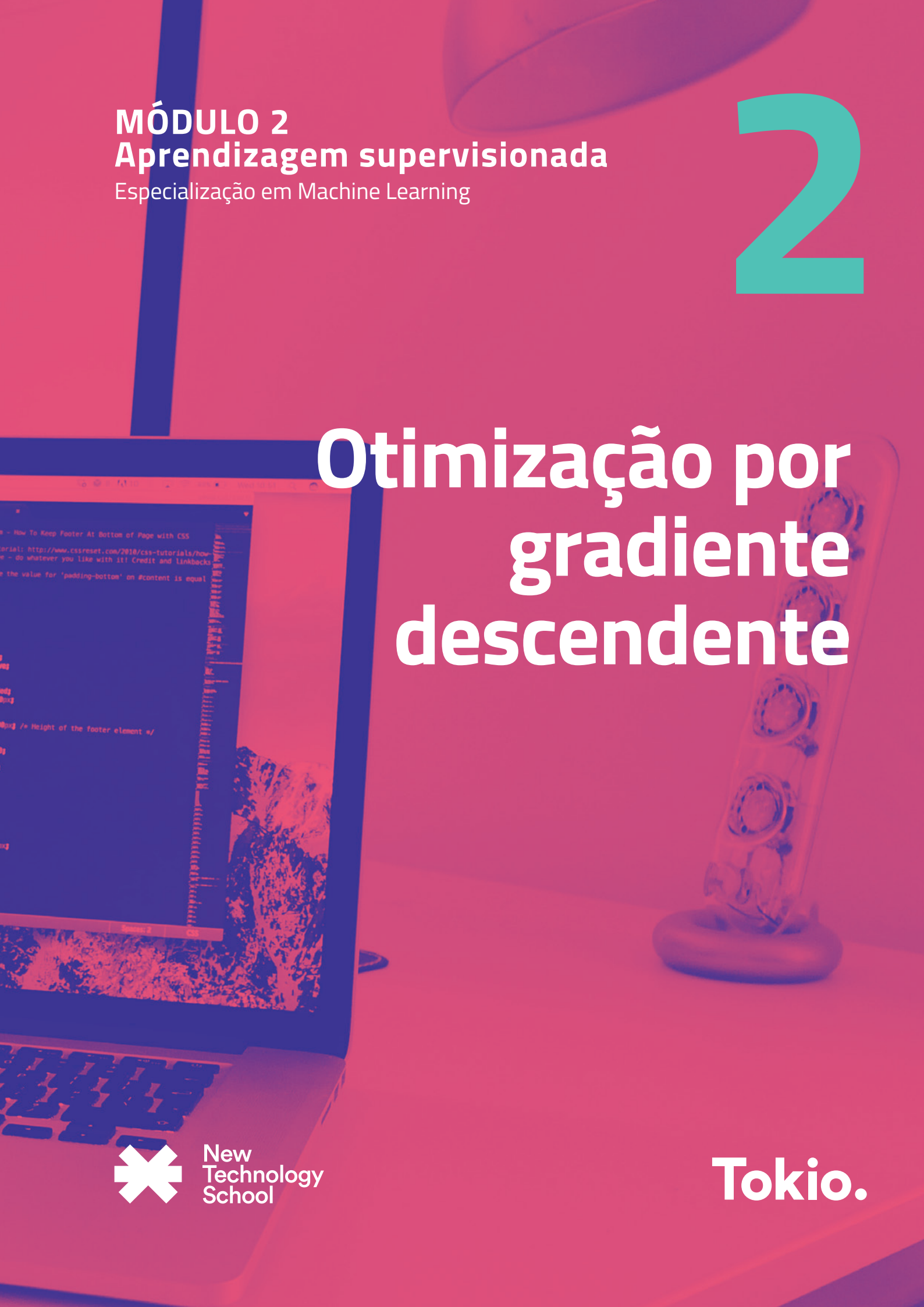
MÓDULO 2
Aprendizagem supervisionada
Especialização em Machine Learning

