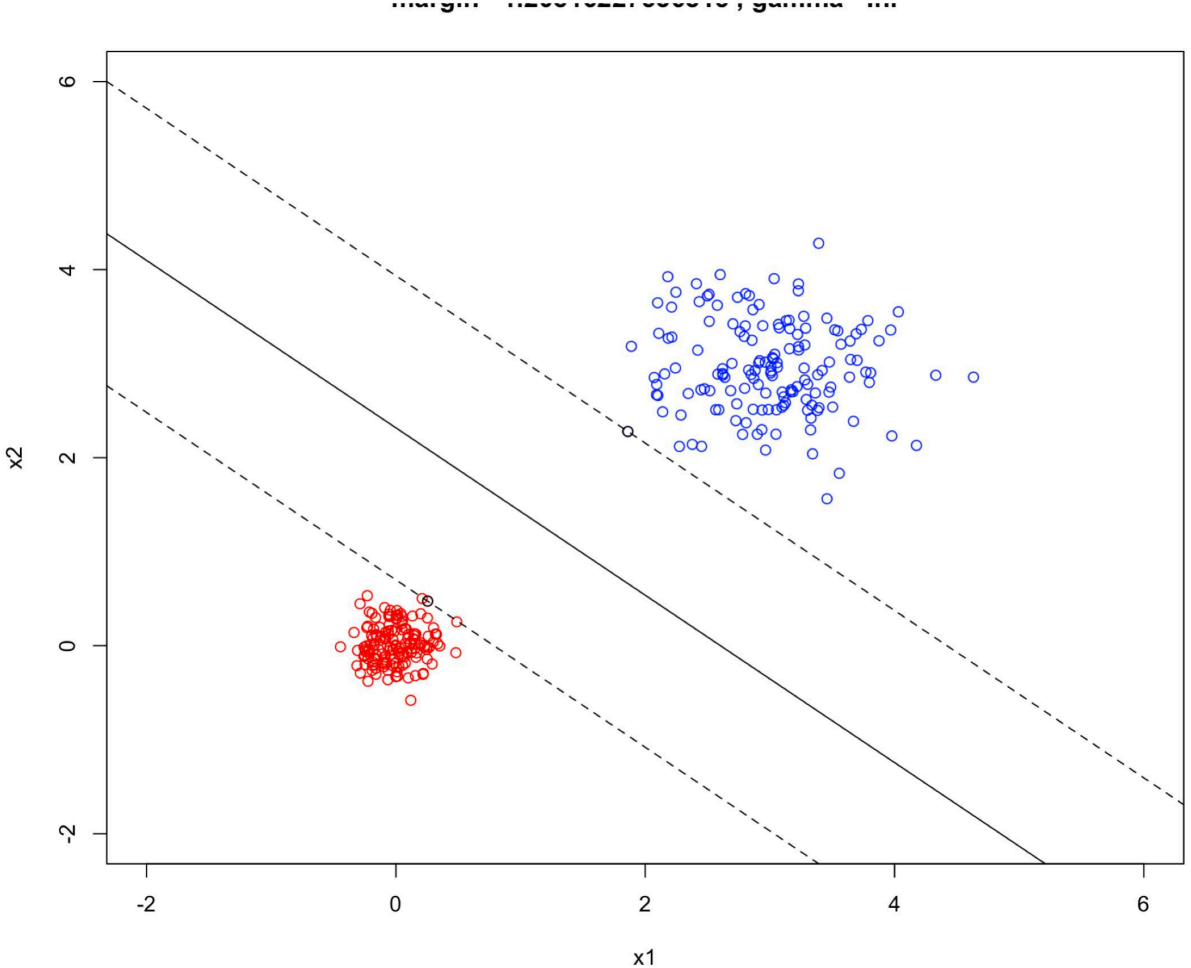
- Optimal separating hyperplane: it separates two classes by *maximizing the distance to the closest point from either class* (also known as the **margin**).
- Unique solution to the separating hyperplane problem
- More robust
- Search for the optimal hyperplane is modeled as the optimization problem

$$\max_{\beta,\beta_0} C$$

$$\text{subject to } \frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq C \quad \text{for } i = 1, \ldots, N$$

- Constraint ensures that all the points are at least a distance C from the decision boundary β, β_0.
- We seek the largest C that satisfies the constraints and the associated parameters: quadratic program that can be solved by standard convex optimisation techniques.

