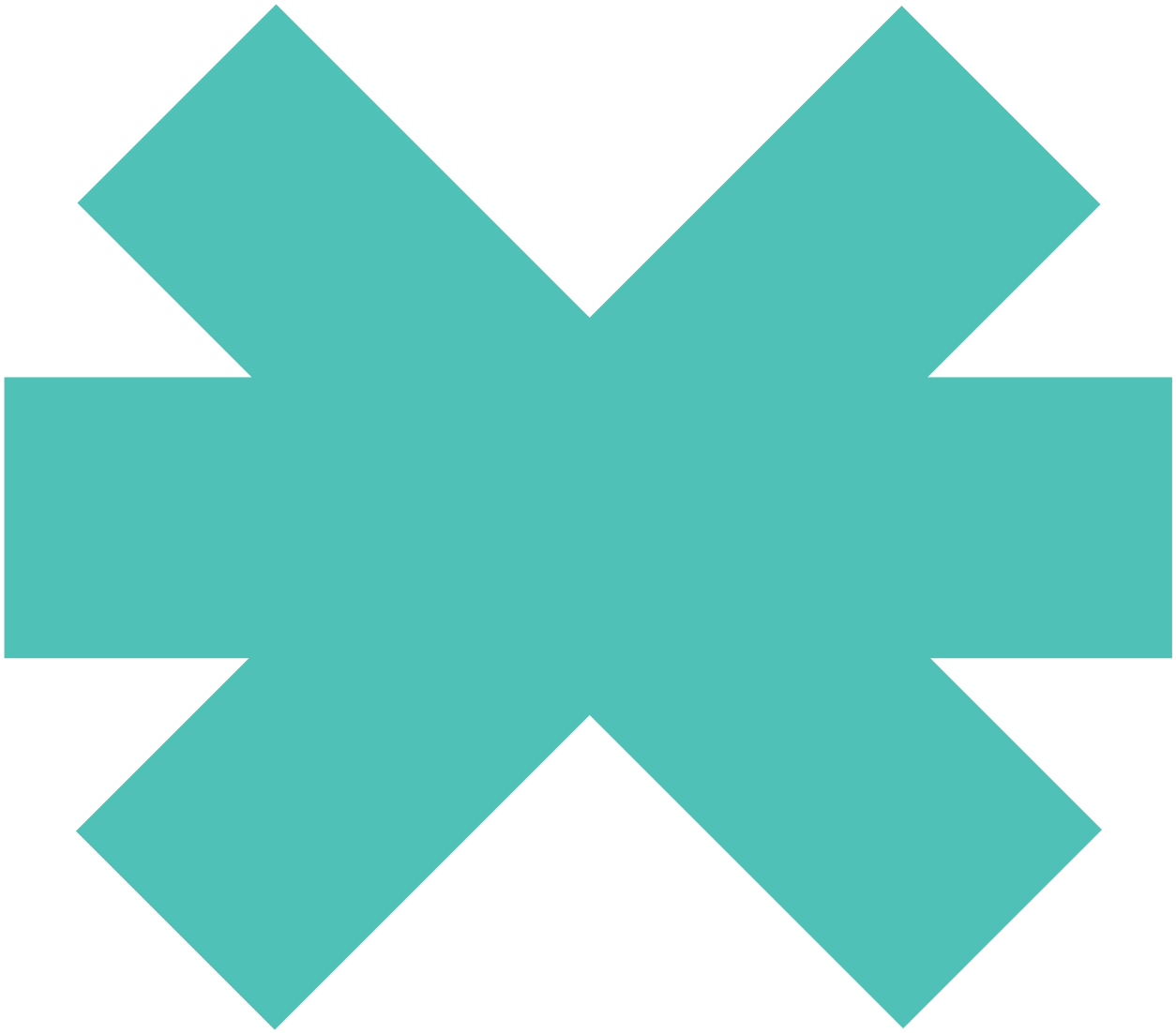
MÓDULO 2
Aprendizagem supervisionada
Especialização em Machine Learning

8

Regressão logística

New Technology School

Tokio.



8 Regressão logística

Sumário

A regressão logística é semelhante à regressão linear, mas, neste caso, em vez de estar destinada a realizar uma predição sobre um valor numérico contínuo, classificamos os exemplos entre duas ou mais classes, previamente conhecidas e definidas.