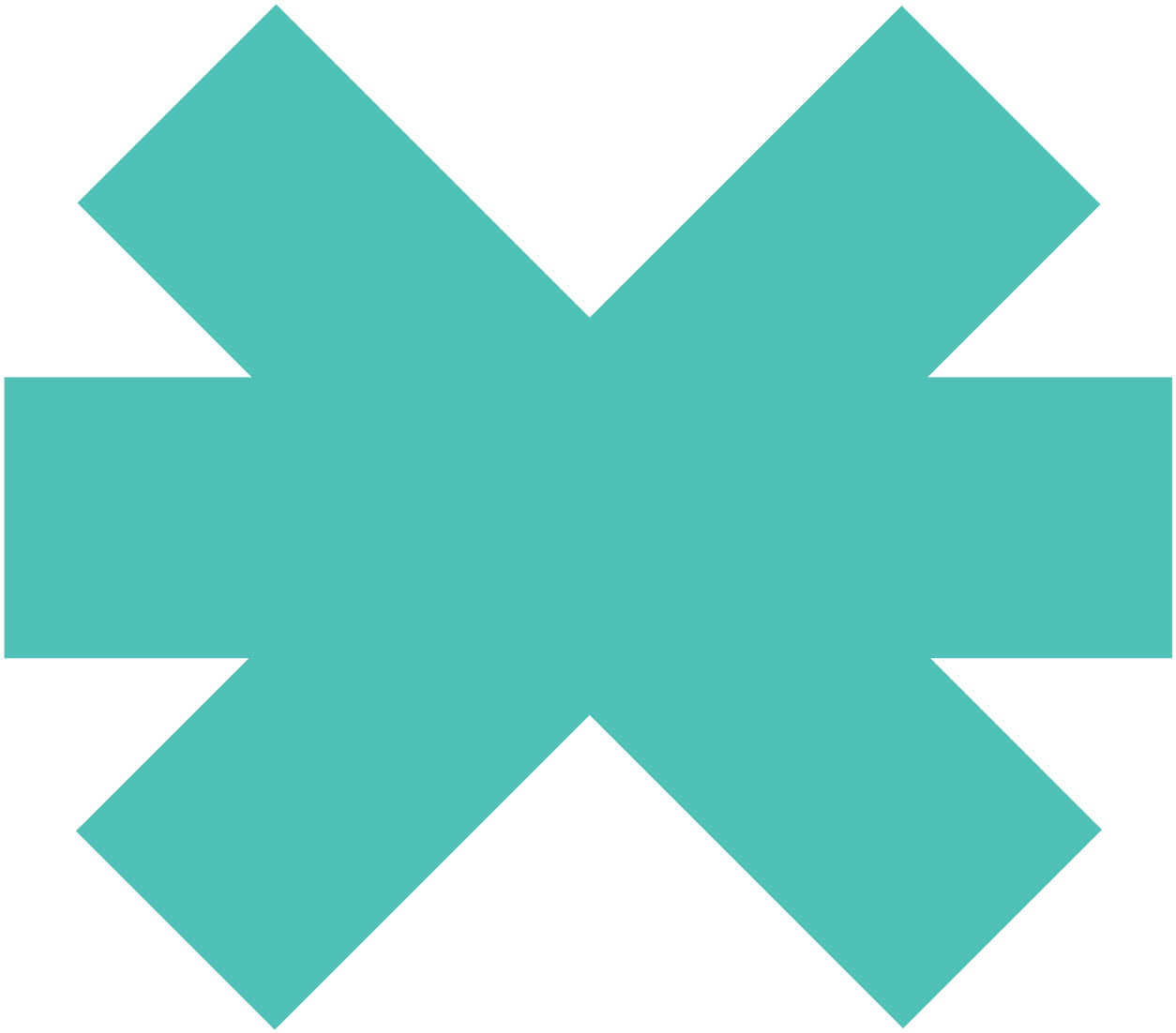
2

# Otimização por gradiente descendente



New Technology School

Tokio.



# 2 Otimização por gradiente descendente

## Sumário

# 2.1 Gradiente descendente

Um dos possíveis algoritmos de otimização matemática e, de longe, o mais utilizado em machine learning é o do gradiente descendente.

O gradiente descendente (ou *gradient descent*) simula que *J* é uma superfície multidimensional, a que podemos recorrer, com vales, montanhas e diferentes altitudes, representando o valor do custo ou erro total.