Optimal separating hyperplane



Naive Bayes classifier

- Naive Bayes (NB) classifier has shown in some domains an accuracy comparable to that of neural networks and decision tree learning.
- Classification task with n discrete inputs and a random output categorical variable $\mathbf{y}$ in $\{c_1, \ldots, c_K\}$.
- Bayes' optimal classifier should return

$$c^*(x) = \arg \max_{k=1,\ldots,K} \text{Prob}\{\mathbf{y} = c_k | x\}$$

- Using Bayes' theorem

$$c^*(x) = \arg \max_{k=1,\ldots,K} \frac{\text{Prob}\{x|\mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}}{\text{Prob}\{x\}} =$$
$$= \arg \max_{k=1,\ldots,K} \text{Prob}\{x|\mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}$$

Naive Bayes classifier

- Prob $\{\mathbf{y} = c_k\}$ easy to estimate.
- Prob $\{x|\mathbf{y} = c_k\}$ much harder to estimate.
- NB **assumption**: inputs are conditionally independent given the target:

$$\text{Prob}\{x|\mathbf{y} = c_k\} = \text{Prob}\{x_1, \ldots, x_n|\mathbf{y} = c_k\} = \prod_{j=1}^{n} \text{Prob}\{x_j|\mathbf{y} = c_k\}$$

- NB classification:

$$c_{NB}(x) = \arg \max_{k=1,\ldots,K} \text{Prob}\{\mathbf{y} = c_k\} \prod_{j=1}^{n} \text{Prob}\{x_j|\mathbf{y} = c_k\}$$

- Discrete inputs x_j: estimation of Prob $\{x_j|\mathbf{y} = c_k\}$ from frequencies of the occurrences of the different values of x_j for a class c_k.

Example

Obs	$x_{.1}$	$x_{.2}$	$x_{.3}$	y
1	P.LOW	P.HIGH	N.HIGH	P.HIGH
2	N.LOW	P.HIGH	P.HIGH	N.HIGH
3	P.LOW	P.LOW	N.LOW	P.LOW
4	P.HIGH	P.HIGH	N.HIGH	P.HIGH
5	N.LOW	P.HIGH	N.LOW	P.LOW
6	N.HIGH	N.LOW	P.LOW	N.LOW
7	P.LOW	N.LOW	N.HIGH	P.LOW
8	P.LOW	N.HIGH	N.LOW	P.LOW
9	P.HIGH	P.LOW	P.LOW	N.LOW
10	P.HIGH	P.LOW	P.LOW	P.LOW

What is the NB classification for the query $\{x$=[N.LOW,N.HIGH,N.LOW] $\}$?

Example

$$\text{Prob}\{\mathbf{y} = P.HIGH\} = 2/10, \quad \text{Prob}\{\mathbf{y} = P.LOW\} = 5/10$$
$$\text{Prob}\{\mathbf{y} = N.HIGH\} = 1/10, \quad \text{Prob}\{\mathbf{y} = N.LOW\} = 2/10$$
$$\text{Prob}\{x_{.1} = N.LOW|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{x_{.1} = N.LOW|\mathbf{y} = P.LOW\} = 1/5$$
$$\text{Prob}\{x_{.1} = N.LOW|\mathbf{y} = N.HIGH\} = 1/1, \quad \text{Prob}\{x_{.1} = N.LOW|\mathbf{y} = N.LOW\} = 0/2$$
$$\text{Prob}\{x_{.2} = N.HIGH|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{x_{.2} = N.HIGH|\mathbf{y} = P.LOW\} = 1/5$$
$$\text{Prob}\{x_{.2} = N.HIGH|\mathbf{y} = N.HIGH\} = 0/1, \quad \text{Prob}\{x_{.2} = N.HIGH|\mathbf{y} = N.LOW\} = 0/2$$
$$\text{Prob}\{x_{.3} = N.LOW|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{x_{.3} = N.LOW|\mathbf{y} = P.LOW\} = 3/5$$
$$\text{Prob}\{x_{.3} = N.LOW|\mathbf{y} = N.HIGH\} = 0/1, \quad \text{Prob}\{x_{.3} = N.LOW|\mathbf{y} = N.LOW\} = 0/2$$

