Chapter 1

Maximum likelihood : refresher

1.1 Basics

We recall some basics from Maximum Likelihood (ML) inference which is one of the building blocks of this course.

Assume that a sample of N observations $x_1, \ldots, x_N$ is given. Each observation x_i is the independent realisation of a random variable X following the probability distribution $p_\theta(x)$, depending on the parameter(s) θ (i.i.d. framework). The random variable X can be discrete (e.g. Bernoulli, Poisson, Binomial etc.) or continuous (e.g. Gaussian, Exponential etc.). It can be a scalar (e.g. univariate Gaussian) or live in $\mathbb{R}^D$, with $D > 1$ (e.g. multivariate Gaussian). Similarly θ may be a single parameter (e.g. Exponential distribution) or a set of parameters (e.g. Gaussian distribution).

Our aim is to estimate the (most likely) value of θ given the data $x_1, \ldots, x_N$. The maximum likelihood estimate of θ can be found by solving the following optimization problem

$$\begin{aligned} \hat{\theta}_{ML} :&= \arg\max_{\theta \in \mathcal{D}_\theta} p_\theta(x_1, \ldots, x_N) \\ &= \arg\max_{\theta \in \mathcal{D}_\theta} \prod_{i=1}^{N} p_\theta(x_i), \end{aligned} \tag{1.1}$$

where $\mathcal{D}_\theta$ is the domain set inside which we look for the optimal value of θ (e.g. $\mathbb{R}^+$ for the Exponential distribution) and the second equality comes from independence. Since logarithm is a monotonic function, in order to "ease" calculations it is customary to attack the following maximization problem,

equivalent to (1.1) in order to compute the ML estimate of θ

$$\max_{\theta \in \mathcal{D}_\theta} \left(\sum_{i=1}^{N} \log p_\theta(x_i) \right). \tag{1.2}$$

Still equivalently, but more common in the machine learning literature, one can minimize the following **loss** function

$$- \min_{\theta \in \mathcal{D}_\theta} \left(\sum_{i=1}^{N} \log p_\theta(x_i) \right). \tag{1.3}$$

Remark (Subtle)**.** *Since we focus on a* single *sample of N observations, $x_1, \dots, x_N$ are given (not random) and $\hat{\theta}_{ML}$ is an estimate (not random). However, if we allow $x_1, \dots, x_N$ to change in the universe of all possible samples of N observations, they become random variables and so does $\hat{\theta}_{ML}$, thus becoming and* ***estimator****. Under proper (mild) assumptions, maximum likelihood estimators have several nice properties, including consistency and asymptotic normality. Discussing these properties goes beyond the scope of this course, the interested reader is referred to Wasserman (2004), Ch.9.*

When p_θ is a differentiable function of θ, the standard approach to solve any of the above problems is to compute the **gradient** w.r.t. θ of the objective function

$$l(\theta) := \sum_{i=1}^{N} \log p_\theta(x_i)$$

and set it to zero in order to find stationary points.

Example. Consider $\theta \in \mathbb{R}^+$ and $p_\theta(x) = \theta e^{-\theta x}$ with $x > 0$ (Exponential law). Then

$$l(\theta) = N \log \theta - \theta \sum_{i=1}^{N} x_i.$$

The derivative $l'(\theta)$ can easily be computed and set equal to zero in order to find $\hat{\theta}_{ML} = \frac{N}{\sum_{i=1}^{N} x_i}$ (**exercise**).

In general, problems (1.1)-(1.2)-(1.3) cannot always be solved (easily). Indeed, p_θ might be not differentiable nor convex (local maxima/minima).

And although if it was, explicit formulas for $\hat{\theta}_{ML}$ might not be available and the best we can do is to look for numerical solutions to problem (1.1) (and (1.2) and (1.3)) and this is the precise framework we deal with in this course. However, before diving into the core of this module, it is worth spending some time with two important models whose parameters can be inferred from the data via ML estimation: the linear Gaussian model and logistic regression.

1.2 Gaussian linear regression

We consider the linear regression model

$$Y = X\beta + \epsilon, \tag{1.4}$$

where $Y = (y_1, \ldots, y_N)^T$ are N observed **response** variables in $\mathbb{R}$, $X \in \mathbb{R}^{N \times P}$, with $P < N$, is the **feature** matrix whose i-th row correspond the P feature values observed for the i-th individual and it is assumed that

$$\text{rank}(X) = P$$

so that $X^T X$ is invertible. The unknown parameter β is a column vector in $\mathbb{R}^P$ and must be estimated and $\epsilon = (\epsilon_1, \ldots, \epsilon_N)^T$ are i.i.d. residuals such that

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Also σ^2 is unknown and must be estimated. Although very simple, this generative model is extremely important for at least two reasons: i) it serves as a building block for the construction of more sophisticated models either linear or not and ii) is it fully interpretable, meaning that once fit to the data, we immediately know whether a feature is positively, negatively or not correlated with the output, which is not the case with more sophisticated models (e.g. non parametric regression models, such as neural networks).