Imaginemos o seguinte gráfico 3D e pensemos que se refere a um mapa de altitudes:

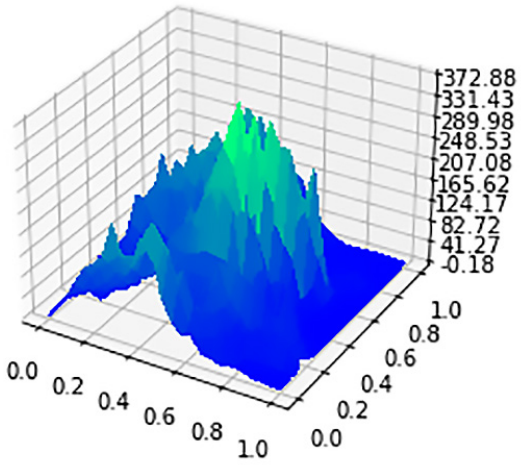



**Figura 8**

Neste gráfico:

- O **eixo *Y*** representa a altura nesse ponto do mapa ou o custo total nessas coordenadas.
- Os **eixos $X_1$ e $X_2$** representam a longitude e latitude no mapa, as suas coordenadas ou o valor dos coeficientes de $\theta_1$ e $\theta_2$.

Se pudermos determinar a inclinação em cada ponto, no qual encontremos a iteração, poderemos estabelecer a direção na qual a altura (ou erro) diminui com mais intensidade e segui-la, do mesmo modo que poderíamos seguir esse caminho, se quiséssemos encontrar o ponto mais baixo num terreno.

Logo, o método do *gradient descent* calcula a inclinação ou derivada total em cada ponto e, em função da sua direção amplitude/valor, atualiza os valores de $\Theta$ (todos os valores em simultâneo). O parâmetro $\alpha$, por sua vez, ajusta a velocidade ou a amplitude de cada passo, em função do valor da inclinação.

$$\theta_j := \theta_j - \alpha \frac{\partial J_{\theta_j}}{\partial \theta_j}$$

**Figura 9**

Código Latex:

```
\theta_j := \theta_j - \alpha \frac{\partial J_{\theta_j}}{\partial \theta_j}
```

## 2.2 Convergência

Paramos as iterações e temos como valor ótimo o valor atual de Θ quando o treino, ou caminho seguido pelo *gradient descent,* "converge".

"Convergir" significa que alcançou um ponto onde encontrou um valor mais ou menos estável, pelo que não continua a avançar, mas pode girar à sua volta e/ou os seus passos são já demasiado pequenos.

Podemos estabelecer vários limiares de amplitude de passos mínimos para determinar que convergiu; assim, por exemplo, usa-se, habitualmente $\varepsilon < 10^{-3}$.

Além disso, podemos encontrar inúmeros problemas, relativos à convergência de um modelo:

- O modelo não converge, mas diverge: quando parece que se aproxima de um valor ótimo e os seus passos são mais pequenos, "passa" e começa a "divergir" ou a dar passos em direções opostas ou de amplitude crescente.
- O modelo não encontra a direção ótima, confunde-se e opta por direções que não o levam ao ponto mais baixo.
- O modelo avança demasiado rápido e "passa" do ponto mínimo.
- O modelo avança demasiado lentamente e o treino é demasiado prolongado ou parece que continua a avançar constantemente.
- O modelo converge num valor, mas é subótimo, isto é, não é o valor mais ótimo possível.