$$\text{Prob}\{\mathbf{y} = \text{P.H}|\mathbf{x} = [\text{N.L,N.H,N.L}]\} =$$
$$= \frac{P(\mathbf{x}_{.1} = \text{N.L}|\mathbf{y} = \text{P.H})P(\mathbf{x}_{.2} = \text{N.H}|\mathbf{y} = \text{P.H})P(\mathbf{x}_{.3} = \text{N.L}|\mathbf{y} = \text{P.H})P(\mathbf{y} = \text{P.H})}{P(\mathbf{x})}$$
$$c_{NB}(x) = \arg \max_{P.H,P.L,N.H,N.L}$$
$$\{0 * 0 * 0 * 2/10, 1/5 * 1/5 * 3/5 * 5/10, 1 * 0 * 1 * 1/10, 0 * 0 * 0 * 2/10\} = P.LOW$$

A $\mathcal{K}$NN classifier

Labeled training set $[N, n]$ and classification required for a $[1, n]$ *query* point.

$\mathcal{K}$NN classification:

1. Compute the distance between the query and the training samples according to a predefined metric.
2. Rank the neighbors on the basis of their distance to the query.
3. Select a subset of the $\mathcal{K}$ nearest neighbors. Each of these neighbors has an associated class.
4. Return the class which characterizes the majority of the $\mathcal{K}$ nearest neighbors.

Evaluation of a classifier

- **Error rate** or **misclassification rate**: proportion of test samples misclassified by the rule. Most intuitive measure of performance.
- Misclassification error is not necessarily the most appropriate criterion. It assumes that the costs of different types of misclassification are equal.
- **Confusion matrix**: when there are only a few or a moderate number of classes, it is the most convenient way of summarising the classifier performance.

Confusion matrix in a two-class problem

N test classifications with N_P examples of class 1 (or positive) and N_N examples of class 0 (or negative).

	Negative (0)	Positive (1)	
Classified as negative	T_N	F_N	$\hat{N}_N$
Classified as positive	F_P	T_P	$\hat{N}_P$
	N_N	N_P	N

- F_P is the number of False Positives
- F_N is the number of False Negatives
- N_N/N is an estimator of the a priori probability of class 0.
- N_P/N is an estimator of the a priori probability of class 1.

Balanced Error Rate

- If two classes are not balanced the misclassification error rate

$$ER = \frac{F_P + F_N}{N}$$

  can lead to a too optimistic interpretation of the rate of success
- If $N_P = 90$ and $N_N = 10$, the naive classifier always returning the positive class has $ER = 0.1$ since $F_N = 0$ and $F_P = 10$.
- In unbalanced cases, better to use balanced error rate

$$BER = \frac{1}{2}\left(\frac{F_P}{T_N + F_P} + \frac{F_N}{F_N + T_P}\right)$$

  which averages errors on each class.

Specificity and sensitivity

- **Sensitivity** (true positive rate or recall): the ratio (to be maximised)

$$SE = \frac{T_P}{T_P + F_N} = \frac{T_P}{N_P} = \frac{N_P - F_N}{N_P} = 1 - \frac{F_N}{N_P}, \quad 0 \leq SE \leq 1$$

  It increases by reducing the number of false negatives.

- **Specificity** (true negative rate): the ratio (to be maximized)

$$SP = \frac{T_N}{F_P + T_N} = \frac{T_N}{N_N} = \frac{N_N - F_P}{N_N} = 1 - \frac{F_P}{N_N}, \quad 0 \leq SP \leq 1$$

  It increases by reducing the number of false positive.

Specificity and sensitivity (II)

Sensitivity is the proportion of positive samples classified as positive. Specificity is the proportion of negative samples classified as negative.

Trade-off:

- Classifier always returning 0: $\hat{N}_P = 0, \hat{N}_N = N$, $F_P = 0$, $T_N = N_N$ and $SP = 1$ but $SE = 0$.
- Classifier always returning 1: $\hat{N}_P = N, \hat{N}_N = 0$, $F_N = 0$, $T_P = N_P$ and $SE = 1$ but $SP = 0$.

False Positive and False Negative Rate

- False Positive Rate:

$$FPR = 1 - SP = 1 - \frac{T_N}{F_P + T_N} = \frac{F_P}{F_P + T_N} = \frac{F_P}{N_N}, \qquad 0 \leq FPR \leq 1$$

  It decreases by reducing the number of false positive.