A sua implementação é, assim, semelhante à regressão linear, pois codificamos as classes com valores numéricos, mantendo um algoritmo e equações semelhantes.

Do mesmo modo, continuamos a procurar uma equação ou relação linear entre as variáveis que separem os exemplos, finalmente classificados entre as distintas classes.

Para passar de um valor numérico para uma das classes consideradas, utilizaremos uma função logística (ou lógica) que tenha em conta apenas valores lógicos de *0* e *1*.

APLICAÇÕES

- Saúde: classificação de amostras para detetar distintas doenças.
- Astronomia: classificação de sinais para localizar planetas, asteroides e outros corpos celestes.
- Classificação de utilizadores segundo diversas categorias: de nível económico "baixo", "médio" e "alto", segundo o seu comportamento, segundo a sua probabilidade de possuir um alto valor durante o seu ciclo de vida como utilizador, etc.
- Reconhecimento de imagens por visão computacional, reconhecimento ótico de caracteres, etc.
- Classificação de frutos segundo a gama.

8.1 Modelação de dados

Procuramos uma função linear que defina uma linha divisória entre duas classes de dados.

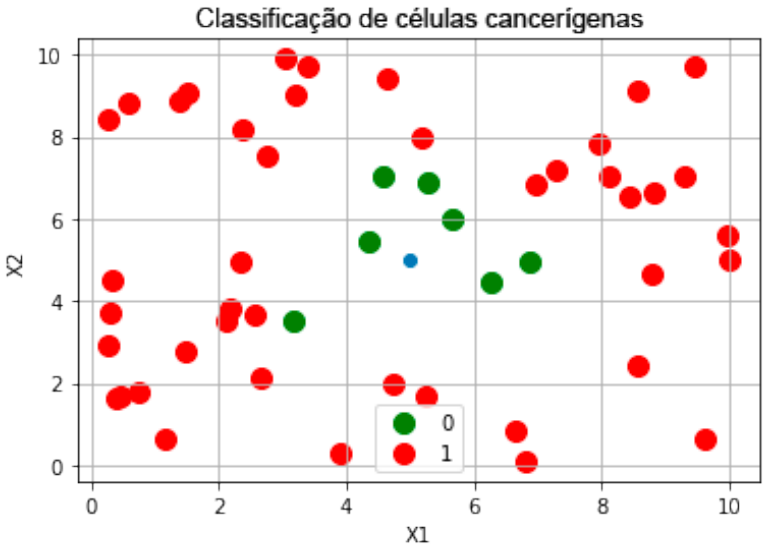


Figura 21

Exemplos:

- Células cancerígenas e não cancerígenas.
- Utilizador de alto valor ou não.
- Possível cliente moroso ou não, ao conceder um empréstimo.
- É ou não um planeta.
- "É um cachorro quente ou não" ("*Hot dog or not*", a aplicação que surge na série *Silicon Valley*).

Por vezes, encontrar uma função linear que separe essas classes não é possível, logo veremos outros métodos para a selecionar.

8.2 Classificação binária e multiclasse

Na regressão linear apenas podemos considerar duas classes por modelo, logo toda a classificação é, essencialmente, binária.

Para alcançarmos uma regressão logística ou classificação multiclasse, simplesmente treinamos um modelo por cada classe, que distribuirá os exemplos entre os pertencentes a essa classe e os não pertencentes (os que pertençam a qualquer outra classe), conseguindo assim estabelecer uma classificação multiclasse a partir de múltiplas classificações binárias.

