Machine Learning 1

1. Introduction to Machine Learning

Machine learning is a field that lies at the intersection of statistics, probability, computer science, and optimization. The main goal is to explore **automatic methods for inferring models from data** (for example: finding structure, making predictions).

Examples of learning tasks:

- SUPERVISED LEARNING: uses labeled data.
  - **Classification:** predicting a class or category to each example; note multi-label, probabilistic generalizations.
  - **Regression:** predicting a real value for each example; note multi-variable generalization.
- UNSUPERVISED LEARNING: does not use or have data labels.
  - **Clustering:** discovering homogeneous groups (clusters) in data.
  - **Dimensionality reduction:** finding lower-dimensional data representations.
  - **Density estimation:** estimating the probabilistic mechanism that generates data.
  - **Novelty detection:** finding anomalous/novel/outlying data.
- SEMI-SUPERVISED LEARNING: uses partly labeled data.
  - **Ranking:** ordering examples according to some criterion.
  - **Reinforcement:** delayed rewarding.
- TRANSFER LEARNING: learning in a new task through the transfer of knowledge from a related task that has already been learned.

Useful probability and statistics facts

- **Central Limit Theorem:**

  If $X_1, \ldots, X_n$ are *independent identically distributed random variables*, with $\mathbb{E}[X_i] = \mu$ and $\mathrm{Var}(X_i) = \sigma^2$, then the sample mean

  $$\frac{X_1 + \ldots + X_n}{n} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

  approaches a normal distribution as $n \to \infty$.

- **Product rule:**

  If $X_1, \ldots, X_n$ have a joint probability distribution $p(X_1, \ldots, X_n)$, then we can factorize the distribution as the product

  $$p(X_1, \ldots, X_n) = p(X_1) \prod_{i=2}^{n} p(X_i | X_1, \ldots, X_{i-1}).$$

- **Bayes Theorem:**

  $$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_j P(A|B_j)P(B_j)} = \frac{P(A|B_i)P(B_i)}{P(A)}.$$

- **Bayes formula for densities:**

  In a data analysis context, θ is a parameter vector and the following equality holds:

$$\pi_{\text{POST}}(\theta|\text{data}) = \frac{\pi_{\text{LIK}}(\text{data}|\theta) \cdot \pi_{\text{PRIOR}}(\theta)}{\int_\Theta \pi_{\text{LIK}}(\text{data}|\theta) \cdot \pi_{\text{PRIOR}}(\theta) d\theta}.$$

  This can also be expressed loosely as

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} = \frac{P(D|\theta)P(\theta)}{\int_\Theta P(D|\theta)P(\theta)d\theta},$$

  where D is the data. This expression gives rise to the notions of **likelihood**, **prior**, **posterior**, and **unconditional (expected likelihood)** distributions:

- $P(\theta)$: prior probability, confidence in θ before observing D.
- $P(D|\theta)$: likelihood, probability of observing D if parameters are θ.
- $P(D)$: expected likelihood of observing data D, also unconditional.
- $P(\theta|D)$: posterior probability, confidence in θ after observing D.

- **Conjugacy:**

  Definition: Suppose a prior distribution $\pi_{\text{PRIOR}}(\theta)$ belongs to a class of parametrized distributions Π. Then the distribution is said to be **conjugate** with respect to a likelihood $\pi_{\text{LIK}}(\cdot|\theta)$ if the posterior distribution $\pi_{\text{POST}}(\theta|\cdot) \in \Pi$.

  Remember that $\pi_{\text{POST}}(\theta|\cdot) \propto \pi_{\text{LIK}}(\cdot|\theta)\pi_{\text{PRIOR}}(\theta)$. For example, Gaussian is conjugate to Gaussian, and Beta is conjugate to Binomial.

Using the posterior:

$$\hat{\theta}_{\text{MAP}} := \text{argmax}_{\theta\in\Theta}\{P(\theta|D)\}\text{: the value of } \theta \text{ that maximizes the posterior.}$$
$$\hat{\theta}_{\text{ML}} := \text{argmax}_{\theta\in\Theta}\{P(D|\theta)\}\text{: the value of } \theta \text{ that maximizes the likelihood.}$$
$$\hat{\theta}_{\text{EV}} := \mathbb{E}[P(\theta|D)] = \int_\Theta P(\theta|D) \cdot P(\theta) d\theta\text{: the expected value of theta.}$$

Inductive bias

Example: complete the following series: $2, 4, 6, 8, \ldots$

Answer 1: 132 (model 1: $f(n) = n^4 - 10n^3 + 35n^2 - 48n + 24$)

Answer 2: 10 (model 2: $f(n) = 2n$)

How can we rule out the more complex model?

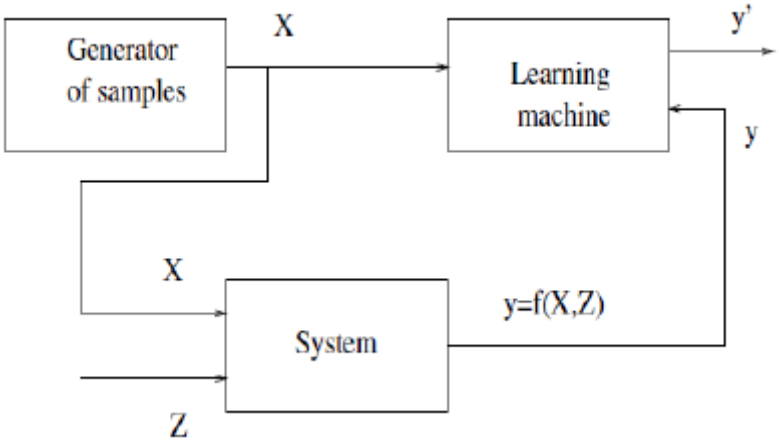
1. Supply more training data: $2, 4, 6, 8, 10, 12, 14, \ldots$
2. Regularize: add penalty to higher-order terms.
3. Reduce the hypothesis space; for example, restrict to quadratic models.

$$\text{Class of functions: } \mathcal{F} := \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}.$$

So, the conclusions are this: based *only* on training data D, there is no means of choosing which function f is better (generalitzation is not *guaranteed*). Thus, we must add control to the **fitting ability** of our methods (complexity control).

$$\text{true error}(f) \leq \text{training error}(f) + \text{complexity of } f.$$

Formulation of ML



X are the measured variables, Z are the unmeasured ones, y is the true function and y', which would be $\hat{y}$, is the modeled function.

Prediction vs. inference

Prediction: produce a good estimate for the predicted variable.

Inference:

1. Which predictors actually affect the target variable?
2. How strong are these dependencies?
3. Are these relationships positive or negative?

Common Tasks

- **Regression:** The goal is to predict some quantitative outcome subject to probabilistic uncertainty.
- **Classification:** The goal is to obtain a model based on available **training data** (*known* examples) with high classification accuracy on unseen *unknown* examples (**test data**), i.e. achieving good **generalization**.
- **Clustering:** The goal is to find homogeneous groups of data and set them apart accordingly. Looks like a very different task from regression or classification, but it's both of them with some added difficulty: it has an inherent large subjectivity.

Why are these tasks stochastic?

We have a (complete) input data object (x, z) and an output data object y. The true relationship is $f_c : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$, that is $f_c(x, z) = y$. When we measure data about f_c, we measure only the x portion of the input variables. Therefore, the relation between x and y becomes stochastic.

Setting up the tasks

There are (at least) two ways of setting up these tasks formally:

Optimization view

$$\min_{\theta \in \Theta} E(\theta) := \frac{1}{n} \sum_{i=1}^{n} l(y_i, f_\theta(x_i)) + \Omega(f_\theta).$$

true error of $f_\theta \leq$ training error of f_θ + complexity of f_θ (empirical risk + regularizer)

$l(y_i, f_\theta(x_i))$ is called the *loss/error function*.

Statistics view

Use Bayes' formula to compute $P(\theta|\text{data})$ and choose one according to this (posterior) distribution.

Many times these two views can yield the same results (which is good!). An example would be LSQ $\equiv$ MaxLik+Gaussian.

The most general description of the data generation mechanism is in terms of the pdf $p(x, y)$ in the joint input-output space: this is the key to generalization.

$$p(x, y) = p(y|x) \cdot p(x), \text{ where } p(x) = \int p(y, x)\, dy.$$

Some techniques use $p(x)$, others do not. The important pdf is $p(y|x)$. *Discriminative* methods use only $p(y|x)$, while *generative* methods use the joint pdf $p(x, y)$.

So, what is a Machine Learning algorithm/technique?

A ML algorithm gets a dataset D and returns a model of D (a representation of D that either gives structure to D or that allows to make predictions on unseen observations), together with an estimation of the model quality. The algorithm itself typically determines the model space $\mathcal{F}$ and the loss function l.

And why are linear models so nice?

We will begin our analyses with linear models and techniques. A model is linear when, up to an invertible mapping, it is **a linear function of its parameters**; $f_\theta(x)$ is linear when is depends linearly on θ, but we do not say anything about x. For example, $f_\theta(x) = \sum_{i=0}^{m} \theta_i \sin(\exp(-x_i^2))$ is linear. A linear model:

- **Is analytically tractable:** we have closed-form solutions or fast convergent iterative methods for the solution.
- **Has a unique solution:** there are no local optima.
- **Is highly interpretable.**
- **Is amenable to inference:** we can ask (and answer) questions about the importance and weight on the target of the different variables.
- Has **user-defined fitting ability**, via the basis functions.
- Is capable of being **regularized:** complicated models can be penalized.

General form of a linear model

A linear model has a general expression as

$$f(x; \theta) = g\left(\theta_0 + \sum_{i=1}^{h} \theta_i \varphi_i(x)\right).$$

The functions φ_i are called **basis functions** (they constitute a *feature map*) and are non-linear wrt x. g is a strictly monotonic function: in Neural Networks, this is called an **activation function**.

On data pre-processing

Each problem requires a different approach in what concerns data cleaning and preparation. This pre-processing procedure is very important because it can have a deep impact on performance; it can easily take us a significant part of the time. So, the important things to take into account on data pre-processing are:

- Treatment of missing, anomalous, and incoherent or incorrect values.

