Model fitting for classification/regression of binary outputs (YES/NO): problem statement

Antonio Sala

Modeling, Identification & Control of Complex Systems

Universitat Politècnica de València

Presentations in video: http://personales.upv.es/asala/YT/V/**clasifintr1EN**.html, http://personales.upv.es/asala/YT/V/**clasifintr2EN**.html, http://personales.upv.es/asala/YT/V/**clasifNoLSEN**.html

Outline

Motivation:

Devoting more "study time" to a course should increase the "likelihood of passing". Presence of "Nigeria" and "inheritance" should increase the "probability of being junk mail"... I wish to know if this is a picture of a "Dog"...

Objectives:

Understanding which are the problems to be posed when I must fit some labelled data, distinguishing it from least-squares fitting.

Contents:

Problem statement: goals, examples. Conclusions
Appendix: Why not just carrying out least squares fit as usual?

The problem of binary supervised classification

We have a dataset (x_i, y_i) of samples of variables X and Y with:

- $x_i \in \mathbb{R}^n$ (or "cathegorical" components {0,1}), in general $x_i \in \mathbb{X}$
- $y_i \in \{0, 1\}$ cathegorical; [known, **supervised** learning]

The meaning of "cathegorical" is that 0 and 1 are "labels", not "numbers to carry out algebraic operations", at least in principle.
We will assume "binary" for simplicity, albeit we might have multi-class problems {"Dog", "Cat", "Flower"}...
*We can always translate to binary with $y \in \{0, 1\}^3$ ={"Is it a Dog?", "is it a Cat?", "is it a Flower?"}; "Dog" class would be labelled as $\{1, 0, 0\}$; "Not a Dog" labelled as $\{0, X, X\}$.

Examples:

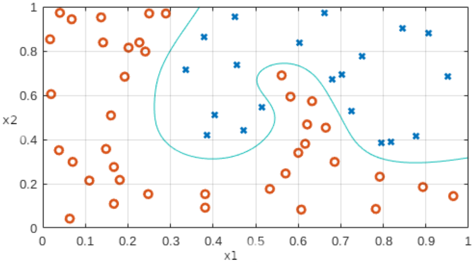
- Poll on "study time" + exam results: {John: (78h, pass); Mary: (22h, fail); Anne: (72h, fail), . . . }
- {Picture 1: dog; Picture 2: NOT dog; Picture 3: dog, . . . }

Goals (1)

[A] Perfect classification, learn ALL labels of (x_i, y_i).

- It might not be posible (apart from brute force, "rote learning", that does not "generalise") or not advisable (predicting test results from study time: 72h pass, 73h fail, 74h pass)... Or it may be possible in many ways but we wish to classify in the "best" way, in a particular sense.

Example: ["letters" from "32x32 pixels"] ideally we wish perfect fitting; fig. below



$(x_1, x_2) \in \text{RED} \Leftrightarrow f(x_1, x_2, \theta^*) < 0$

Goals (2)

[B] Imperfect matching, “learn” parameters θ of $f(x,\theta)$ that:

- With $f(x,\theta):\mathbb{X}\mapsto\{0,1\}$, minimize some “error” measure, loss function $\mathcal{L}$, $\mathcal{L}(y_i, f(x_i,\theta))$, even assymetric with different loss for false positives (diabetes diagnosis for a healthy patient) and false negatives (undiagnosed diabetes).

$$\mathcal{L}(1,1)=\mathcal{L}(0,0)=0; \qquad \mathcal{L}(1,0)=7, \;\; \mathcal{L}(0,1)=2.$$

[deterministic interpretation]

Goals (3)

[C] **Imperfect** matching, "learn" parameters θ of $f(x,\theta)$ that:

- With $f(x,\theta) : \mathbb{X} \mapsto [0,1]$, maximize likelihood of all y_i given x understanding $f(x,\theta) \equiv p_\theta(y=1|X=x)$.

[probabilistic interpretation]

Trivial solution for deterministic/probabilistic setups: $f(x_i) = 1$ if $y_i = 1$; $f(x_i) = 0$ it $y_i = 0$. [rote learning]

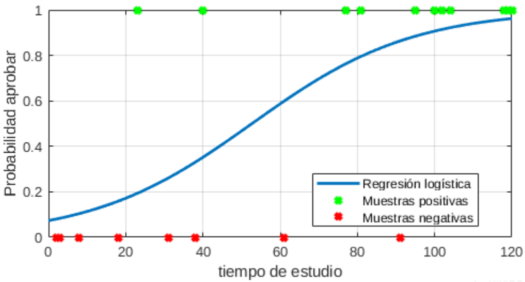
Not valid... the "shape f" must root on some base theory/assumption; at least, it must be "sensibly smooth"... that is what the parametrization θ encodes.