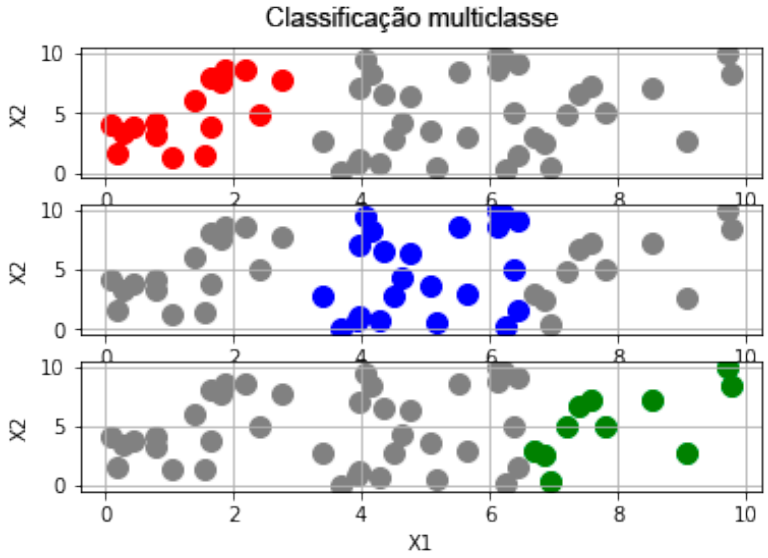


Figura 22

8.3 Hipótese

Para a regressão logística, utilizaremos uma hipótese diferente da regressão linear:

- Se *Y = 0*, catalogaremos o exemplo como a classe base, isto é, a classe nula ou a contrária à que consideramos na classificação binária. Por exemplo: "não é uma célula cancerígena".
- Pelo contrário, se *Y = 1*, catalogá-lo-emos como a classe a detetar. Por exemplo: "sim, é uma célula cancerígena".

Para passar de um valor contínuo a um valor binário, *0* ou *1*, usaremos uma das diferentes funções de ativação disponíveis:

$$Y = h_\theta = g(\Theta \times X);$$
$$Y \in [0, 1]$$

Figura 23

Código Latex:

```
Y = h_ \theta = g(\Theta \times X); \\
Y \in [0,1]
```

Assim, treinaremos o modelo para obter a *theta* ótima, enquanto, para realizar predições, multiplicaremos o vetor de Θ e o vetor *X* e aplicaremos a função logística, sobre o resultado correspondente, para o transformar em *0* ou *1*.

8.4 Função de ativação: sigmoide

Uma das funções de ativação mais utilizadas é a sigmoide:

$$g(z) = \frac{1}{1+e^{-z}}$$
$$Y = h_\theta(x) = g(\Theta \times X) =$$
$$= \frac{1}{1+e^{-\Theta^T x}}$$

Figura 24

Código Latex:

```
g(z) = \frac{1}{1 + e^{-z}} \\
Y = h_\theta(x) = g(\Theta \times X) = \\
 = \frac{1}{1 + e^{-\Theta^T x}}
```

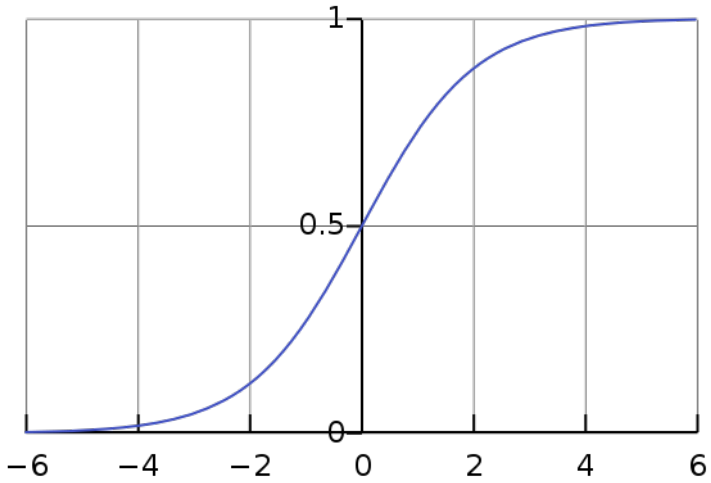


Figura 25. ***By Qef (talk) - Created from scratch with gnuplot, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4310325***

A sua principal característica é transformar os valores de *z* em valores binários, em função do quão extremo seja o seu valor, desde que seja contínua em todo o seu intervalo. Esta característica é muito interessante, já que nos permite derivá-la em todos os pontos, pelo que podemos utilizá-la com *gradient descent* para encontrar a inclinação, em cada iteração, e atualizar os coeficientes θ.

8.5 Função de custo

A função de custo é mais complexa, mas muito semelhante à função de custo da regressão linear.

$$J(\Theta) = -[\frac{1}{m}\sum_{i=0}^{m}(y^i log(h_\theta(x^i)) + (1-y^i)log(1-h_\theta(x^i))]$$

Figura 26

Código latex:

```
J(\Theta) = - [\frac{1}{m} \sum\limits_{i=0}^{m} (y^i log(h_\theta(x^i)) \\ + (1 - y^i) log(1 - h_\theta(x^i))]
```

Tal como a função de ativação sigmoide, será derivada em todo o seu intervalo.