- Coding of non-continuous or non-ordered variables.
- Possible elimination of irrelevant or redundant variables (*feature selection*).
- Creation of new variables that can be useful (*feature extraction*).
- Normalization of the variables (standardization).
- Transformations of the variables (for example, corrections of serious skewness and/or kurtosis)

Non-standard data (images, audio, text...) may need completely *ad hoc* treatments.

2. Linear Data Visualization

Dimensionality reduction

There are two main goals associated to these techniques:

- **Signal representation:** the goal is to represent the data accurately in a lower-dimensional space.
- **Signal classification:** the goal is to enhance the class-discriminatory information in the lower-dimensional space.

Unfortunately, there is no systematic way to generate non-linear transforms, so we will focus on **linear** methods for **feature extraction**:

- PCA: Principal Components Analysis.
- FDA/LDA: Fisher's Discriminant Analysis.
- ICA: Independent Components Analysis.

Principal Components Analysis

This feature extraction method is explained in the `../AD/AD.pdf`.

Fisher's Discriminant Analysis

FDA is a technique for **dimensionality reduction, supervised classification, feature extraction and data visualization**.

Idea: projection of the data onto a lower dimensional linear space, such that the separability of projected data is maximized.

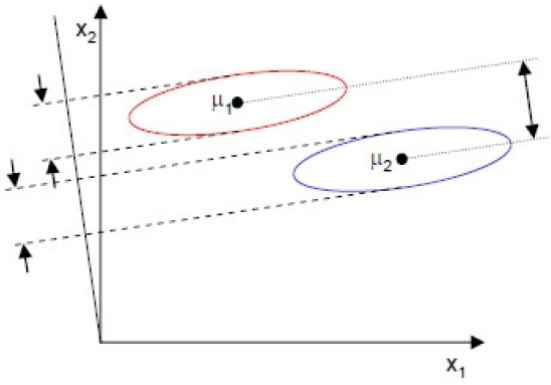
Fisher's idea is to regard **dot product** as the projection y of some $x \in \mathbb{R}^p$ from classes ω_1 or ω_2, via a projection vector w: $y = w^T x \in \mathbb{R}$. In order to find a good projection vector, we need to define a measure of separation between the projections:

$$m_k = \frac{1}{n_k} \sum_{i \in \omega_k} x_i, \qquad k \in \{1, 2\},$$

where $n_1 + n_2 = n$ is the number of examples on every class. We then choose to maximize the *squared* distance between the projected means,

$$(\mu_2 - \mu_1)^2 = (w^T m_2 - w^T m_1)^2 = (w^T (m_2 - m_1))^2.$$

However, the distance between the projected means is not a very good measure since it does not take into account the dispersion (**scatter**) within the classes. The problem is that the covariance matrices for each class are far from being diagonal. We actually want to look for the projection where examples from the same class are projected very close to one another and the projected means are as far apart as possible:



The solution (proposed by R. Fisher) is to maximize a function that represents the difference between the means, normalized by a measure of the within-class scatter:

1. $\forall k$ a class we define the scatter as

$$s_k^2 = \sum_{i \in \omega_k} \left(w^T x_i - \mu_k\right)^2, \qquad k \in \{1, 2\}.$$

2. The total scatter is $s_1^2 + s_2^2$.
3. Fisher's idea was to maximize the following function:

$$J(w) = \frac{(\mu_2 - \mu_1)^2}{s_1^2 + s_2^2}.$$

It can be shown that $J(w)$ can be rewritten as:

$$J(w) = \frac{(\mu_2 - \mu_1)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_W s},$$

where

- $S_B = (m_2 - m_1)(m_2 - m_1)^T$ is the **between-class scatter matrix** (rank 1).
- $S_W = \sum_{i \in \omega_1} (x_1 - m_1)(x_1 - m_1)^T + \sum_{i \in \omega_2} (x_1 - m_2)(x_1 - m_2)^T$ is the **within-class scatter matrix**.

To find the maximum of J we derive and equal to zero,

$$\frac{\partial J}{\partial w} = 0,$$

and upon solving we arrive at the following **generalized eigenvalue problem**:

$$\left(S_W^{-1} S_B\right) w = J(w) w;$$

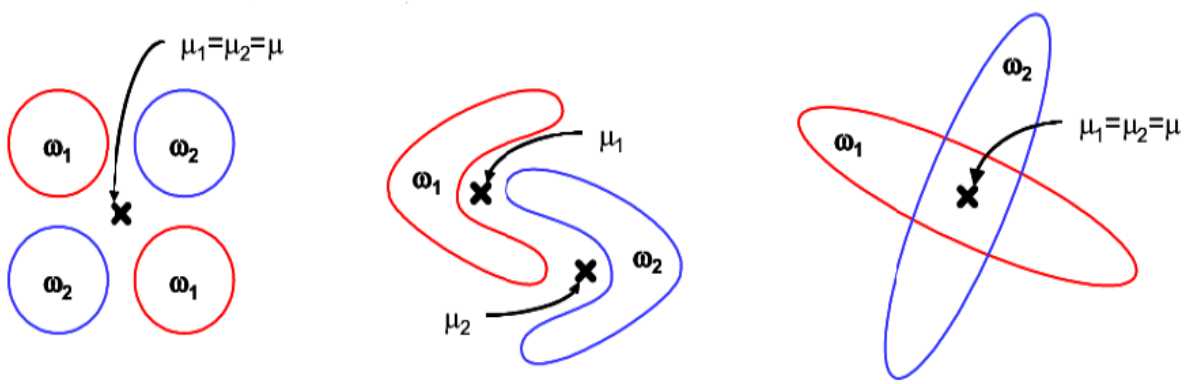
solving it yields $\boxed{\hat{w} = S_W^{-1}(m_1 - m_2),}$ known as **Fisher's Linear Discriminant** (1936), although it is not a discriminant but a specific choice for projection down to one dimension.

FDA generalizes very gracefully for K class problems: the only restriction is that the maximum number of projection directions is $K - 1$. FDA can also be derived as the Maximum Likelihood result for the case of Gaussian class-conditional densities with equal covariance matrices; in this case, it is known as LDA.

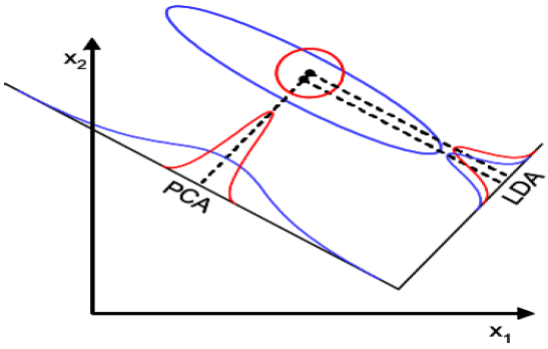
WARNING! FDA is able to extract a maximum of $K - 1$ projection directions, maybe insufficient for complex data. PCA is able to extract d projection directions, but it is not clear how many are necessary.

Counterparts

When will FDA presumably fail? If the classes are far from Gaussian, the FDA projections will not be able to preserve any complex structure; for an example, this image:



FDA will also fail when the discriminatory information is not in the mean but rather in the *variance* of the data (e.g., if $J(w) = 0$); for example,



3. Theory for regression and linear models (I).

The regression framework

Given data $D = \{(x_n, t_n)\}_{n=1,\dots,N}$, where $x_n \in \mathbb{R}^d, t_n \in \mathbb{R}$,

- **Statistics:** estimation of a continuous random variable T conditioned on a random vector X .
- **Mathematics:** estimation of a real function f based on a finite number of *noisy* examples $(x_n, f(x_n))$.

The departing statistical setting is $t_n = f(x_n) + \varepsilon_n$; a model is any approximation of f. We assume ε_n are iid random variables such that $\mathbb{E}[\varepsilon_n] = 0$ and $\mathrm{Var}(\varepsilon_n) = \sigma^2 < \infty$, and that ε_n and x_n are independent variables.

The **risk** of a model y is

$$R(y) := \int_{\mathbb{R}} \int_{\mathbb{R}^d} L(t, y(\boldsymbol{x}))p(t, \boldsymbol{x})\, d\boldsymbol{x}\, dt,$$

where L is a suitable **loss** function that satisfies:

- $L(t, y(\boldsymbol{x})) \geq 0$
- $L(t, y(\boldsymbol{x})) = 0 \iff t = y(\boldsymbol{x})$ (not necessarily in the other direction)
- $L(t, y(\boldsymbol{x}))$ does not increase when $|t - y(\boldsymbol{x})|$ decreases.

L is closely related to the distribution of the noise model ε_n.

Example: if we assume for example that $\varepsilon_n \sim \mathcal{N}\left(0, \sigma^2\right)$, using a maximum likelihood argument it can be shown that the *right* loss function is the **square error**:

$$L_{\mathrm{SE}}(t, y(\boldsymbol{x})) := (t - y(\boldsymbol{x}))^2.$$

The **risk** is therefore

$$R(y) = \int_{\mathbb{R}} \int_{\mathbb{R}^d} (t - y(\boldsymbol{x}))^2 p(t|\boldsymbol{x})p(\boldsymbol{x})\, d\boldsymbol{x}\, dt$$

If we enjoy complete freedom to choose y, the solution is:

$$y^*(\boldsymbol{x}) = \int_{\mathbb{R}} tp(t|\boldsymbol{x})\, dt = f(\boldsymbol{x}),$$

known as the **regression function**. Since $\mathbb{E}[\varepsilon_n] = 0$, we can alternatively express the regression setting by stating that t is a continuous random variable such that $f(\boldsymbol{x}) = \mathbb{E}[t \mid X = \boldsymbol{x}]$.

Claim: $y^*(\boldsymbol{x}) = f(\boldsymbol{x})$.