- False Negative Rate:

$$FNR = 1 - SE = 1 - \frac{T_P}{T_P + F_N} = \frac{F_N}{T_P + F_N} = \frac{F_N}{N_P} \quad 0 \leq FNR \leq 1$$

  It decreases by reducing the number of false negatives.

Predictive value

- Positive Predictive value: the ratio (to be maximized)

$$PPV = \frac{T_P}{T_P + F_P} = \frac{T_P}{\hat{N}_P}, \qquad 0 \leq PPV \leq 1$$

  This quantity is also called *precision* in information retrieval.

- Negative Predictive value: the ratio (to be maximized)

$$PNV = \frac{T_N}{T_N + F_N} = \frac{T_N}{\hat{N}_N}, \qquad 0 \leq PNV \leq 1$$

- False Discovery Rate: the ratio (to be minimized)

$$FDR = \frac{F_P}{T_P + F_P} = \frac{F_P}{\hat{N}_P} = 1 - PPV, \qquad 0 \leq FDR \leq 1$$

F_1 score

- F_1 score: harmonic mean of precision and recall (to be maximised)

$$F_1 = \frac{2 \cdot \mathrm{TPR} \cdot \mathrm{PPV}}{\mathrm{TPR} + \mathrm{PPV}} = \frac{2T_P}{2T_P + F_P + F_N}, \qquad 0 \leq F_1 \leq 1$$

Link between FPR, FNR and PPV

It can be shown that that, once set $p = \text{Prob}\{\mathbf{y} = 1\}$

$$\text{FPR} = \frac{p}{1-p}\frac{1-\text{PPV}}{\text{PPV}}(1-\text{FNR})$$

Receiver Operating Characteristic (ROC) curve

- Plot of true positive rate (i.e. sensitivity or power) vs. false positive rate (1- specificity) for different classification thresholds.
- ROC visualizes the probability of detection vs. the probability of false alarm.
- Different points on the curve correspond to different thresholds used in the classifier.
- A perfect ROC curves would follow the two axes.
- Real-life classification rules produce ROC curves which lie between these two extremes.

ROC curve

- If the classifier is a random guesser

$$P(\hat{\mathbf{y}} = 1|\mathbf{y} = 1) = P(\hat{\mathbf{y}} = 1|\mathbf{y} = 0)$$

  then

$$T_P/N_P = F_P/N_N$$

  i.e. for each threshold, we have the same proportion of true positive and false positives
- ROC curve of a random classifier follows the bissetrix line.
- A random classifier does not separate the classes at all.
- By comparing ROC curves one can study relationships between classifiers: the area under the ROC curve (AUROC) is an easy way to summarize the curve to a number.

ROC example

Non separable tasks: $p(x|+) \sim \mathcal{N}(1,1)$ and $p(x|-) \sim \mathcal{N}(-1,1)$.

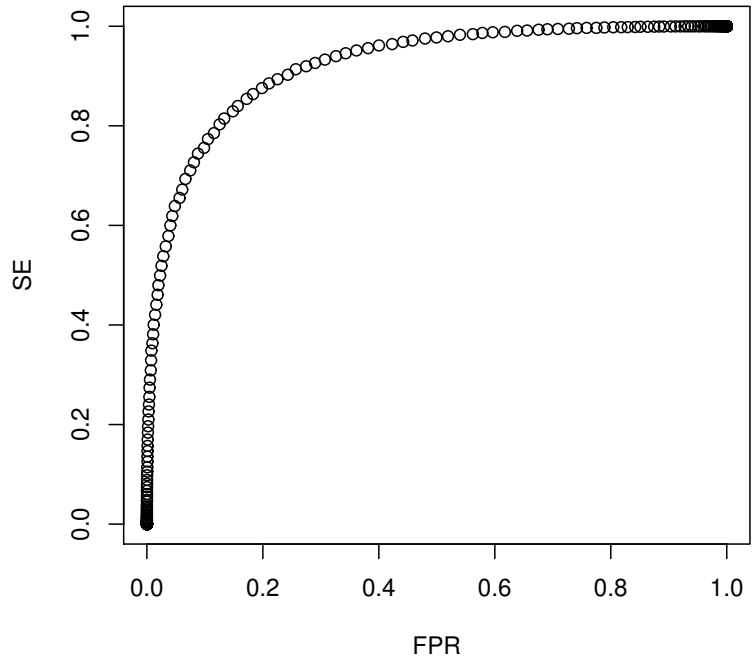
Classification rule:

$$\begin{cases} + & \textit{if } \text{x}>\text{THR} \\ - & \textit{if } \text{x}<\text{THR} \end{cases}$$

where THR is a threshold.