With the notation introduced in the previous section, our observations are $y_1, \ldots, y_N$. The i-th observation is an observed outcome of the distribution

$$p(y_i; X, \beta, \sigma^2) = \mathcal{N}(y_i, X_i\beta, \sigma^2),$$

where X_i is the i-th row of X (and $X_i\beta$ is the standard dot product between vectors). Two remarks: first, the set of parameters now include the unknown parameters $\theta := (\beta, \sigma^2)$ and the known one X. Second, our observations are

independent (why?) but *not* identically distributed (why?). We thus have everything we need in order to compute the log-likelihood

$$l(\theta) = -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\|Y - X\beta\|_2^2 + C \tag{1.5}$$

where C contains all the terms not depending on θ and $\|\cdot\|_2$ denotes the Euclidean norm. When computing the gradient with respect to $\theta = (\beta, \sigma^2)$ and setting it equal to zero, we find the following normal equations

$$\nabla_\beta l(\theta) = -\frac{1}{2\sigma^2}(2X^TX\beta - 2X^TY) = 0 \tag{1.6}$$

$$\nabla_{\sigma^2} l(\theta) = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\|Y - X\beta\|_2^2 = 0. \tag{1.7}$$

As is can be seen, Eq. (1.6) can be solved independently from σ^2 and leads to the famous OLS formula

$$\hat{\beta} := (X^TX)^{-1}X^TY,$$

thus showing that the MLE and OLS estimator are the same for the Gaussian linear model. Replacing β with $\hat{\beta}$ in Eq. (1.7) and solving for σ^2 one finds the MLE estimator of σ^2

$$\hat{\sigma}^2 := \frac{1}{N}\|Y - X\hat{\beta}\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\hat{\epsilon}_i^2,$$

where $\hat{\epsilon}_i$ are the OLS residuals. Several extensions of the linear model described so far exist allowing one to account for i) non-linear relationships between the response variables and the features, ii) autocorrelated residuals possibly not having the same variance (heteroskedasticity) or not being normally distributed, iii) ... Here we just wish to mention the special case where

$$\text{rank}(X) < P$$

which systematically happens in high dimensional statistics ($P > N$). Well, in that case, Eq. (1.6) does not admit a unique solution. One option is to compute the pseudo-inverse[1] of X^TX in order to select a least squares solution

[1] https://en.wikipedia.org/wiki/Moore_Penrose_inverse

among the infinite ones. An elegant alternative (also having other major advantages) is to consider a penalized version of Eq. (1.5) such as

$$l_R(\theta) := -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\|Y - X\beta\|_2^2 - \frac{\lambda}{2\sigma^2}\|\beta\|_2^2 + C$$

where λ is a positive parameter. This solution (**Ridge** regression) leads to the following alternative normal equation

$$\nabla_\beta l_R(\theta) = -\frac{1}{2\sigma^2}(2(X^TX + \lambda I_P)\beta - 2X^TY) = 0$$

replacing Eq. (1.6), where I_P is the identity matrix of order P. This equation always admits a unique solution $\hat{\beta}_R := (X^TX + \lambda I_P)^{-1}X^TY$ since the matrix $(X^TX + \lambda I_p)$ is always non-singular independently on the rank of X.

1.3 Logistic regression

Whereas in the previous section the response variable y_i was assumed to be continuous, we now consider the case it is binary (0-1). Thus, we face a **classification** problem since the i-th observation, described by its feature vector X_i (the i-th row of X) either belongs to one class ($y_i = 1$) or another ($y_i = 0$). Our aim is to learn a rule (i.e. to train a **classifier**) allowing us to predict the class of each observation based on its feature vector. One of the simplest yet powerful classifiers one can think of is the logistic regression model. Indeed, each observed y_i is seen as the outcome of a Bernoulli random variable Y_i, whose conditional probability of success $\mathbb{P}(Y_i = 1|X_i)$ is denoted by p_i. The N random variables $Y_1, \ldots, Y_N$ are assumed to be independent (not identically distributed) and the main assumption is

$$\log\left(\frac{p_i}{1-p_i}\right) = X_i\beta, \tag{1.8}$$

with β still being an unknown parameter in $\mathbb{R}^P$. The likelihood for such a model is

$$\mathcal{L}(\beta) = \prod_{i=1}^{N} p_i^{y_i}(1-p_i)^{1-y_i}$$

and the corresponding log-likelihood

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{N} \left[y_i \log\left(\frac{p_i}{1-p_i}\right) + \log\left(1-p_i\right) \right] \\ &= \sum_{i=1}^{N} \left[y_i X_i \beta - \log(1+e^{X_i\beta}) \right], \end{aligned} \tag{1.9}$$

where we replaced $p_i = \frac{e^{X_i\beta}}{1+e^{X_i\beta}}$ (from Eq. (1.8)) in the first of the above equations in order to obtain the second. As we did for the linear Gaussian model, also in this case we might wish to consider the following penalized log-likelihood

$$l_R(\beta) = \sum_{i=1}^{N} \left[y_i X_i \beta - \log(1+e^{X_i\beta}) \right] - \frac{\lambda}{2}\|\beta\|_2^2, \tag{1.10}$$

in order to reduce the risk of over fitting and possibly achieve a better generalisation. Our aim being to maximize the above quantity with respect to β, it is easy to show that (**exercise**):

$$\nabla_\beta l_R(\beta) = \sum_{i=1}^{N} (y_i - p_i) X_i^T - \lambda\beta. \tag{1.11}$$

Note that β appears in the above equation two times: multiplied by λ (positive scalar) and "inside" $p_i = \frac{e^{X_i\beta}}{1+e^{X_i\beta}}$. Due to the last term, when setting the above gradient equal to zero, we realize that the equation cannot be solved in β analytically (not even when $\lambda = 0$). Thus, we encounter for the first time one of the difficult scenarios mentioned at the end of Section 1.1.

Although an analytical solution can't be obtained, we can still estimate $\hat{\beta}_R$ numerically via **gradient ascent**. Indeed, given the current value of the parameter β, say β_c, we can compute the gradient at β_c via Eq. (1.11) and calculate an update of β (say β_n) as

$$\beta_n = \beta_c + \alpha \nabla_\beta l_R(\beta_c), \tag{1.12}$$

where $\alpha > 0$ is a user defined learning rate. Algorithm 1 illustrates a pseudo-code of the optimization algorithm to obtain $\hat{\beta}_R$. The update in the above equation is iterated up to "convergence". Indeed, when a stationary point is

Algorithm 1 Numerical optimization of β

Require: β_0 ▷ the initial random value of β
  $\beta_c \leftarrow \beta_0$
  while not convergence **do**
    $\beta_n = \beta_c + \alpha \nabla_\beta l_R(\beta_c)$
    $\beta_c = \beta_n$
  end while

reached the gradient is null and $\beta_n = \beta_c$, this is what we mean by convergence. Concretely, one fixes a small positive ϵ (something like 1.0^{-4}) and say that convergence is reached as soon as $|l_R(\beta_n) - l_R(\beta_c)| < \epsilon$.

Now although moving in the direction of the gradient allows us to reach a stationary point corresponding to a local maximum, how can we be sure that the stationary point actually is $\hat{\beta}_R$? In more general frameworks, for instance when the classifier is a neural net, the answer to the above question is that we can't. However, for the logistic regression it can be shown that the penalized log-likelihood in Eq (1.9) is strictly concave, thus admitting a unique global maximum. You can see Appendix 1.A for a proof of this statement. Moreover, that proof also suggests an alternative gradient ascent algorithm which converges to $\hat{\beta}_R$ much more rapidly.

1.A Concavity of the logistic regression model's log-likelihood

As a preliminary remark, let us notice that

$$\begin{aligned}\nabla_\beta(p_i) &= \nabla_\beta\left[\frac{1}{1+e^{-X_i\beta}}\right] \\ &= \left[\frac{e^{-X_i\beta}}{(1+e^{-X_i\beta})^2}\right] X_i^T = p_i(1-p_i)X_i^T.\end{aligned} \tag{1.13}$$

We can now compute the gradient with respect to β of the j-th entry of vector in Eq. (1.11) in order to get the j-th column of the Heissian matrix of $l_R(\beta)$ in Eq (1.10):

$$\nabla_\beta\left[\nabla\beta(l_R(\beta))\right]_j = -\sum_{i=1}^{N}\left(p_i(1-p_i)X_i^T x_{ij}\right) - \lambda e_j,$$

where e_j is a P-vector with all entries equal to zero except for the j-th one, equal to 1. From the above equation we easily conclude that

$$H_\beta l_R(\beta) = -\sum_{i=1}^{N}\left(p_i(1-p_i)X_i^T X_i\right) - \lambda I_P. \tag{1.14}$$

It can be easily verified (**exercise**) that $H_\beta l_R(\beta)$ is negative definite for all $\beta \in \mathbb{R}^P$. Thus $l_R(\cdot)$ is a concave function of β with a single stationary point being a global maximum. Moreover, the Newton-Raphson algorithm can be employed to replace the update rule in Algorithm 1 with

$$\beta_n = \beta_c - \left(H_\beta l_R(\beta_c)\right)^{-1}\nabla_\beta l_R(\beta_c).$$

The above sequence will converge faster to the stationary point than (full-batch) gradient descent. However the calculation of the inverse of Heissian matrix might be infeasible (or not interesting) in large datasets.

Exercise. Write a code to implement Algorithm 1 as well as the Newton-Raphson alternative. Comment on the number of iterations needed by each in order to reach convergence.