Proof:

$$\begin{aligned} &y^*(\boldsymbol{x}) = \mathbb{E}[t \mid X = \boldsymbol{x}] = \mathbb{E}[f(\boldsymbol{x}) + \varepsilon \mid X = \boldsymbol{x}] = \\ &= \mathbb{E}[f(\boldsymbol{x}) \mid X = \boldsymbol{x}] + \mathbb{E}[\varepsilon \mid X = \boldsymbol{x}] = f(\boldsymbol{x}) + 0 = \\ &= f(\boldsymbol{x}). \end{aligned}$$ □

In a practical setting, we don't know $p(t|\boldsymbol{x})$. Instead, we have a finite i.i.d. **data sample** of N labeled observations $D = \{(\boldsymbol{x}_n, t_n)\}_{n=1,\dots,N}$, where $\boldsymbol{x}_n \in \mathbb{R}^d, t_n \in \mathbb{R}$. Then, **intuition** tells us to solve for y

$$\min_y \int_{\mathbb{R}^d} (f(\boldsymbol{x}) - y(\boldsymbol{x}))^2 p(\boldsymbol{x})\, d\boldsymbol{x}.$$

This is equivalent to minimizing the risk function; we'll see this in the next part. For now, we must impose restrictions on the possible solutions y, this is, we must restrict the search space to a specific **class of functions** $\mathcal{Y}$.

We can compute an approximation to the true risk, called the **empirical risk**, by averaging the loss function on the available data D:

$$R_{\text{emp}}(y) := \frac{1}{N} \sum_{n=1}^{N} (t_n - y(\boldsymbol{x}_n))^2.$$

This quantity is also known as the **apparent error**. The **Empirical Risk Minimization (ERM)** principle stats that a learning algorithm should choose a hypothesis (model) $\hat{y}$ which minimizes the empirical risk among a predefined class of functions $\mathcal{Y}$:

$$\hat{y} := \operatorname*{argmin}_{y \in \mathcal{Y}} R_{\text{emp}}(y).$$

The quantity $R_{\text{emp}}(\hat{y})$ is known as the **training error**. In theoretical ML, we are very much interested in:

- How this error fluctuates as a function of the data D.
- How far this error is from the true error, this is, to bound $|R_{\text{emp}}(\hat{y}) - R(y)|$; at the very least, to bound $|\mathbb{E}[R_{\text{emp}}(\hat{y})] - R(y)|$.
- How far this error is from the best possible error, this is, to bound $|R_{\text{emp}}(\hat{y}) - R(y^*)|$; at the very least, to bound $|\mathbb{E}[R_{\text{emp}}(\hat{y})] - R(y^*)|$.

Bias-Variance analysis

Recall the assumption that $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$. In this case, using the square error, the risk can be decomposed as:

$$R(y) = \int_{\mathbb{R}} \int_{\mathbb{R}^d} (t - y(\boldsymbol{x}))^2 p(t, \boldsymbol{x})\, d\boldsymbol{x}\, dt = \int_{\mathbb{R}} \int_{\mathbb{R}^d} (t - f(\boldsymbol{x}))^2 p(t, \boldsymbol{x})\, d\boldsymbol{x}\, dt$$
$$+ \int_{\mathbb{R}^d} (f(\boldsymbol{x}) - y(\boldsymbol{x}))^2 p(\boldsymbol{x})\, d\boldsymbol{x} = \sigma^2 + \int_{\mathbb{R}^d} (f(\boldsymbol{x}) - y(\boldsymbol{x}))^2 p(\boldsymbol{x})\, d\boldsymbol{x} = \boxed{\sigma^2 + \text{MSE}(y),}$$

where f is the **regression function**. Therefore, we arrive at $R(y) = \sigma^2 + \text{MSE}(y)$. We can now *forget* about σ^2 and the risk and instead aim at minimizing the $\text{MSE}(y)$:

$$\text{MSE}(y) = \int_{\mathbb{R}^d} (f(\boldsymbol{x}) - y(\boldsymbol{x}))^2 p(\boldsymbol{x})\, d\boldsymbol{x}.$$

A **learning algorithm** for **regression** is a procedure that, given data D and the search space $\mathcal{Y}$, outputs a model $y_D \in \mathcal{Y}$ that aims at minimizing $\text{MSE}(y)$.

Consider now one particular $\boldsymbol{x}_0$; different D will produce different y_D and therefore different predictions $y_D(\boldsymbol{x}_0)$. Let us concentrate on the quantity $(f(\boldsymbol{x}_0) - y(\boldsymbol{x}_0))^2$: we wish to eliminate the dependence on D. Therefore, we investigate its expected value, $\mathbb{E}\left[(f(\boldsymbol{x}_0) - y(\boldsymbol{x}_0))^2\right]$, taking over all possible D of size N. If we develop a little more their formulas,

$$\mathbb{E}\left[(f(\boldsymbol{x}_0) - y(\boldsymbol{x}_0))^2\right] = (f(\boldsymbol{x}_0) - \mathbb{E}[y_D(\boldsymbol{x}_0)])^2 + \mathbb{E}\left[(y_D(\boldsymbol{x}_0) - \mathbb{E}[y_D(\boldsymbol{x}_0)])^2\right].$$

We can interpret these summands as $f(\boldsymbol{x}_0) - \mathbb{E}[y_D(\boldsymbol{x}_0)] = \text{Bias}(y_D(\boldsymbol{x}_0))$, and $\mathbb{E}\left[(y_D(\boldsymbol{x}_0) - \mathbb{E}[y_D(\boldsymbol{x}_0)])^2\right] = \text{Var}(y_D(\boldsymbol{x}_0))$. Then, the formula is more clearly stated as

$$\text{MSE}(y_D(\boldsymbol{x}_0)) = \text{Bias}^2(y_D(\boldsymbol{x}_0)) + \text{Var}(y_D(\boldsymbol{x}_0)),$$

and the **risk** can be expressed as a sum of three summands:

$$R(y_D(\boldsymbol{x}_0)) = \sigma^2 + \mathrm{Bias}^2(y_D(\boldsymbol{x}_0)) + \mathrm{Var}(y_D(\boldsymbol{x}_0)).$$

The derivation above depends on a particular point $\boldsymbol{x}_0$, so let us put it all back in place within their integrals:

$$\mathrm{Bias}^2(y_D) = \int_{\mathbb{R}^d} \mathrm{Bias}^2(y_D(\boldsymbol{x}))p(\boldsymbol{x})\, d\boldsymbol{x},$$
$$\mathrm{Var}(y_D) = \int_{\mathbb{R}^d} \mathrm{Var}(y_D(\boldsymbol{x}))p(\boldsymbol{x})\, d\boldsymbol{x},$$
$$\boxed{R(y_D) = \sigma^2 + \mathrm{Bias}^2(y_D) + \mathrm{Var}(y_D).}$$

In general, an **underfit** model will have a big bias, while an **overfit** model will have a high variance. The *abilityi to fit* has a name: it's called the **complexity** of the function class. Both models that are more or less complex than needed will tend to have large prediction errors. In the former, this will be dominated by the variance term, while in the latter, it will be dominated by the (square) bias term.

4. Regression theory and linear regression models (II)

Our departing statistical model still is

$$t_n = f(\boldsymbol{x}_n) + \varepsilon_n,\ \boldsymbol{x}_n \in \mathbb{R}^d,\ t \in \mathbb{R}$$

where ε_n is a continuous rv such that $\mathbb{E}[\varepsilon_n] = 0$ and $\mathrm{Var}(\varepsilon_n) = \sigma^2$. Let's assume again that we further model $\varepsilon_n \sim \mathcal{N}\left(0, \sigma^2\right)$, and:

$$f(\boldsymbol{x}) \approx y(\boldsymbol{x}; \boldsymbol{\beta}) = \sum_{i=0}^{d} \beta_i x_i = \boldsymbol{\beta}^T \boldsymbol{x}$$

with $\boldsymbol{x} = (1, x_1, \ldots, x_d)^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)^T$. Suppose we have an iid sample of N labeled observations $D = \{(\boldsymbol{x}_n, t_n)\}_{n=1,\ldots,N}$, where $\boldsymbol{x}_n \in \mathbb{R}^d, t_n \in \mathbb{R}$. Therefore, our statistical model is $t_n \sim \mathcal{N}\left(y(\boldsymbol{x}_n; \boldsymbol{\beta}), \sigma^2\right)$ or:

$$p(t_n | \boldsymbol{x}_n; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \left(t_n - \boldsymbol{\beta}^T \boldsymbol{x}_n\right)^2 \right),$$

with unknown parameters $\theta := \{\beta_0, \beta_1, \ldots, \beta_p, \sigma^2\}$. Put $\boldsymbol{t} = (t_1, \ldots, t_N)^T$ and $X_{N\times(d+1)}$ the matrix of the $\boldsymbol{x}_n$. Define the **likelihood** as $\mathcal{L}(\theta) := P(\boldsymbol{t} | X; \theta)$. Let us maximize the log-likelihood:

$$\begin{aligned}
l(\theta) := \log \mathcal{L}(\theta) = \log \prod_{n=1}^{N} p(t_n | \boldsymbol{x}_n; \theta) = \sum_{n=1}^{N} \log p(t_n | \boldsymbol{x}_n; \theta) = \\
= -\frac{N}{2} \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(t_n - \boldsymbol{\beta}^T \boldsymbol{x}_n\right)^2 = \\
= -\frac{N}{2} \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} (\boldsymbol{t} - X\boldsymbol{\beta})^T (\boldsymbol{t} - X\boldsymbol{\beta}) = \\
= -\frac{N}{2} \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} \|\boldsymbol{t} - X\boldsymbol{\beta}\|^2.
\end{aligned}$$

If we derive this wrt $\boldsymbol{\beta}$ and σ^2, and set equal to zero, we get:

$$\begin{aligned}
\frac{\partial l}{\partial \boldsymbol{\beta}} &= -\frac{1}{2\sigma^2} \left(-2X^T \boldsymbol{t} + 2X^T X \boldsymbol{\beta}\right) = 0 \\
\frac{\partial l}{\partial \sigma^2} &= -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4} (\boldsymbol{t} - X\boldsymbol{\beta})^T (\boldsymbol{t} - X\boldsymbol{\beta}) = 0.
\end{aligned}$$

Therefore, we can calculate the estimates for both parameters:

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \left(X^T X\right)^{-1} X^T \boldsymbol{t}, \\
\hat{\sigma}^2 &= \frac{1}{N} (\boldsymbol{t} - X\hat{\boldsymbol{\beta}})^T (\boldsymbol{t} - X\hat{\boldsymbol{\beta}}) = \frac{1}{N} \|\boldsymbol{t} - X\hat{\boldsymbol{\beta}}\|^2.
\end{aligned}$$

Note that $\hat{\sigma}^2 = R_{\mathrm{emp}}(y_D)$, which is a **biased estimator** for σ^2. An unbiased estimator is

$$\bar{\sigma}^2 = \frac{N}{N-d} \hat{\sigma}^2.$$

It's also known that $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of $\boldsymbol{\beta}$ and that $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \left(X^T X\right)^{-1} \sigma^2$. All of this implies that $\hat{\boldsymbol{\beta}} \sim \mathcal{N}\left(\boldsymbol{\beta}, \left(X^T X\right)^{-1} \sigma^2\right)$.

The matrix $X^+ = \left(X^T X\right)^{-1} X^T$ is known as the **Moore-Penrose pseudo-inverse** of X. It is the generalization of the notion of an inverse matrix to non-square matrices. It has the property that $X^+ X = I$, although in general $XX^+ \neq I$. However, both are symmetric.

Theorem. Let $X_{N\times M}$, with $N > M$. If the column vectors of X are linearly independent, *i.e.*, if $\mathrm{rank}(X) = M$, then:

1. The matrix $X^T X$ is symmetric and positive definite. In particular, it is non-singular.
2. The least squares problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^M} \|\boldsymbol{t} - X\boldsymbol{\beta}\|^2,$$

   has a unique solution.

3. This solution can be found solving the so-called Gauss' normal equations,

$$\left(X^T X\right)\boldsymbol{\beta} = X^T \boldsymbol{t}$$

   for $\boldsymbol{\beta}$.

Quality of the fit

- In statistics, $-2l = -2\log\mathcal{L}$ is called the **deviance**.
- In ML, this quality measure is the **square error**:

$$N\log\left(2\pi\sigma^2\right) + \frac{1}{\sigma^2}\|\boldsymbol{t} - X\hat{\boldsymbol{\beta}}\|^2$$

- A much better quantity to report is the NRMSE,

$$\mathrm{NRMSE}(\hat{\boldsymbol{\beta}}) = \sqrt{\frac{\|\boldsymbol{t} - X\hat{\boldsymbol{\beta}}\|^2}{(N-1)\mathrm{Var}(\boldsymbol{t})}}.$$

In statistics, $R^2 = 1 - \mathrm{NRMSE}^2$ is the proportion of the target variability *explained* by the model.

Leaping forward: basis functions

Recall that a model is **linear** if up to an invertible function its parameters play a linear role in the model. For example,

$$y(x;\boldsymbol{\beta}) = \sum_{j=0}^{d} \beta_j x^j,\ x\in\mathbb{R}$$

is a polynomial on x, but also a linear model on $\boldsymbol{\beta}$.

A simple but powerful idea is the introduction of **basis functions**:

$$y(\boldsymbol{x};\boldsymbol{w}) = \sum_{j=0}^{M} w_j\varphi_j(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}),$$

where $\varphi_0(\boldsymbol{x}) = 1, \boldsymbol{\varphi}(\boldsymbol{x}) = (1, \varphi_1(\boldsymbol{x}), \ldots, \varphi_M(\boldsymbol{x}))^T, \boldsymbol{w} = (w_0, w_1, \ldots, w_M)^T$. This is still a **linear model**. Define $\boldsymbol{t} = (t_1, \ldots, t_N)^T$ as the vector of targets, and $\boldsymbol{\varphi}_{N\times(M+1)}$ as the matrix of the $\boldsymbol{\varphi}_{ij} = \varphi_j(\boldsymbol{x}_i), i = 1,\ldots,N, j = 1,\ldots,M$:

$$\boldsymbol{\varphi} = \begin{pmatrix} 1 & \varphi_1(\boldsymbol{x}_1) & \varphi_2(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ 1 & \varphi_1(\boldsymbol{x}_2) & \varphi_2(\boldsymbol{x}_2) & \cdots & \varphi_M(\boldsymbol{x}_2) \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ 1 & \varphi_1(\boldsymbol{x}_N) & \varphi_2(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}.$$

So, let us maximize the new log-likelihood: the Gauss' normal equations are

$$\left(\boldsymbol{\varphi}^T \boldsymbol{\varphi}\right) w = \boldsymbol{\varphi}^T \boldsymbol{t}$$

and their solution is

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{\varphi}^T \boldsymbol{\varphi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{t} = \boldsymbol{\varphi}^+ \boldsymbol{t},$$
$$\hat{\sigma}^2 = \frac{1}{N}(\boldsymbol{t} - \boldsymbol{\varphi}\hat{\boldsymbol{w}})^T(\boldsymbol{t} - \boldsymbol{\varphi}\hat{\boldsymbol{w}}) = \frac{1}{N}\|\boldsymbol{t} - \boldsymbol{\varphi}\hat{\boldsymbol{w}}\|^2.$$

Singular Value Decomposition

The direct computation of the pseudo-inverse of $\boldsymbol{\varphi}$ has two major drawbacks:

- When M is large, $\boldsymbol{\varphi}^T\boldsymbol{\varphi}$ is a large $(M+1)\times(M+1)$ matrix; then, the computation of the required inverse $\left(\boldsymbol{\varphi}^T\boldsymbol{\varphi}\right)^{-1}$ can be costly.
- If $\boldsymbol{\varphi}^T\boldsymbol{\varphi}$ is singular, or close to, then the required inverse $\left(\boldsymbol{\varphi}^T\boldsymbol{\varphi}\right)^{-1}$ can be impossible, or numerically delicate.

Theorem. Every matrix $X_{N\times M}$ can be expressed as $X = U\Delta V^T$, with $U \in \mathcal{M}_N(\mathbb{R})$,$V \in \mathcal{M}_M(\mathbb{R})$,$\Delta \in \mathcal{M}_{N\times M}(\mathbb{R})$ diagonal. The columns of U are the eigenvectors of XX^T, and the columns of V are the eigenvectors of X^TX.

Let $\operatorname{rank}(X) = r \leq \min(N, M)$. Then exactly r elements λ_k in the diagonal of Δ are strictly positive; the remaining elements are null. These $\lambda_k > 0$ are called the **singular values** and correspond to the square roots of the positive eigenvalues of XX^T (same as X^TX).

Sometimes an *economy* size decomposition is delivered: If X is $N \times M$ with $N > M$, then only the first M columns of U are given and Δ is $M \times M$.

SVD for least squares

Given the least squares problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^M} \|\boldsymbol{t} - X\boldsymbol{w}\|^2,$$

the solution can be obtained with the SVD as:

- Compute the economy size SVD of $X = U\Delta V^T$.
- Solve for $\boldsymbol{w}$ as $\hat{\boldsymbol{w}} = V\operatorname{diag}\left(\lambda_k^{-1}\right)U^T\boldsymbol{t}$, where only the $\lambda_k > 0$ are considered.

Regularized least squares

The maximum likelihood framework can yield unstable parameter estimates, specially when

- the explanatory variables are highly correlated;
- there is an insufficient number of observations (N) relative to the number of predictors (basis functions $M+1$ or dimensions $d+1$).

In the context of regression with Gaussian noise (square error), it is quite common to penalize the parameter vector. Define the **penalized empirical error** as:

$$R_{\text{emp}}(y(\cdot;\boldsymbol{w})) := \|\boldsymbol{t} - \boldsymbol{\varphi}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2,\ \lambda > 0.$$

If we set its derivative wrt $\boldsymbol{w}$ equal to zero

$$-2\boldsymbol{\varphi}^T\boldsymbol{t} + 2\boldsymbol{\varphi}^T\boldsymbol{\varphi}\boldsymbol{w} + 2\lambda\boldsymbol{w} = 0,$$

we solve for $\boldsymbol{w}$ and we get

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{\varphi}^T\boldsymbol{\varphi} + \lambda I)^{-1}\boldsymbol{\varphi}^T\boldsymbol{t}.}$$

This is known as **Tikhonov** or L^2 **regularization** in ML. Perhaps it's best known as **ridge regression** in statistics, where it's usually explained as a "penalized log-likelihood". This can also be derived from Bayesian statistics arguments. Tikhonov regularization has some advantages:

- Pushing the length of the parameter vector $\|\boldsymbol{w}\|$ to 0 allows the fit to be under explicit control with the regularization parameter λ.
- The matrix $\boldsymbol{\varphi}^t\boldsymbol{\varphi}$ is positive semi-definite; therefore $\boldsymbol{\varphi}^T\boldsymbol{\varphi} + \lambda I$ is guaranteed to be positive definite (hence non-singular), for all $\lambda > 0$.

Bayesian derivation:

$$\left.\begin{array}{c} D \text{ data} \\ \boldsymbol{\theta} \text{ parameter vector} \end{array}\right\} \underset{\theta\in\Theta}{\operatorname{argmax}} P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \sim \underset{\theta\in\Theta}{\operatorname{argmax}} P(D|\theta)P(\theta) \sim$$

$$\sim \underset{\theta\in\Theta}{\operatorname{argmax}} \ln(P(D|\theta)P(\theta)) \sim \underset{\theta\in\Theta}{\operatorname{argmax}}\{\ln P(D|\theta) + \ln P(\theta)\} \sim$$

$$\sim \underset{\theta\in\Theta}{\operatorname{argmin}}\{-\ln P(D|\theta) - \ln P(\theta)\}$$

We change names of $\theta = w$, for it is fancier. If we assume that the $w \sim \mathcal{N}\left(0, \sigma^2\mathrm{Id}\right)$, then $-\ln P(w) = \frac{\|w\|^2}{2\sigma_w^2}$ and then, observe that $-\ln\mathcal{L}$ is the mean square error,

$$\ldots \underset{w\in W}{\operatorname{argmin}}\{-\ln\mathcal{L} - \ln P(w)\} = \underset{w\in W}{\operatorname{argmin}}\left\{\|t - \varphi w\|^2 + \frac{\|w\|^2}{2\sigma_w^2} + C\left(d, \sigma_w^2\right)\right\} \sim$$

$$\sim \underset{w\in W}{\operatorname{argmin}}\left\{\|t - \varphi w\|^2 + \lambda\|w\|^2\right\},\ \lambda > 0.$$

This is all nice, but how do we control the fit explicitly?

- Regularization allows the specification of models that are more complex than needed because it limits the effective complexity.
- Instead of trial-and-error on complexity, we can set a large complexitr and adjust the λ.

And how do we set the value of λ? Using a technique called **Leaving-one-out cross validation (LOOCV)**, because

- In this case, λ is a very forgiving parameter; we usually perform a log search.
- There is a closed efficient formula for the LOOCV for **linear models**.

To get to the best model we can, we follow this steps:

1. Choose a (large) set of values Λ.
2. For every $\lambda \in \Lambda$,
    1. Solve for $\hat{w} = \left(\boldsymbol{\varphi}^T\boldsymbol{\varphi} + \lambda I\right)^{-1}\boldsymbol{\varphi}^T\boldsymbol{t}$.
    2. Compute the **hat matrix** $H := \boldsymbol{\varphi}\boldsymbol{\varphi}^+ \equiv \boldsymbol{\varphi}\left(\boldsymbol{\varphi}^T\boldsymbol{\varphi} + \lambda I\right)^{-1}\boldsymbol{\varphi}^T$.
    3. Compute the LOOCV of $y(\cdot) = \hat{\boldsymbol{w}}^T\boldsymbol{\varphi}(\cdot)$ in D as

$$\texttt{LOOCV}(y) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{t_n - \hat{\boldsymbol{w}}^T\boldsymbol{\varphi}(x_n)}{1 - h_{nn}}\right)^2.$$

3. Choose the model with the lowest LOOCV.

A very popular method is **Generalized Cross-Validation (GCV)**:

$$\texttt{GCV}(y) = \frac{1}{N} \frac{\sum_{n=1}^{N} \left(t_n - \hat{\boldsymbol{w}}^T \boldsymbol{\varphi}(x_n)\right)^2}{\left(1 - \frac{\text{tr}(H)}{N}\right)^2},$$

which is a more stable computation for the LOOCV. Note that λ is needed to compute both $\hat{\boldsymbol{w}}$ and H.

LASSO Regression

The LASSO (**Least Absolute Shrinkage and Selection Operator**) regression is L^1-regularized linear regression. The choice for the regularizer is $\|\boldsymbol{w}\|_1$ and we get:

$$R_{\text{emp}}(y(\cdot;\boldsymbol{w})) = \|\boldsymbol{t} - \boldsymbol{\varphi}\boldsymbol{w}\|^2 + \tau\|\boldsymbol{w}\|_1, \quad \tau > 0.$$

This turns out to be equivalent to

$$R_{\text{emp}}(y(\cdot;\boldsymbol{w})) = \|\boldsymbol{t} - \boldsymbol{\varphi}\boldsymbol{w}\|^2, \quad \text{subject to } \|\boldsymbol{w}\|_1 \leq \tau.$$

In ridge regression, as the penalty λ is increased, all coefficients are reduced while still remaining non-zero. In the LASSO regression, increasing the τ penalty causes more and more of the coefficients to be driven to zero. As the dimension d increases, the multidimensional L^1-spheres have an increasing number of corners, and so it is highly likely that some coefficients will be set equal to zero. Hence, the LASSO regression model performs **shrinkage** and therefore, **feature selection**.

The LASSO loss function is no longer quadratic, but it is still convex. The **minimization problem** tied to LASSO regression is a special **quadratic programming (QP)** problem, for which the **Least Angle Regression (LARS)** procedure is used. It exploits the special structure of the problem, and provides an efficient way to compute the solutions for all possible values of $\tau > 0$ (the **regularization path**).

Conclusions

We have introduced **linear models** as linear combinations of non-linear **basis functions (BF)**:

ADVANTAGES:

- We can represent non-linear functions of the data using linear fitting techniques; we have the freedom to choose the form of the BFs.
- The fit can be under tight explicit control by regularization.
- The computations can be very efficient, no need to refit for LOOCV.
- Interpretability of the model is rather high.

LIMITATIONS: the most important weak point is the BFs.

- Many interesting BFs scale very poorly with dimension (polynomials, Fourier series, splines, ...)
- Our BFs are not flexible; they are data-independent.
- As a consequence, their number may be very high, which in turn leads to unstability (because of low significance of the coefficients).

The solution is to **develop basis functions with parameters** such that:

- This BFs scale well with dimension (inner products, distances, ...)
- They are **data-dependent**, because of the parameters.
- As a consequence, their number might be much lower, and the coefficients will be significant.
- Unfortunately, the new parameters will play a **non-linear role** in the model: their optimization is plagued with local optima.

5. Classification theory and linear classification models (I). Bayesian decision theory.

Introduction: Bayes' formula

Discrete Random Variables. Let A be a discrete r.v. with probability mass function (*pmf*) P_A. We use the shorthand notation $P(a)$ to mean $P_A(A=a)$. Similarly, we write $P(b|a)$ to mean $P_{B|A}(B=b|A=a)$, etc, where

$$P(b|a) = \frac{P(b,a)}{P(a)},\ P(a) > 0.$$

Let $\mathcal{A} = \{a_1, \ldots, a_n\}, \mathcal{B} = \{b_1, \ldots, b_m\}$ be the possible values that A and B can take, respectively. Then, $\forall a \in \mathcal{A}$,

$$P(a) = \sum_{i=1}^{m} P(a, b_i) = \sum_{i=1}^{m} P(a|b_i)P(b_i).$$

Since $P(a,b) = P(b,a)$, it follows that, for any $a \in \mathcal{A}, b \in \mathcal{B}$

$$P(b|a) = \frac{P(a|b)P(b)}{\sum_{b' \in \mathcal{B}} P(a|b')P(b')}, \quad \text{with } \sum_{b' \in \mathcal{B}} P(b|a) = 1.$$

Continuous Random Variables. Let X, Y two continuous r.v. with pdfs p_X, p_Y and joint density p_{XY}. We use the shorthand notation $p(x)$ to mean $p_X(X=x)$, etc.

$$p(x) = \int_{\mathbb{R}} p(x,y)\ dy; \quad p(y) = \int_{\mathbb{R}} p(x,y)\ dx.$$

Therefore,

$$p(y|x) = \frac{p(x|y)p(y)}{\int_{\mathbb{R}} p(x|y)p(y)\ dy}, \quad \text{with } \int_{\mathbb{R}} p(y|x)\ dy = 1.$$

Observation. *Mixed random variables.*

Suppose X is a continuous r.v. and Y is a discrete r.v. with values in $\{y_1, \ldots, y_m\}$. In this case, $p(\cdot|y_i)$ is a continuous r.v. and $P(\cdot|x)$ is a discrete r.v. Moreover,

$$P(y_j|x) = \frac{p(x|y_j)P(y_j)}{\sum_{i=1}^{m} p(x|y_j)P(y_j)}, \qquad \text{with } \sum_{j=1}^{m} P(y_j|x) = 1.$$

Decision rules

We are interested in determining the class or category of objects of nature according to Ω, a discrete r.v. with values $\{\omega_1, \omega_2\}$ that represent the two possible classes. The prior probabilities are $P(\omega_1), P(\omega_2)$. How should we classify objects?

Decision Rule 1. We don't measure any variable. We have no information other than "a new object comes".

`If` $P(\omega_1) > P(\omega_2)$ `then class of object is` ω_1 `else class is` ω_2`.`

This rule classifies all objects into the same class; therefore, it will eventually classify an object into the wrong class. Thus, the **probability of error** of this rule is

$$P_e(\texttt{rule1}) = \min\{P(\omega_1), P(\omega_2)\}.$$

This rule is useful only if $P(\omega_1) << P(\omega_2)$ or if $P(\omega_2) << P(\omega_1)$. This is the optimum rule when no information is measured.

Discrete feature measuring. Suppose now that X is a discrete r.v. taking values in $\mathcal{X} = \{x_1, \ldots, x_d\}$ that measures a **feature** of objects. Now, $P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)}$ is the **posterior** probability that an object with measured feature x belongs to class $\omega_i,\ i \in \{1, 2\}$. Moreover, $P(x) = P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)$.

Upon observing x, the Bayes formula converts **prior** class probabilities $P(\omega_i)$ into **posterior** probabilities $P(\omega_i|x)$. How should we classify objects now?

Decision Rule 2. We now measure a feature x of the object coming forth.

$$\boxed{\texttt{If } P(\omega_1|x) > P(\omega_2|x) \texttt{ then class of object is } \omega_1 \texttt{ else class is } \omega_2.}$$

The probability of error for this rule is

$$P_e(\texttt{rule2}) = \sum_{i=1}^{d} \min\{P(\omega_1|x_i), P(\omega_2|x_i)\}P(x_i).$$

This rule is known as the **Bayes rule** or **classifier**.

Lemma. For all $a, b, c, d \in \mathbb{R}$, $\min(a, b) + \min(c, d) \leq \min(a + c, b + d)$.

Proposition. $P_e(\texttt{rule2}) \leq P_e(\texttt{rule1})$.

Proof:

$$\begin{aligned}
\sum_{i=1}^{d} \min\{P(\omega_1|x_i), P(\omega_2|x_i)\}P(x_i) = \\
= \sum_{x \in \mathcal{X}} \min\{P(\omega_1|x)P(x), P(\omega_2|x)P(x)\} = \\
= \sum_{x \in \mathcal{X}} \min\{P(x|\omega_1)P(\omega_1), P(x|\omega_2)P(\omega_2)\} \leq \\
\leq \min\left\{\sum_{x \in \mathcal{X}} P(x|\omega_1)P(\omega_1), \sum_{x \in \mathcal{X}} P(x|\omega_2)P(\omega_2)\right\} = \\
= \min\left\{P(\omega_1)\sum_{x \in \mathcal{X}} P(x|\omega_1), P(\omega_2)\sum_{x \in \mathcal{X}} P(x|\omega_2)\right\} = \\
= \min\{P(\omega_1), P(\omega_2)\}\ \square
\end{aligned}$$

Equality holds only if $P(x|\omega_1) = P(x|\omega_2),\ \forall x \in \mathcal{X}$.

Continuous feature measuring. The next step is to consider a r.v. X with pdf $p(x)$ that measures a *continuous* feature of an object. Let $\mathcal{P}$ be the support of p, i.e. $\mathcal{P} = \{x \in \mathbb{R} | p(x) > 0\}$. In this setting, $p(x|\omega_i),\ i \in \{1, 2\}$ are the conditional densities of x for every class.