Goals (3b)

There are some popular "shapes of f" in literature for the probabilistic version (logit, probit, ...) each justified from some underlying assumptions.

Example: ["prob. passing" as a function of "study time"] we don't want to "match" all data samples.

$$Prob(pass, | x) \approx f(x, \theta^*)$$



Conclusions

We discussed the meaning of "fitting models $f_\theta(input) := f(input, \theta)$ to yes/no labelled data".

[A] perfect classification (letters, images)... decision rule $f_\theta(input) > 0$.

[B] If perfect is not possible, minimise cost related to false positives and false negatives.

[C] Sometimes, a probabilístic interpretation is sought... $p(y = true|x) = g_\theta(x)$.

- These options are closer than it might seem:

  (1) A more positive value of f might indicate how sure the algorithm is of its ouptut. Sometimes not failing is impossible: recognising "dog" from "the average green intensity in pixels" isn't easy.

  (2) Sometimes, a probabilistic cost is first optimized and, later, a threshold is decided to classify as one class or the other, depending on the importance of false negatives or false positives.

Why not sticking to least squares as usual?

Output data $\{0, 1\}$ could be fitted by minimising $\mathcal{L}(\theta) = \sum_i (y_i - f(x_i, \theta))^2$ with linear regression, neural network, polynomial or whatever. Why not doing it that way? Why complicating things with other tools? **It may work and be very computationally efficient** (for $f(x_i, \theta) = \Phi^T(x_i) \cdot \theta$), but...

"Perfect" classification:

- In principle, it may be a valid option in a "deterministic" setting: we wish to fit a function that returns '0' or '1' when required... but maybe other options achieve the same with "simpler" functions...

  Indeed, we are just interested in $f(x_i, \theta) > 0.5$ in positive samples, $f(x_i, \theta) < 0.5$ in negative ones. Maybe a "simple" function achieves that (no problem in $f(x_7, \theta) = -1241$) but does not "fit" the data (forcing $f(x_7, \theta) \approx 0$ may distort f elsewhere if it is not "flexible" enough)... but adding 'sign' loses 'gradient'.

Why not sticking to least squares as usual?

Output data $\{0, 1\}$ could be fitted by minimising $\mathcal{L}(\theta) = \sum_i (y_i - f(x_i, \theta))^2$ with linear regression, neural network, polynomial or whatever. Why not doing it that way? Why complicating things with other tools? **It may work and be very computationally efficient** (for $f(x_i, \theta) = \Phi^T(x_i) \cdot \theta$), but...

"Perfect" classification:

- In principle, it may be a valid option in a "deterministic" setting: we wish to fit a function that returns '0' or '1' when required... but maybe other options achieve the same with "simpler" functions...

  Indeed, we are just interested in $f(x_i, \theta) > 0.5$ in positive samples, $f(x_i, \theta) < 0.5$ in negative ones. Maybe a "simple" function achieves that (no problem in $f(x_7, \theta) = -1241$) but does not "fit" the data (forcing $f(x_7, \theta) \approx 0$ may distort f elsewhere if it is not "flexible" enough)... but adding 'sign' loses 'gradient'.

More comments on this at, say, https://notesonai.com/Least+squares+for+classification

Why not sticking to least squares as usual?

Output data $\{0,1\}$ could be fitted by minimising $\mathcal{L}(\theta) = \sum_i (y_i - f(x_i,\theta))^2$ with linear regression, neural network, polynomial or whatever. Why not doing it that way? Why complicating things with other tools? **It may work and be very computationally efficient** (for $f(x_i,\theta) = \Phi^T(x_i)\cdot\theta$), but...

"Imperfect" classification:

- Assymetric cost to 'false +' or 'false −' requires modifying $\mathcal{L}$.
- In probabilistic settings, quadratic error is the log-likelihood of $e^{-\epsilon^2/\sigma^2}$ (normal distribution), but it does not "feel correct" with $\{0,1\}$ outputs (Bernoulli).
- In quite a few cases maybe the "truncated" quadratic might seem a more sensible choice: $\mathcal{L}(y_i, f_i) = \begin{cases} \mathbf{0} & y_i = 1 \,\&\, f_i \geq 1 \mid y_i = 0 \,\&\, f_i \leq 0 \\ (y_i - f_i)^2 & \text{rest of cases} \end{cases}$

  That would give additional flexibility to "f", to solve problems with a lower number of adjustable parameters... and well, we might even think on more complicated $\mathcal{L}$, of course.

More comments on this at, say, https://notesonai.com/Least+squares+for+classification