Podemos interpretá-la como a soma entre dois termos diferentes, onde multiplicamos o logaritmo da hipótese (dentro da qual aplicaremos a função de ativação) num termo, pelo valor de *y* noutro termo por *(1 - y)*.

Estes dois termos distintos permitir-nos-ão somar o erro correspondente, consoante tenhamos acertado ou falhado a predição:

- Se *y* for 1, definido como a classe a classificar, somaremos ao erro o logaritmo da hipótese; se a hipótese for 1, o seu logaritmo será 0. Se não, somaremos o resultado ao erro ou "custo" total.
- Se *y* for 0, definido como a classe base, o primeiro termo será 0, mas não o segundo, já que *(1 - y)* será 1, logo somaremos o valor do segundo termo ao erro, consoante tenhamos acertado ou falhado a classe predita.
- Uma vez que *log(0)* não está definido, recordemos que o sigmoide apresenta duas assimptotas em valores de *0* e *1*, nunca chega a alcançar esses valores.

Por outro lado, devemos regularizar a função de custo da mesma forma que para a regressão linear, ou seja, incluindo o parâmetro de regularização λ:

$$J(\Theta) = -[\frac{1}{m}\sum_{i=0}^{m}(y^i log(h_\theta(x^i)) + (1-y^i)log(1-h_\theta(x^i))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2$$

Figura 27

Código Latex:

```
J(\Theta) = - [\frac{1}{m} \sum\limits_{i=0}^{m} (y^i log(h_\theta(x^i)) + (1 - y^i) log(1 - h_\theta(x^i))] \\
+ \frac{\lambda}{2m} \sum_{j=1}^{n} \Theta_j^2
```

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^i) - y^i) x_0^i$$

$$\theta_j := \theta_j - \alpha [\frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^i) - y^i) x_0^i + \frac{\lambda}{m} \theta_j];$$

Figura 28

Código Latex:

```
\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum\limits_{i=0}^{m} (h_\theta (x^i) - y^i) x_0^i \\
\theta_j := \theta_j - \alpha [\frac{1}{m} \sum\limits_{i=0}^{m} (h_\theta (x^i) - y^i) x_0^i + \frac{\lambda}{m} \theta_j]; \\
j \in [1, n]
```

8.6 Algoritmo de treino: classificação binária

1. Compilar os exemplos *X* e os seus resultados previamente conhecidos *Y*.
2. Codificar *Y* para *[0, 1]* segundo a classe.
3. Normalizar os exemplos *X*.
4. Dividir o conjunto de dados em subconjuntos de treino, validação e teste.
5. Escolher um rácio de aprendizagem α.
6. Inicializar os pesos Θ, de forma aleatória.
7. Iterativamente, calcular o custo, assim como as suas derivadas/inclinação, e atualizar Θ sobre o subconjunto de treino.
8. Finalizar quando Θ convergir num valor ótimo.
9. Modificar o rácio de aprendizagem α se necessário.
10. Obter a Θ ótima.
11. Otimizar λ iterativamente sobre o subconjunto de validação.
12. Obter a λ ótima.
13. Avaliar o modelo sobre o subconjunto de teste e obter o custo total final.
14. Usar a Θ e λ para formar o nosso modelo e realizar predições.

8.7 Algoritmo de treino: classificação multiclasse

1. Compilar os exemplos *X* e os seus resultados previamente conhecidos *Y*.
2. Normalizar os exemplos *X*.
3. Dividir o conjunto de dados em subconjuntos de treino, validação e teste.
4. Treinar um modelo binário por cada classe:

   4.1 Codificar *Y* para *[0, 1]* segundo a classe.

   4.2 Escolher um rácio de aprendizagem α.

   4.3 Inicializar os pesos Θ, de forma aleatória.

   4.4 Iterativamente, calcular o custo, assim como as suas derivadas/inclinação, e atualizar Θ sobre o subconjunto de treino.