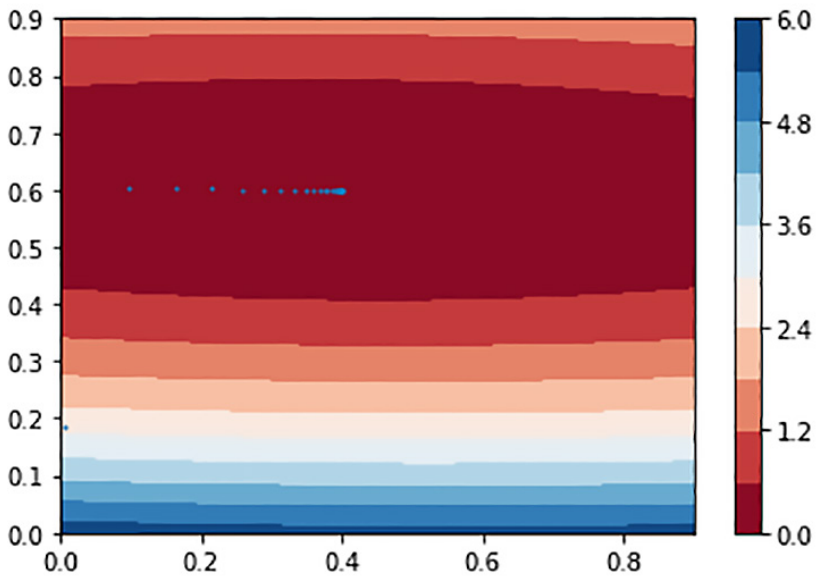



**Figura 10**

## 2.3 Mínimos locais e globais

Numa superfície completamente convexa onde há uma única "cratera" e todos os pontos estão inclinados até ao mesmo ponto mínimo central, encontrar o ponto ótimo ou mínimo global (de altura mínima) é trivial.

Contudo, se a superfície dos valores possíveis de $\Theta$ não for perfeitamente convexa, podem surgir vários mínimos locais, juntamente com um valor mínimo global.

Consoante o local onde iniciemos a nossa posição e os valores de $\Theta$ iniciais, a inclinação de que partimos pode levar-nos até um mínimo local, em vez de a um mínimo global.

Para o evitar, ao longo deste curso, iremos propor várias soluções.

# 2.4 Rácio de aprendizagem

O fator *alpha* da atualização de valores de Θ controla o rácio de avanço ou "aprendizagem" do modelo.

$$\theta_j := \theta_j - \alpha \frac{\partial J_{\theta_j}}{\partial \theta_j}$$

**Figura 9**

Código Latex:

\theta_j := \theta_j - \alpha \frac{\partial J_{\theta_j}}{\partial \theta_j}

Se este valor for muito elevado, significa que daremos passos maiores sobre a mesma inclinação. Com passos muito grandes, o nosso modelo pode não alcançar a conversão com o ponto mínimo e os seus passos mais pequenos podem ser demasiado grandes, e em vez de ficarem no fundo podem começar a divergir e a subir novamente pelas paredes.

Contudo, se este valor for muito baixo, a aprendizagem do modelo avançará mais lentamente e necessitaremos de muito mais tempo para o treinar.

Como podemos escolher o valor do rácio de aprendizagem ótimo?

## Escolha do rácio de aprendizagem

Para escolher um rácio de aprendizagem, recomendam-se os valores seguintes:

- $\alpha \in [0.001, 0.003, 0.01, 0.03, ..., 1]$

Estes valores apresentam um rácio de, aproximadamente, x3 entre si.

Conselhos:

- Escolher um valor de $\alpha$ inicial de entre os propostos.
- Vigiar a evolução da função de custo, representada num gráfico em relação ao n.º de iterações.
- Modificar o valor de $\alpha$ e repetir para experimentar, se for necessário.

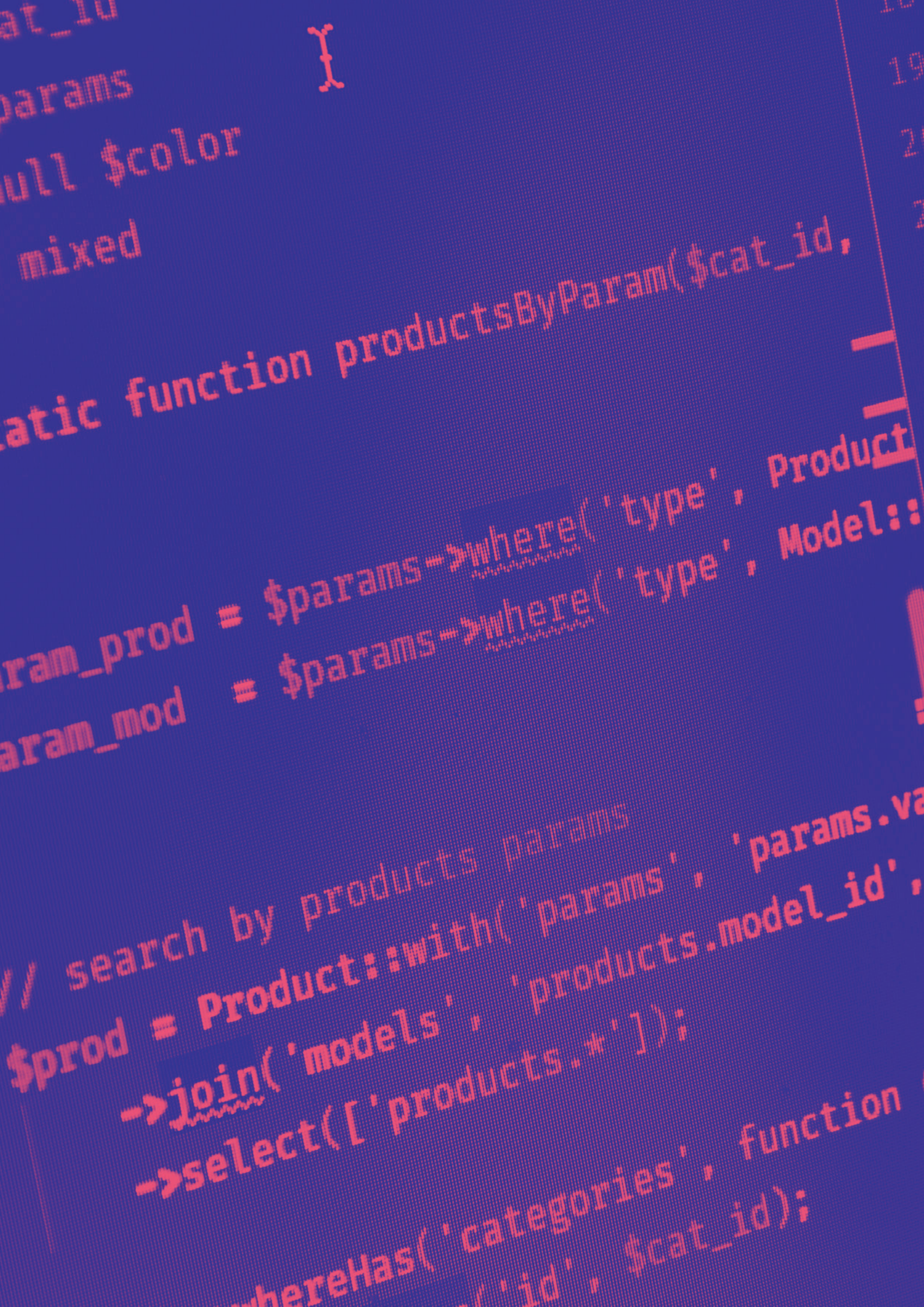
Há outros métodos mais avançados para escolher um valor do rácio de aprendizagem ou dispor de um rácio modificável entre iterações:

- Modificar o rácio de aprendizagem. Com um rácio de aprendizagem variável, podemos conseguir que os passos finais sejam mais pequenos ainda ou detetar o momento no qual podemos estar a divergir, não alcançando o ponto mínimo e reduzindo o rácio.
- Adicionar inércia à descida. Se estamos a avançar numa solução, poderíamos claramente saltar alguns montes intermédios e não ficarmos presos na primeira elevação ou terreno.

## 2.5 Algoritmo de treino

O algoritmo de treino completo de um problema de regressão linear múltipla poderia ser:

1. Compilar os exemplos *X* e os seus resultados previamente conhecidos *Y*.
2. Escolher um rácio de aprendizagem $\alpha$.
3. Inicializar os pesos $\Theta$, de forma aleatória.
4. Iterativamente, calcular o custo, assim como as suas derivadas/inclinações, e atualizar $\Theta$.
5. Finalizar quando $\Theta$ convirja num valor ótimo.
6. Modificar o rácio de aprendizagem $\alpha$ se necessário.
7. Obter a $\Theta$ ótima e realizar predições com ela.


at_id
params
ull $color
mixed
atic function productsByParam($cat_id,
ram_prod = $params->where('type', Produc
aram_mod = $params->where('type', Model::
// search by products params
$prod = Product::with('params', 'params.va
->join('models', 'products.model_id',
->select(['products.*']);
whereHas('categories', function
'id', $cat_id);