Proposition. $P_e(\texttt{rule2}) \leq P_e(\texttt{rule1})$.

Proof:

$$\begin{aligned}
\int_{\mathcal{P}} \min\{P(\omega_1|x), P(\omega_2|x)\}p(x)\ dx = \\
\int_{\mathcal{P}} \min\{P(\omega_1|x)p(x), P(\omega_2|x)p(x)\}\ dx = \\
\int_{\mathcal{P}} \min\{p(x|\omega_1)P(\omega_1), p(x|\omega_2)P(\omega_2)\}\ dx \leq \\
\min\left\{\int_{\mathcal{P}} p(x|\omega_1)P(\omega_1)\ dx, \int_{\mathcal{P}} p(x|\omega_2)P(\omega_2)\ dx\right\} = \\
\min\left\{P(\omega_1)\int_{\mathcal{P}} p(x|\omega_1)\ dx, P(\omega_2)\int_{\mathcal{P}} p(x|\omega_2)\ dx\right\} = \\
\min\{P(\omega_1), P(\omega_2)\}\ \square
\end{aligned}$$

Equality holds only if $p(\cdot|\omega_1) = p(\cdot|\omega_2)$.

The Bayes classifier

The Bayes classifier can be extended in two ways:

1. Consider a vector $X = (X_1, \ldots, X_d)^T$ of continuous r.v. with pdf $p(\boldsymbol{x}) = p(x_1, \ldots, x_d)$ that measures d continuous features.
2. Consider a finite set of classes Ω, a discrete r.v. with values $\omega_1, \ldots, \omega_K$, that represent the possible classes ($K \geq 2$).

Therefore, we have new probabilities $p(\boldsymbol{x}|\omega_i), P(\omega_i|\boldsymbol{x}), 1 \leq i \leq K$. The new Bayes rule says:

Decision rule.

$$\boxed{\texttt{The class } \hat{\omega}(\boldsymbol{x}) \texttt{ of object } \boldsymbol{x} \texttt{ is } \omega_k \texttt{ when } k = \underset{i=1,\ldots,K}{\operatorname{argmax}} P(\omega_i|\boldsymbol{x}).}$$

The sets $\mathcal{R}_k = \{\boldsymbol{x} | \hat{\omega}(\boldsymbol{x}) = k\}$ are called **regions**, and depend on the specific classifier. It is worth noting they form a partition of the total space, which is in general thought of as $\mathbb{R}$ or, in the vector setting, $\mathbb{R}^d$.

We now want to see that the Bayes classifier is **optimal** in terms of probability of error. To do this, let us assume a classifier with regions $\mathcal{R}_1, \mathcal{R}_2$. Then,

$$\begin{aligned}
P_e = P(\boldsymbol{x} \in \mathcal{R}_2, \omega_1) + P(\boldsymbol{x} \in \mathcal{R}_1, \omega_2) = \\
P(\boldsymbol{x} \in \mathcal{R}_2|\omega_1)P(\omega_1) + P(\boldsymbol{x} \in \mathcal{R}_1|\omega_2)P(\omega_2) = \\
\int_{\mathcal{R}_2} p(\boldsymbol{x}|\omega_1)P(\omega_1)\ d\boldsymbol{x} + \int_{\mathcal{R}_1} p(\boldsymbol{x}|\omega_2)P(\omega_2)\ d\boldsymbol{x} \geq \\
\int_{\mathcal{P}} \min\left\{p(\boldsymbol{x}|\omega_1)P(\omega_1), p(\boldsymbol{x}|\omega_2)P(\omega_2)\right\}\ d\boldsymbol{x} = \\
P_e(\texttt{Bayes})\square
\end{aligned}$$

So, if any other classifier has a smaller error, the Bayes classifier is optimal.

The Bayes classifier can also have a **rejection class** (illustrated here for two classes); if we fix $\varepsilon \in (0,1)$,

$$\begin{aligned}&\texttt{if } P(\omega_1|\boldsymbol{x}) - P(\omega_2|\boldsymbol{x}) > \varepsilon \texttt{ then class of object } \boldsymbol{x} \texttt{ is } \omega_1\\ &\texttt{else if } P(\omega_1|\boldsymbol{x}) - P(\omega_2|\boldsymbol{x}) < \varepsilon \texttt{ then class of object } \boldsymbol{x} \texttt{ is } \omega_2\\ &\texttt{else do not classify.}\end{aligned}$$

For every feature vector $\boldsymbol{x}$ we take one of three possible **actions**.

Consider a finite set of actions $A = \{a_1, \ldots, a_m\}$. For each $a_i \in A$, denote by $l(a_i|\omega_j)$ the **loss** for choosing a_i when $\boldsymbol{x}$ is known to be in ω_j. This is a simplified setting in which this loss does not depend on $\boldsymbol{x}$.

Example: Let $m = K + 1$ and let a_i stand for *"classify $\boldsymbol{x}$ into class ω_i"* for $1 \le i \le K$; let a_{K+1} stand for *"do not classify $\boldsymbol{x}$"*. A possible set of losses is:

$$l(a_i|\omega_j) = \begin{cases} 1, & \text{for } 1 \le i, j \le K,\ i \ne j; \\ 0, & \text{for } 1 \le i \le K,\ i = j; \\ \frac{1}{2}, & \text{for } i = K+1,\ 1 \le j \le K. \end{cases}$$

This example suggests that a decision not to classify is less costly than a misclassification.

The notion of risk

Definition. *Conditional risk.* For a given feature vector $\boldsymbol{x}$, define the conditional risk of an action as:

$$r(a_i|\boldsymbol{x}) := \sum_{j=1}^{K} l(a_i|\omega_j) P(\omega_j|\boldsymbol{x}).$$

Definition. *Decision rule, Total risk.* A decision rule is any function $a : \mathcal{P} \to A$ from the support of the probability density function $\mathcal{P}$ to the action set A that assigns an action $a(\boldsymbol{x})$ to every $\boldsymbol{x}$ such that $p(\boldsymbol{x}) > 0$. The total risk of a decision rule is

$$R(a) := \mathbb{E}_{\boldsymbol{X}}[r(a(\boldsymbol{x})|\boldsymbol{x})] = \int_{\mathcal{P}} r(a(\boldsymbol{x})|\boldsymbol{x}) p(\boldsymbol{x})\, d\boldsymbol{x}.$$

We are interested in the decision rule that minimizes the total risk. Consider the rule

$$\hat{a}(\boldsymbol{x}) = \operatorname*{argmin}_{1 \le j \le m} r(a_j|\boldsymbol{x}).$$

You may recognize it as the Bayes rule. Given that this rule minimizes the argument of the integral for every possible $\boldsymbol{x}$, it follows that the Bayes rule has the lowest possible risk. The value $R(\hat{a})$ is called the **Bayes risk** and is the minimum risk possible in global terms.

0/1 losses

In many applications the 0/1 loss is used, usually in absence of more precise information:

$$l_{ij} = \begin{cases} 0, & \text{if } i = j; \\ 1, & \text{if } i \ne j. \end{cases}$$

Consider K classes, and actions a_i : $\texttt{classify } \boldsymbol{x} \texttt{ into } \omega_i$. Then, the conditional risk for each action is

$$r\left(a_i|\boldsymbol{x}\right) = \sum_{j=1}^{K} l_{ij} P\left(\omega_j|\boldsymbol{x}\right) = \sum_{j=1, i \ne j}^{K} P\left(\omega_j|\boldsymbol{x}\right) = 1 - P\left(\omega_i|\boldsymbol{x}\right).$$

Discriminant functions

Functions of the form $g_k : \mathcal{P} \to \mathbb{R}$ are a useful tool to build an abstract classifier. An object $\boldsymbol{x}$ is assigned to class ω_i when $g_i(\boldsymbol{x})$ is the highest among the values $g_1(\boldsymbol{x}), \dots, g_K(\boldsymbol{x})$. For example,

- $g_k(\boldsymbol{x}) = P(\omega_k|\boldsymbol{x})$.
- $g_k(\boldsymbol{x}) = P(\omega_k)p(\boldsymbol{x}|\omega_k)$.
- $g_k(\boldsymbol{x}) = -r(a_k|\boldsymbol{x})$.

If g_k is a discriminant function, then so is $h \circ g_k$, for any strictly monotonic function h. For two classes, we can use a single discriminant function, called a **dichotomizer**:

1. Define $g(\boldsymbol{x}) := g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$.
2. Assign $\boldsymbol{x}$ to class ω_1 if $g(\boldsymbol{x}) > 0$ and to class ω_2 if $g(\boldsymbol{x}) < 0$.

The Gaussian Distribution

A normally distributed $d-$variate random vector $X = (X_1, \dots, X_d)^T$ has a pdf like

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

where $\boldsymbol{\mu}$ is the **mean vector** and $\Sigma_{d\times d} = \left(\sigma_{ij}^2\right)$ is the real symmetric and positive definite **covariance matrix**.

- $\mathbb{E}[X] = \boldsymbol{\mu}$ and $\mathbb{E}\left[(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^T\right] = \Sigma$.
- $\mathrm{Cov}[X_i, X_j] = \sigma_{ij}^2$ and $\mathrm{Var}[X_i] = \sigma_{ii}^2$.

As $X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then X_i, X_j are statistically independent $\iff \mathrm{Cov}[X_i, X_j] = 0$. In general, only $\implies$ holds.

The quantity $d(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$ is called the **Mahalanobis distance**. What is behind the choice of a multivariate Gaussian distribution for a class?

- We want to have a **prototype object**, which is modeled by the mean vector.
- Also, the **noise modeling** is easy because in a multivariate Gaussian, this is modeled by the covariance matrix.
- Even though, it is very important to take into account that the number of parameters of the multivariate Gaussian is $\frac{d(d+1)}{2} + d$, for dimension d.

The surfaces of equal probability, $d(\boldsymbol{x}, \boldsymbol{\mu}) = \mathrm{const}$ are **hyperellipsoids**. The principal directions or components (PC) of the hyperellipsoids are given by the eigenvectors u_i of Σ, which satisfy $\Sigma u_i = \lambda_i u_i$, for $\lambda_i > 0$. The lengths of the hyperellipsoids along these axes are proportional to $\sqrt{\lambda_i}$, the **singular values** associated with u_i. Note that as Σ is positive definite, all $\lambda_i > 0$.