   4.5 Finalizar quando Θ convergir num valor ótimo.

   4.6 Modificar o rácio de aprendizagem α se necessário.

   4.7 Obter a Θ ótima.

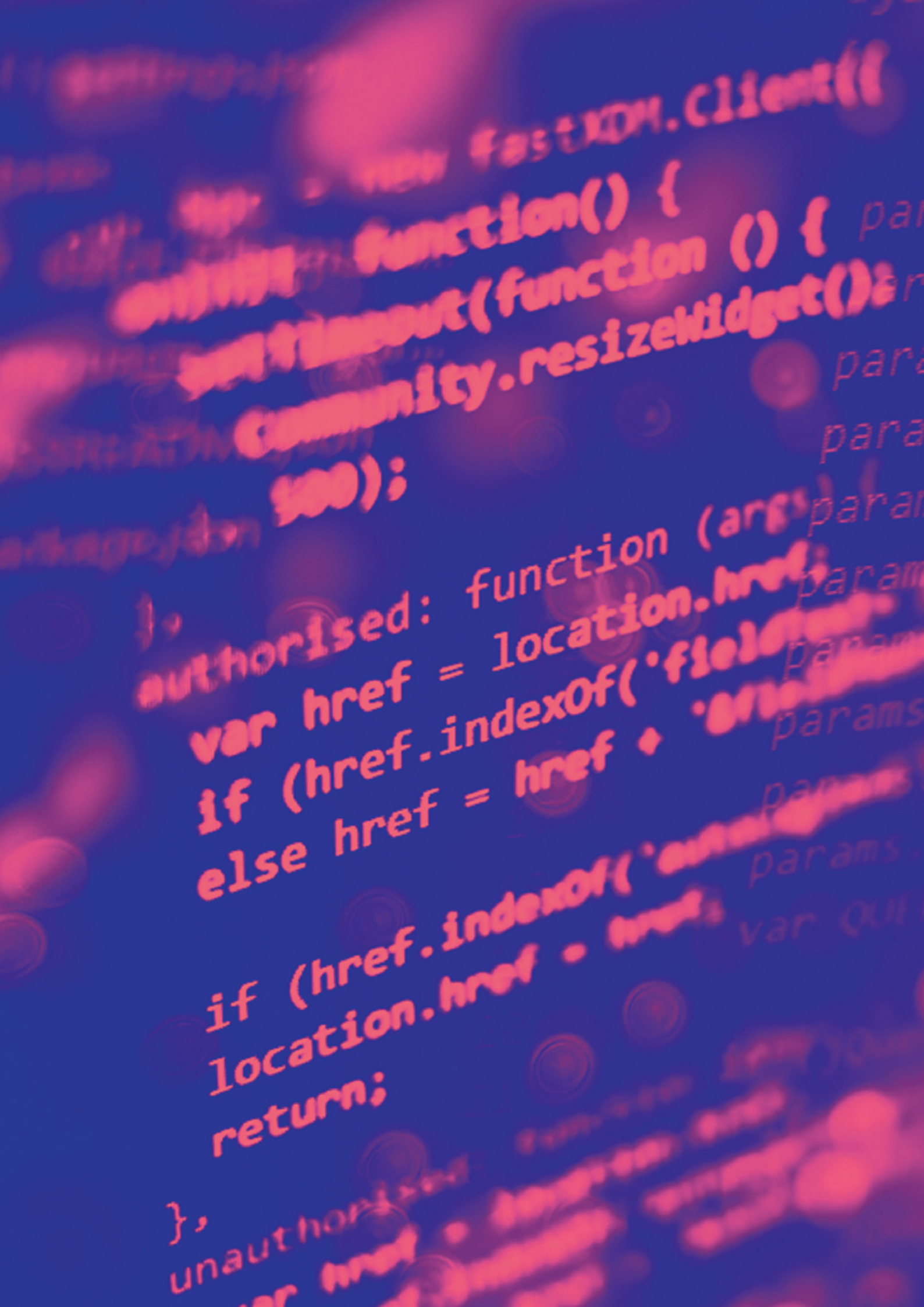
   4.8 Otimizar λ iterativamente sobre o subconjunto de validação.

   4.9 Obter a λ ótima.

   4.10 Avaliar o modelo sobre o subconjunto de teste e obter o custo total final.

   4.11 Obter a Θ *e* λ ótimas para o modelo.

5. Perante novos exemplos, predizemos usando o modelo para cada classe e ficaremos com a única classe predita, ou a que tenha o máximo valor pré-ativado.


function() {
community.resizeWidget();
authorised: function (
var href = location.href;
if (href.indexOf('
else href = href +
if (href.indexOf('
location.href =
return;
},
unauthorised: