Logistic Regression

Prof. Alessandro Lucantonio

Aarhus University

31/10/2023

Binary classification

Recall:

- Classification $\rightarrow$ discrete target.
- Binary classification: $\{0, 1\}$ target. Example: spam/not spam e-mails.

Idea: consider a hypothesis (threshold) such that

$$0 \leq h_{\boldsymbol{w}} \leq 1$$

and

- if $h_{\boldsymbol{w}}(\boldsymbol{x}) \geq 0.5$, predict 1;
- if $h_{\boldsymbol{w}}(\boldsymbol{x}) < 0.5$, predict 0.

Logistic Regression

In particular, take $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$, where

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

is the **sigmoid function**. $h_{\boldsymbol{w}}$ gives us the **probability** that the output is 1.

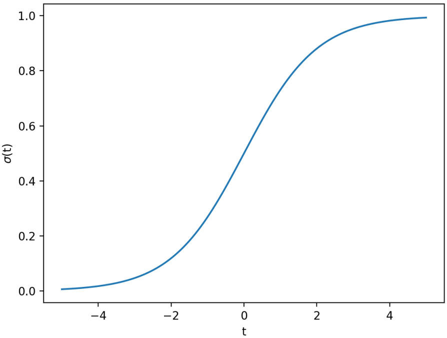

Figure: Sigmoid function

Linear decision boundary

Model: $h_{\boldsymbol{w}}(x_1, x_2) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$

$h_{\boldsymbol{w}}(x_1, x_2) \geq 0.5$ when $w_0 + w_1 x_1 + w_2 x_2 \geq 0 \implies y = 1$
$h_{\boldsymbol{w}}(x_1, x_2) < 0.5$ when $w_0 + w_1 x_1 + w_2 x_2 < 0 \implies y = 0$

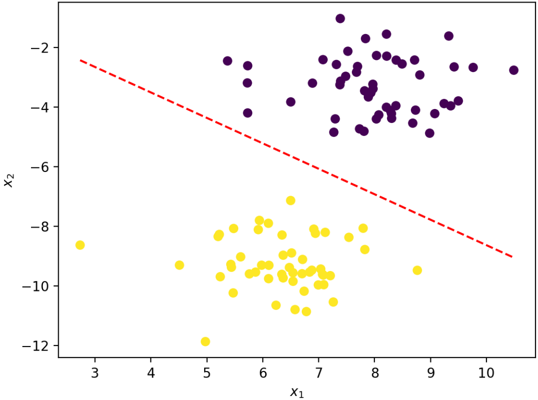

Figure: An example of linear decision boundary

Non-linear decision boundary

Model: $h_{\boldsymbol{w}}(x_1, x_2) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$

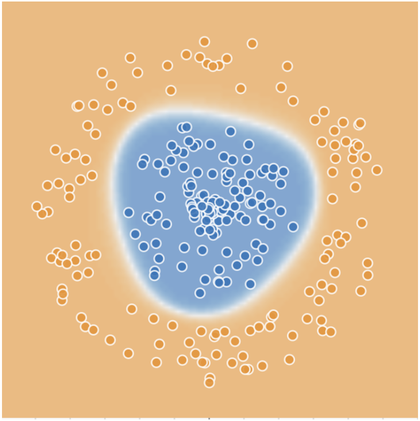

Figure: An example of non-linear decision boundary

Cost function

First attempt: MSE

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=0}^{N}(\sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}^{(i)}) - y^{(i)})^2$$

Problem: σ is *non-convex*, hence MSE is *non-convex* (possibly many local minima).

Main idea: consider the a loss term such that

- $-\log h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})$ if $y = 1$,
- $-\log(1 - h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})) = 0$ if $y = 0$

Notice that:

- if $y = 1$ and $h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) = 1$, $-\log h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) = 0$;
- if $y = 1$ and $h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) = 0$, $-\log h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) \to \infty$;
- if $y = 0$ and $h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) = 1$, $-\log(1 - h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})) \to \infty$;
- if $y = 0$ and $h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) = 0$, $-\log(1 - h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})) = 0$;

Cross-entropy

Combine the log loss terms: **Binary cross-entropy**

$$E(\boldsymbol{w}) := -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log(h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}))$$

This cost function is *convex* (sum of convex terms) with respect to the weights.

Vectorized version:

$$E(\boldsymbol{w}) = -\frac{1}{N}\left(\boldsymbol{y}^T \log(h_{\boldsymbol{w}}(\mathsf{X})) + (1 - \boldsymbol{y}^T) \log(1 - h_{\boldsymbol{w}}(\mathsf{X}))\right)$$

with $h_{\boldsymbol{w}}(\mathsf{X}) = \sigma(\mathsf{X}\boldsymbol{w})$.

Convexity

Any local minimum of a convex function is also a global minimum. Instead, a non-convex function has potentially many local minima and *saddle points* (vanishing gradient but neither minimum nor maximum).

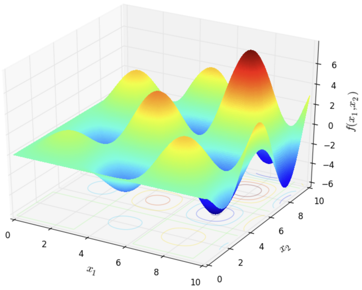

Figure: An example of a non-convex function

Remember the main challenge/goal of ML: **generalization**.

- A global minimum of the cost function corresponds to the best fit of the training set: this may lead to *overfitting*.

Derivative of the sigmoid

Goal: computing the gradient of the cross-entropy with respect to the weights.

Recall: $\sigma(t) := \frac{1}{1+e^{-t}}$.

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\sigma(t) &= \frac{e^{-t}}{(1+e^{-t})^2} \\
&= \left(\frac{1}{1+e^{-t}}\right)\left(\frac{e^{-t}}{1+e^{-t}}\right) \\
&= \sigma(t)\left(1-\frac{1}{1+e^{-t}}\right) \\
&= \sigma(t)(1-\sigma(t)).
\end{aligned}
$$

Gradient of the cross-entropy /1

(assuming sum on repeated indices)

$$N\frac{\partial}{\partial \boldsymbol{w}_j}E(\boldsymbol{w}) = -\left[\frac{y^{(i)}\frac{\partial}{\partial w_j}h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})}{h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})} - \frac{(1-y^{(i)})\frac{\partial}{\partial \boldsymbol{w}_j}(1-h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}))}{1-h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})}\right]$$

with

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{w}_j}h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) &= \sigma'(\boldsymbol{w}^T\tilde{\boldsymbol{x}}^{(i)})\tilde{\boldsymbol{x}}_j^{(i)} = \sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}^{(i)})(1-\sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}^{(i)}))\tilde{\boldsymbol{x}}_j^{(i)} \\ &= h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})(1-h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}))\tilde{\boldsymbol{x}}_j^{(i)}\end{aligned}$$

$\rightarrow$

Gradient of the cross-entropy /2

$$
\begin{aligned}
N\frac{\partial}{\partial \boldsymbol{w}_j}E(\boldsymbol{w}) &= -[y^{(i)}(1-h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}))\tilde{\boldsymbol{x}}_j^{(i)} - (1-y^{(i)})h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})\tilde{\boldsymbol{x}}_j^{(i)}] \\
&= -[y^{(i)} - h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)})]\tilde{\boldsymbol{x}}_j^{(i)} \\
&= [h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) - y^{(i)}]\tilde{\boldsymbol{x}}_j^{(i)}.
\end{aligned}
$$

Final result:

$$
\frac{\partial}{\partial \boldsymbol{w}_j}E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=0}^{N}[h_{\boldsymbol{w}}(\tilde{\boldsymbol{x}}^{(i)}) - y^{(i)}]\tilde{\boldsymbol{x}}_j^{(i)}.
$$

Vectorized version:

$$
\nabla E(\boldsymbol{w}) = \frac{1}{N}\mathrm{X}^T(\sigma(\mathrm{X}\boldsymbol{w}) - \boldsymbol{y}).
$$

Multi-class classification

Targets: $y \in \{0, \ldots, k\}$

Idea: solve $k+1$ binary classification problems. Given a data sample $\boldsymbol{x}^{(i)}$

- For each $0 \leq j \leq k$, compute the probability $h_{\boldsymbol{w}}^{(j)}(\tilde{\boldsymbol{x}}^{(i)})$ that $\tilde{\boldsymbol{x}}^{(i)}$ belongs to the class j.
- The prediction will be the class that corresponds to the maximum probability, *i.e.*

$$\arg\max_{j} h_{\boldsymbol{w}}^{(j)}(\tilde{\boldsymbol{x}}^{(i)})$$