- If $THR = -\infty$, all the examples are classed as positive: $T_N = F_N = 0$ which implies $SE = \frac{T_P}{N_P} = 1$ and $FPR = \frac{F_P}{F_P + T_N} = 1$.
- If $THR = \infty$, all the examples are classed as negative: $T_P = F_P = 0$ which implies $SE = 0$ and $FPR = 0$.

ROC example



A measure to summarize the ROC curve is AUROC, i.e. the area under the ROC curve

Point $[0, 0]$: classifier never issuing a positive classification

Point $[1, 1]$: classifier unconditionally issuing a positive classification

Precision-recall curves

- Sensitivity (also known as recall) is the probability that a sample is classified as positive given that is positive:

$$P(\hat{\mathbf{y}} = 1 | \mathbf{y} = 1)$$

- Precision is the probability that a sample is positive given that it has been classified as positive:

$$P(\mathbf{y} = 1 | \hat{\mathbf{y}} = 1)$$

- Precision is dependent on the probability a priori of the positive class.
- In largely unbalanced problems (e.g. few positive classes like in fraud detection), the PR curve is more informative than the AUC.

Fraud detection example

Fraud detection problem: $N_P = 100$ frauds out of $N = 2 \cdot 10^6$ transactions

Two algorithms: Algo1 returns 100 alerts and Algo2 returns 1000 alerts.

Algorithm 1: 90 relevant out of 100 identified

	Real - (gen)	Real + (fr)	
Classified - (gen)	$1,999,890$	10	$1,999,900$
Classified + (fr)	10	90	100
	$1,999,900$	100	$2 \cdot 10^6$

Fraud detection example

Algorithm 2: 90 relevant out of 1000 alerts

	Real - (gen)	Real + (fr)	
Classified - (gen)	$1,998,990$	10	$1,999,000$
Classified + (fr)	910	90	1000
	$1,999,900$	100	$2 \cdot 10^6$

Though the TPR (recall) is identical, algorithm 1 is preferable because of the smaller number of false positive.

- A1:
  - TPR$=T_P/N_P = 90/100 = 0.9$,
  - FPR$= F_P/N_N = 10/1,999,900 = 0.00000500025$
- A2:
  - TPR$=90/100 = 0.9$,
  - FPR$=$ FPR$=910/1,999,900 = 0.00045502275$
- The FPR difference between A1 and A2 (visible in the ROC curve) is negligible
- A1: PR$=T_P/(T_P + F_P) = 90/100 = 0.9$, recall$=0.9$
- A2: PR$=90/1000 = 0.09$, recall$= 0.9$
- The precision difference (0.81) between A1 and A2 (visible in the PR curve) is much more apparent

Multi-class problems

- So far we have simplified our analysis by limiting to consider binary classification tasks.
- However, we often encounter multi-class problems in real problems (notably bioinformatics).
- There exists several strategies to extend binary classifiers to handle multi-class tasks $\mathbf{y} \in \{c_1, \ldots, c_k\}$.

Multi-class problems

- **One-versus-the rest**:
  - Training: a binary classifier per class c_k that separates it from the rest: k binary classifiers.
  - Classification: outputs of the k classifiers are computed: if there is a single consistent output, this is the returned class. Otherwise, one of the k classes is selected randomly.
- **Pairwise**:
  - Training: a classifier is trained for each pair of classes: $k(k-1)/2$ independent binary classifiers.
  - Classification: outputs of the $k(k-1)/2$ classifiers is computed, which is viewed as a vote. The majority class is returned otherwise ties arebroken randomly.

Multi-class problems

- **Coding**: k classes coded by a binary vector of size d.
  - Training: Each binary classifier is designed to produce one of 0 and 1 as the class label: $d = \lceil \log_2 k \rceil$ binary classifiers
  - Classification: output of d classifiers returns a vector in $\{0, 1\}^d$: the class with the smallest Hamming distance (the number of disagreements) to the *output word* is selected.

Example: $k = 8$ classes: $d = \log_2 8 = 3$ binary classifiers are sufficient.