Properties

- **Simplified forms:** if the X_i are statistically independent, then $p(\boldsymbol{x}) = \prod_{i=1}^d p(x_i)$, Σ is diagonal and the PCs are axis-aligned.
- **Analytical properties**, for example, any moment $\mathbb{E}\left[X^p\right]$ can be expressed as a function of $\boldsymbol{\mu}$ and Σ.
- **Central limit theorem**, the mean of d i.i.d. random variables tends to a normal distribution, as $d \to \infty$.
- **Linear transformation invariance**, the distribution of a linear transformation of the coordinate system remains normal.

6. Classification theory and linear classification models (II).

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

We showed that the Bayes rule minimizing the probability of error could be formulated in terms of a family of **discriminant functions**:

`assign the feature vector` $\boldsymbol{x}$ `to class` ω_k `whenever` $g_k(\boldsymbol{x})$ `is the largest,` $1 \leq k \leq K$.

When $X_{|\Omega=\omega_k} \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$, this family can be reduced to very simple expressions. Using Bayes rule and the natural logarithm, the discriminant function for class ω_k becomes

$$g_k(\boldsymbol{x}) = \ln[P(\omega_k)\, p(\boldsymbol{x}|\omega_k)] = \ln P(\omega_k) - \ln\left[(2\pi)^{\frac{d}{2}} \|\Sigma_k\|^{\frac{1}{2}}\right] - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k).$$

Erasing constant terms,

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) - \frac{1}{2}\left(\ln \|\Sigma_k\| - (\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right).$$

This expression is called a **quadratic discriminant function**, and the **decision boundaries** $g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$ are general hyper-quadrics in $d-$dimensional space. The resulting classifier out of the class of discriminant functions together with the decision rule is called **Quadratic Discriminant Analysis (QDA)**.

- If we assume that all class-conditional distributions $p(\boldsymbol{x}|\omega_k)$ have the **same covariance matrix** Σ, after removing all terms that do not depend on k or $\boldsymbol{x}$ we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) + \boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k.$$

Reorganizing terms we obtain $g_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$, where

$$\begin{aligned} \boldsymbol{w}_k &= \Sigma^{-1}\boldsymbol{\mu}_k, \\ w_{k0} &= -\tfrac{1}{2}\boldsymbol{\mu}_k^T \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\omega_k). \end{aligned}$$

This is because Σ^{-1} is a symmetric matrix for being the inverse of a symmetric matrix. These are **linear discriminant functions** (linear in $\boldsymbol{x}$) and the **decision boundaries** are hyperplanes if $d-$dimensional space.

- If we further assume that all X_i, X_j pairs are **statistically independent**, that is, Σ is a diagonal matrix, we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) - \frac{1}{2}\sum_{i=1}^{d} \frac{(\mu_{ki} - x_i)^2}{\sigma_i^2}.$$

- If we further assume that all X_i have the **same variance** σ^2 (for example, standardizing all variables), that is $\Sigma = \sigma^2 I_d$, we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) - \frac{1}{2\sigma^2}\|\boldsymbol{\mu}_k - \boldsymbol{x}\|^2.$$

- If we further assume that all classes have the **same prior distribution**, $P(\omega_k) = \frac{1}{K}$, we get:

$$g_k(\boldsymbol{x}) = -\|\boldsymbol{\mu}_k - \boldsymbol{x}\|^2.$$

In all cases, we have a **minimum-distance** classifier in $\mathbb{R}^d$:

- In the general case (some covariance matrices are different), the classifier uses a different Mahalanobis distance (a fully-weighted Euclidean distance) from $\boldsymbol{x}$ to each class. This is called **Quadratic Discriminant Analysis (QDA)**.
- In case all covariance matrices are equal, the classifier uses the same Mahalanobis distance from $\boldsymbol{x}$ to all classes. This is called **Linear Discriminant Analysis (LDA)**.
- In case all covariance matrices are diagonal, the classifier uses a simply-weighted Euclidean distance from $\boldsymbol{x}$ to all classes.
- In case all covarianve matrices are a multiple of the identity matrix, the classifier uses an unweighted Euclidean distance from $\boldsymbol{x}$ to all classes.

A numerical example

Derive a linear discriminant function for the two-class classification problem defined by the following Gaussian class-conditional densities:

$$\boldsymbol{\mu}_1 = (0,0,0)^T, \boldsymbol{\mu}_2 = (1,1,1)^T, \Sigma_1 = \Sigma_2 = \operatorname{diag}\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), P(\omega_2) = 2P(\omega_1).$$

Solution: since all the X_i, X_j are statistically independent ($i \neq j$) and have the same variance $\sigma^2 = \frac{1}{4}$, that is $\Sigma = \frac{1}{4}I$, we get:

$$g_1(\boldsymbol{x}) = \ln P(\omega_1) - \frac{1}{2}\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{x}\|^2}{\sigma^2} = \ln\frac{1}{3} - \frac{1}{2}\frac{\|(0,0,0) - \boldsymbol{x}\|^2}{\frac{1}{4}},$$
$$g_2(\boldsymbol{x}) = \ln P(\omega_2) - \frac{1}{2}\frac{\|\boldsymbol{\mu}_2 - \boldsymbol{x}\|^2}{\sigma^2} = \ln\frac{2}{3} - \frac{1}{2}\frac{\|(1,1,1) - \boldsymbol{x}\|^2}{\frac{1}{4}}.$$

Then we can build an optimal **dichotomizer**:

$$g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}), \quad \boldsymbol{x} = (x_1, x_2, x_3)^T$$

and the decision rule is

$$\boxed{\texttt{assign } \boldsymbol{x} \texttt{ to } \omega_1 \texttt{ when } g(\boldsymbol{x}) > 0, \texttt{and to } \omega_2 \texttt{ when } g(\boldsymbol{x}) < 0.}$$

Substituting, $g(\boldsymbol{x}) = -\ln 2 - 2\|\boldsymbol{x}\|^2 + 2\|(1,1,1) - \boldsymbol{x}\|^2$, which results in

$$(x_1 + x_2 + x_3) >^? \frac{3}{2} - \frac{1}{4}\ln 2 \approx 1.32.$$

The **prediction** for the test example $\boldsymbol{x}^* = (0.1, 0.7, 0.8)^T$ is $\boldsymbol{x}^* \in \omega_2$, given that $0.1 + 0.7 + 0.8 > 1.32$.

Computations in practice

In practical situations, only an i.i.d. data sample S is available. Let $S_k \subset S$ be the subset of observations known to belong to class ω_k. Then $S_1, \ldots, S_K$ is a partition of S. We can use **unbiased estimates** for the vector means and for the class priors:

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{|S_k|}\sum_{\boldsymbol{x} \in S_k} \boldsymbol{x}, \quad \hat{P}(\omega_k) = \frac{|S_k|}{|S|}.$$

If we know (or assume) that covariance matrices are **different** (we wish to use QDA):

$$\hat{\Sigma}_k = \frac{1}{|S_k| - 1}\sum_{\boldsymbol{x} \in S_k} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)^T.$$

Otherwise, if we know (of assume) that covariance matrices are **equal** (we wish to use LDA):

$$\hat{\Sigma}_{\text{pooled}} = \frac{1}{|S_k| - K} \sum_{k=1}^{K} (|S_k| - 1)\hat{\Sigma}_k.$$

Discussion

Bayesian classifiers are optimal when the class-conditional densities and priors are known; the methods are well-principled, fast and reliable. For Gaussian classes, we get a quadratic classifier $-$ QDA (if all covaraince matrices are equal, a linear classifier, LDA); using a specific distance function corresponds to certain statistical assumptions:

- If the class-conditional densities are far from the assumptions, the model will be poor.
- Even if the class-conditional densities are Gaussian, the parameters should be reliably estimated (particularly for QDA).
- Once we use sample statistics instead of population parameters, we lose optimality.

The question whether these assumptions hold can rarely be answered in practice; in most cases we are limited to posing and answering the question *"does this classifier give satisfactory predictions or not?"*.

Regularized Discriminant Analysis

If $d > |S_k|$ for some k, QDA cannot be applied, because the class covariance matrix $\hat{\Sigma}_k$ is singular. If $d > N$, neither QDA nor LDA can be used, because both $\hat{\Sigma}_k$ and $\hat{\Sigma}_{\text{pooled}}$ are singular. These problems can be overcome by applying **regularization**:

$$\hat{\Sigma}_k(\lambda, \gamma) := (1 - \gamma) \left[(1 - \lambda)\hat{\Sigma}_k + \lambda \hat{\Sigma}_{\text{pooled}} \right] + \frac{\gamma}{d} \text{tr}\left[\hat{\Sigma}_k(\lambda) \right] I_d,$$

where $\hat{\Sigma}_k(\lambda) = (1 - \lambda)\hat{\Sigma}_k + \lambda \hat{\Sigma}_{\text{pooled}}$. LDA is $(\lambda, \gamma) = (1, 0)$ and QDA is $(\lambda, \gamma) = (0, 0)$.

Pros & cons

Pros:

- Adaptable to all class-conditional distributions (not only Gaussian), even with mixed variables.
- Very resistant to overfitting the data sample.
- Accepts class priors and losses for misclasiffications.

Cons:

- Assumption of Gaussianity may be far from true.
- Needs sufficient examples epr class if we wish to use QDA.
- Requires matrix inversions (costly or numerically delicate).

The Naive-Bayes classifier

We showed that the 0/1 loss Bayes rule minimizing the probability of error could be formulated in terms of discriminant functions $g_k(\boldsymbol{x}) = P(\omega_k)P(\boldsymbol{x}|\omega_k)$, for $k = 1, \ldots, K$. We can expand the conditional probability to be

$$P(\omega_k)P(\boldsymbol{x}|\omega_k) = P(\omega_k)P(X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots \wedge X_d = x_d|\omega_k) =$$
$$= P(\omega_k)P(X_1 = x_q|\omega_k) \prod_{j=2}^{d} P(X_j = x_j|\omega_k, X_1 = x_1 \wedge \ldots \wedge X_{j-1} = x_{j-1}),$$

and if we assume $X_1, \ldots, X_d$ are pairwise independent *given the class*, this is equal to

$$\ldots = P(\omega_k)P(X_1 = x_1|\omega_k)\prod_{j=2}^{d} P(X_j = x_j|\omega_k) = P(\omega_k)\prod_{j=1}^{d} P(X_j = x_j|\omega_k) \stackrel{\text{def}}{\equiv} \boxed{\text{NB}_k(\boldsymbol{x})}.$$

Extensions

Multiplying numbers smaller than 1, we can easily get numeric errors in a computer, so we want to ease the computations. To do this, we can numerically extend (and ease) the classifier by taking logarithms as such

$$\text{NB}_k(\boldsymbol{x}) = \ln P(\omega_k) + \sum_{j=1}^{d} \ln P(X_j = x_j|\omega_k)$$

How do we deal with continuous variables? We have two options:

1. Assume a particular pdf for the variable and estimate its parameters from the data.
2. Discretize the variable and treat is as discrete.

Null empirical probabilities

In test examples $\boldsymbol{x}^*$ it may happen that some variable X_j has a value x_j^* **not present in the sample** used to create the classifier, hence $\hat{P}(X_j = x_j|\omega_k) = 0$ and we are in trouble. One way to avoid this kind of situations is to use the **Laplace correction**:

$$\hat{P}_L(X_j = x_j|\omega_k) = \frac{\text{card}(\{\boldsymbol{x} \in S_k : X_j = x_j\}) + p}{\text{card}(S_k) + p},\ p \in \mathbb{N}.$$

Here p is the "weight" assigned to the prior probability and V_k is the number of differents values of variable k.

The kNN classifier

Definition. *Metric.* Let X be a set. A metric in X is a two-parameter function $d : X \times X \to \mathbb{R}^+ \cup \{0\}$ satisfying the following properties $\forall x, y, z \in X$:

1. $d(x, y) = 0 \iff x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

A pair (X, d) is a metric space.

Decision rule. *The 1NN technique.* The 1NN technique classifies any $x \in X$ in the same class of the "nearest neighbour" of x in S, that is

$$\boxed{\text{The class of } x \text{ is the class of } \underset{x' \in S \setminus \{x\}}{\text{argmin}}\, d(x, x').}$$

Decision rule. *The kNN technique.* The kNN technique considers the $k \geq 1$ "nearest neighbours" of x in S and votes for the most represented class:

$$\boxed{\text{The class of } x \text{ is the majority class among its } k \text{ closest elements in } S.}$$

In this rule, ties may happen. These are broken randomly. There are no ties for two-class problems and odd k.

Theorem (Cover & Hart, 1965). *Asymptotic analysis of the 1NN technique.* Call $\varepsilon_{1\text{NN}}$ the probability of error of 1NN and ε_B be the Bayes error. Then, as $N \to \infty$,

$$\varepsilon_B \leq \varepsilon_{1\text{NN}} \leq \varepsilon_B \left(2 - \varepsilon_B \frac{K}{K-1}\right) \leq 2\varepsilon_B.$$

In particular, for two-class problems,

$$\varepsilon_B \leq \varepsilon_{1\text{NN}} \leq 2\varepsilon_B(1 - \varepsilon_B).$$

7. Classification theory and linear classification models (III).

Discriminative classifiers

Maximum Likelihood (ML) framework (I)

Generalized Linear Methods

Generalized linear models are a very general and classical technique for fitting **linear models**. They are the genuine representatives of **discriminative methods**. The main difference between discriminative and generative methods is that using generative methods, we can generate new data from the parameter estimation we have previously done, while in discriminative methods we neither can't or are interested in doing so.

These methods work for many **target types**:

- Binary (two-class) and nominal (multi-class).
- Proportions and counts.
- Ordinal (ordered classes).
- Continuous target variables.

They admit very **general predictors**. Categorical variables are binarized.

Definition. *Generalized linear model.* A GLM is a linear predictor of a convenient function h of the conditional expected value of the target variable. Mathematically,

$$h(\mathbb{E}[T_n|X_n]) = \boldsymbol{\beta}^T X_n + \beta_0.$$

This function h is typically smooth (of class $\mathcal{C}^k$ for some $k \geq 2$) and globally invertible. It's called the **link function**. We optimize the $\boldsymbol{\beta}$ and β_0 parameters directly, without any assumptions of the distribution of the $\boldsymbol{x}$. The target variable rows T_n are taken as independent and drawn from a distribution of the **exponential family**: Poisson, Gaussian, Bernoulli, Gamma, ...

The modeler chooses a suitable distribution for T_n given X_n:

1. If we are predicting a real continuous value, we will usually use the Gaussian distribution; hence, h is the identity function. This is called **linear regression**.
2. If we are in a two-class problem, we will use the Bernoulli distribution, with link function $h = \text{logit}$. This method is called **logistic regression**.
3. If we want to predict a counter, we will usually use the Poisson distribution, with link function $h = \ln$. This is called **Poisson regression**.

This generality comes at a cost: in general we need an iterative procedure for the optimization of the $\boldsymbol{\beta}$ and β_0 parameters. A popular procedure is to set it up as a Maximum Likelihood problem and use a preferred numerical optimization method (e.g. Newton-Rhapson).

Logistic regression

We are in the case of binary classification ($K = 2$). We **model** the posterior probability for class ω_1 as:

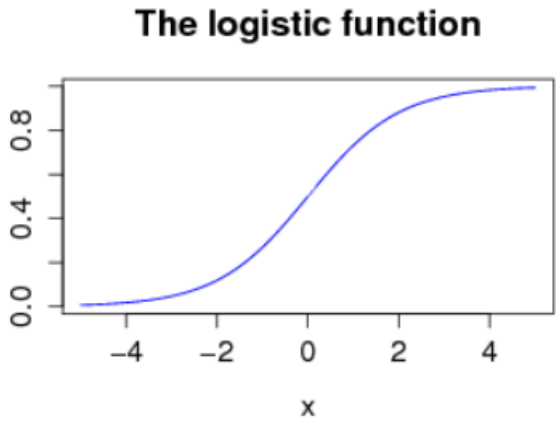
$$P(\omega_1|\boldsymbol{x}) = g\left(\boldsymbol{\beta}^T\boldsymbol{x} + \beta_0\right),$$

where $g(z) = \frac{\exp(z)}{1+\exp(z)} = \frac{1}{1+\exp(-z)}$ is the **logistic function**. Obviously, $P(\omega_2|\boldsymbol{x}) = 1 - P(\omega_1|\boldsymbol{x}) = 1 - g\left(\boldsymbol{\beta}^T\boldsymbol{x} + \beta_0\right)$.

The logistic function is a $\mathcal{C}^\infty$ function $g : \mathbb{R} \to (0,1)$. This function is a bijection, with inverse

$$g^{-1}(z) = \ln\left(\frac{z}{1-z}\right) \stackrel{\text{def}}{\equiv} \text{logit}(z)$$

for $z \in (0,1)$. This is called the **logit function**, and looks like this:



Each $T_n \sim B(p_n)$, where $p_n = g\left(\boldsymbol{\beta}^T X_n + \beta_0\right)$,

$$\implies P(T_n|X_n, \boldsymbol{\beta}) = \begin{cases} p_n, & T_n = 1 \ (X_n \in \omega_1) \\ 1 - p_n, & T_n = 0 \ (X_n \in \omega_2) \end{cases} = p_n^{T_n}(1-p_n)^{(1-T_n)}.$$

Notice that $g\left(\boldsymbol{\beta}^T X_n + \beta_0\right) = \mathbb{E}[T_n] = p_n$. Hence, we are identifying p_n with $P(\omega_1|X_n)$.

Interpretation of the model

The main thing to remember about logistic regression is that "The log of the odds is a linear function of the predictors". Since $P(\omega_1|X) = g\left(\boldsymbol{\beta}^T X + \beta_0\right)$, we have

$$\ln\left(\frac{P(\omega_1|X)}{P(\omega_2|X)}\right) = \ln\left(\frac{P(\omega_1|X)}{1-P(\omega_1|X)}\right) = \text{logit}\left(P(\omega_1|X)\right) = \boldsymbol{\beta}^T X + \beta_0.$$

Parameter obtaining

Suppose we have an i.i.d. sample of N labeled observations $S = \{(\boldsymbol{x}_n, t_n)\}_{n=1,\ldots,N}$, where $\boldsymbol{x}_n \in \mathbb{R}^n, t_n \in \{0,1\}$.

The first thing we notice is that we have $d+1$ parameters to fit. In the "equivalent" case of generative methods (LDA), we had $\frac{d(d+1)}{2} + 2d$ parameters. So, this cases scales better (linearly) than LDA did (quadratically). We re-write $P(\omega_1|X) = g(\boldsymbol{\beta}^T X)$, with

$$X = (1, X_1, \ldots, X_d)^T, \quad \boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)^T.$$

Now we see the obtaining of the parameters from a Maximum Likelihood perspective:

$$l(\boldsymbol{\beta}) = \ln \mathcal{L}(\boldsymbol{\beta}) = \ln P(\{t_n\}_{n=1}^N | \{\boldsymbol{x}_n\}_{n=1}^N, \boldsymbol{\beta}) =$$
$$\ln \prod_{n=1}^N P(t_n | \boldsymbol{x}_n, \boldsymbol{\beta}) = \sum_{n=1}^N \ln P(t_n | \boldsymbol{x}_n, \boldsymbol{\beta}) =$$
$$\sum_{n=1}^N \ln \left(g\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right)^{t_n} \left(1 - g\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right)\right)^{(1-t_n)} \right) =$$
$$\sum_{n=1}^N \ln \left((p_n)^{t_n} (1-p_n)^{(1-t_n)} \right), \quad p_n = g\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right).$$

Now,

$$(p_n)^{t_n} (1-p_n)^{(1-t_n)} = \left(\frac{p_n}{1-p_n} \right)^{t_n} (1-p_n) =$$
$$\left(\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right)\right)^{t_n} \left(1 + \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right)\right)^{-1}.$$

Therefore,

$$\boxed{l(\boldsymbol{\beta}) = \sum_{n=1}^N \left[t_n \boldsymbol{\beta}^T \boldsymbol{x}_n - \ln\left(1 + \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_n\right)\right) \right].}$$

Maximum Likelihood framework (II)