	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
c_1	0	0	0
c_2	0	0	1
c_3	0	1	0
c_4	0	1	1
c_5	1	0	0
c_6	1	0	1
c_7	1	1	0
c_8	1	1	1

Multi-class problems

Given $k > 2$ classes we need

- One-versus-the rest: k binary classifiers
- Pairwise: $k(k-1)/2$ binary classifiers
- Coding: $\lceil \log_2 k \rceil$ binary classifiers

Final remarks

Beyond simplifications...

Real learning tasks have often challenging configurations we have neglected for the sake of simplicity

- Big data
- Unbalancedness of the classification task
- Heteroskedasticity
- Mixed nature of the variables (real, binary and categorical, potentially with many levels)
- Missing values
- Large number of irrelevant (or redundant) variables
- Skewed and long tailed distributions of inputs and outputs
- Batch vs online nature of the learning task
- Nonstationarity, concept drift
- Existing domain knowledge in qualitative form
- Demand for interpretability

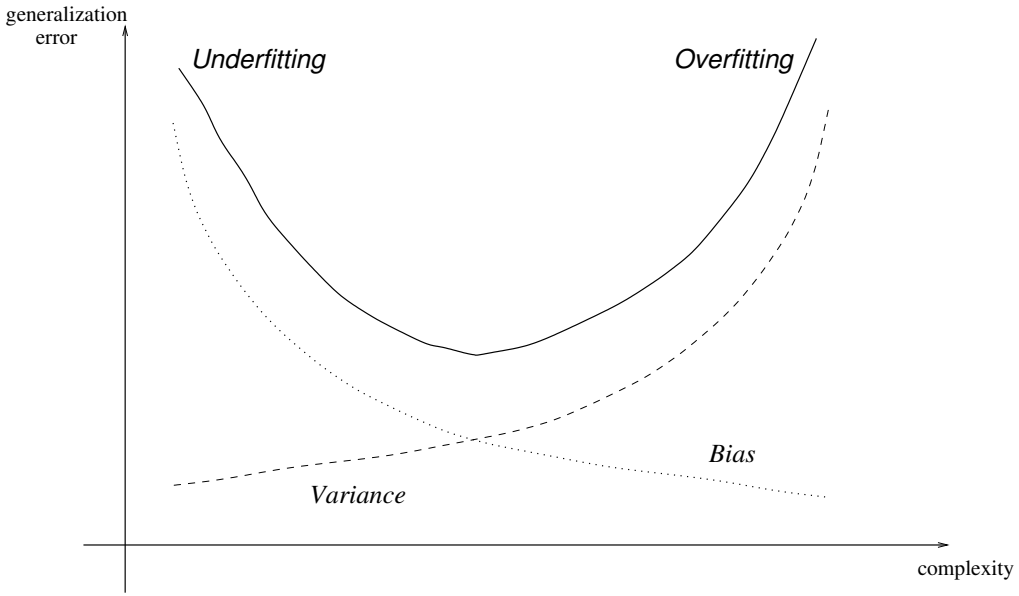
The pillars of supervised learning

1. **Dependence**: a **target y** is dependent on an **input x** if **x** brings information about **y** (i.e. knowing **x** reduces the uncertainty of **y**)

$$\mathbf{y} = f(x) + \mathbf{w}$$

2. **Estimation**: learners are estimators submitted to the bias/variance trade-off (i.e. not necessarily the most complex learner that generalises the best)
3. **Causality implies dependency but not the other way round**: "what if I observe" is not "what if I intervene"...

Bias/variance trade-off



Generating process (nature):	Learner:
Noise $\sigma^2_{\mathbf{w}} \stackrel{+}{\rightarrow}$ V	Complexity (e.g. # parameters, VC dim) $\stackrel{+}{\rightarrow}$ V, $\stackrel{-}{\rightarrow}$ B
Sample size $N \stackrel{-}{\rightarrow}$ V	Regularisation (dropout) $\stackrel{-}{\rightarrow}$ V, $\stackrel{+}{\rightarrow}$ B
Dimension $n \stackrel{+}{\rightarrow}$ V	Feature selection, hyperparameter search $\stackrel{+}{\rightarrow}$ V, $\stackrel{-}{\rightarrow}$ B
Nonlinearity $f \stackrel{+}{\rightarrow}$ B	Combination, averaging $\stackrel{-}{\rightarrow}$ V, $\stackrel{=}{\rightarrow}$ B
Nonstationarity $\stackrel{+}{\rightarrow}$ B	