Tema 02. Regresión y clasificación

Autor: Ismael Sagredo Olivenza

2.3. Linear Regression with Multiple Variables

- So far we have seen regression for one variable
- But this type of problem usually requires more inputs to model the problem.

Size in feet^2 x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) in 1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

2.3.1 How do we generalise the equation?

One variable $f(w, b) = w * x + b$ generalise to multiple variables

$$f(w, b) = w_1 * x_1 + w_2 * x_2 + \ldots w_n x_n + b$$

Previous example, row 1:

X_1 = Size in feet
X_2 = Number of bedrooms
X_3 = Number of floors
X_4 = Age of home

$$f(w, b) = 0.1 * x_1 + 4 * x_2 + 10x_3 - 2x_4 + 80$$

2.3.1.1 Vectorization

$[w_1, w_2, \ldots w_n]$ are parameters of the model and **b** is a number

$$f_{\vec{w},b}(\vec{x}) = \vec{w} * \vec{x} + b$$

Some implementation details (Numpy)

```
w = np.array([1.0,2.5,-3.3])
b = 4
x = np.array([10,20,30])

f = np.dot(x,y)+b
## Without vectorization for j in range (0,n)
```

Without vectorization

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

t_0

```
f = f + w[0] * x[0]
```

t_1

```
f = f + w[1] * x[1]
```

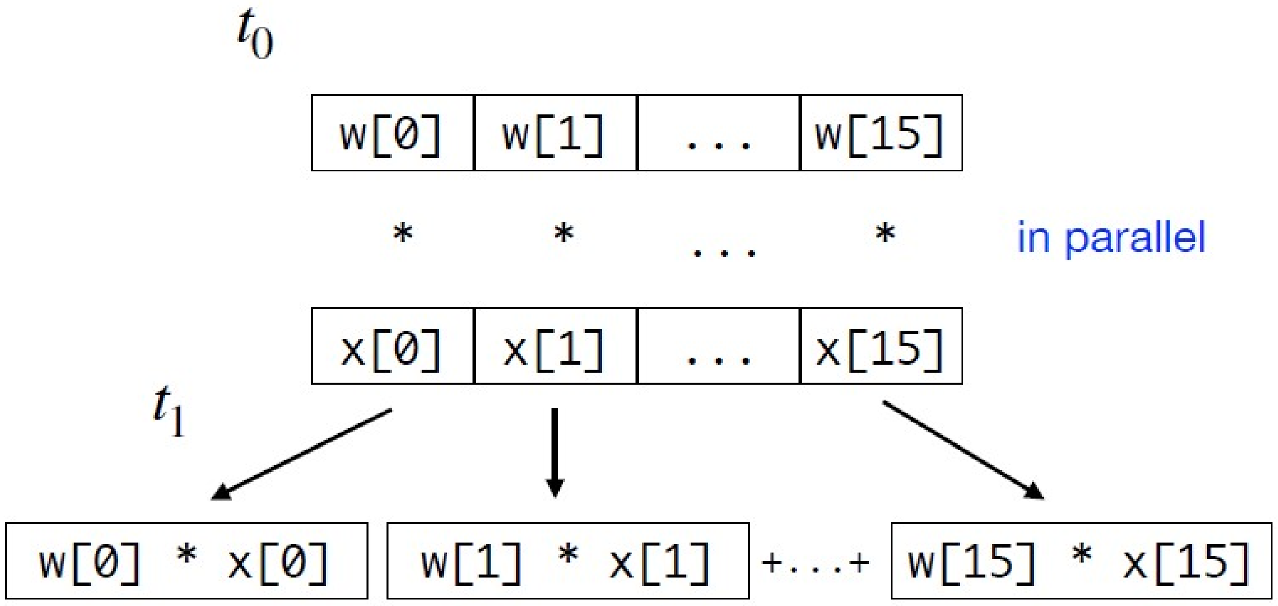
...

t_{15}

```
f = f + w[15] * x[15]
```

Vectorization

```
np.dot(w,x)
```



efficient → scale to large datasets

Numpy dot product it can use an optimized implementation obtained as part of "BLAS" (the Basic Linear Algebra Subroutines)

* **https://es.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms**

This library obtain adventage of hardware (if exist) such as instructions SIMD, cache, multicore, etc.

if possible, it is preferable for efficiency reasons to describe the operations in vector form

2.3.1.2 Gradient descent implementation details

Gradient descent

$\vec{w} = (w_1 \; w_2 \; \cdots \; w_{16})$ $\quad b$ parameters

derivatives $\vec{d} = (d_1 \; d_2 \; \cdots \; d_{16})$

```
w = np.array([0.5, 1.3, ... 3.4])
d = np.array([0.3, 0.2, ... 0.4])
```

learning rate α

compute $w_j = w_j - 0.1 d_j$ for $j = 1 \dots 16$

Without vectorization

$$w_1 = w_1 - 0.1 d_1$$
$$w_2 = w_2 - 0.1 d_2$$
$$\vdots$$
$$w_{16} = w_{16} - 0.1 d_{16}$$

```
for j in range(0,16):
    w[j] = w[j] - 0.1 * d[j]
```

With vectorization

$$\vec{w} = \vec{w} - 0.1 \, \vec{d}$$

```
w = w - 0.1 * d
```

2.3.1.3 Gradient descent equations

$$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (y\prime - y_i) x_{j,i} \forall j = 1..n$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (y_i\prime - y_i)$$

$$\vec{y}\prime = F_{\vec{w},b}(\vec{x})$$

repeat for all n
$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (y_1\prime - y_i) x_{1,i}$ ...
$w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (y_n\prime - y_n) x_{n,i}$

2.3.2 Normal equation

- An alternative to gradient descent
- Only for linear regression, solve for w, b without iterations
- Doesn’t generalize to other learning algorithms
- Slow when number of features is large
- Gradient descent is the recommended method for finding parameters but Normal equation method may be used in some libraries

Normal equation

Examples: $m = 4.$

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

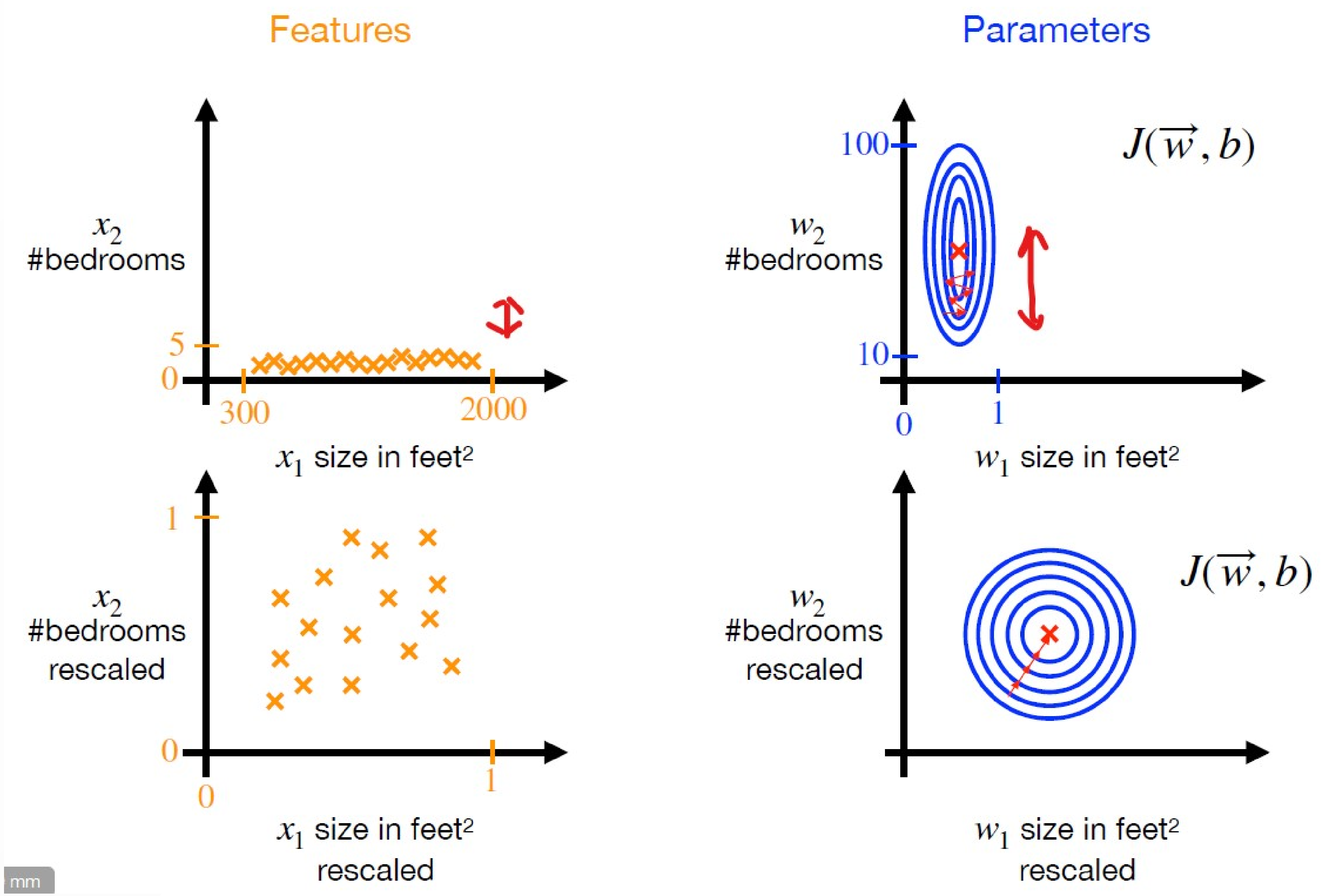
$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y$$

2.3.5 Feature Scaling

- Features must be in same scale.
- What should be the scale of the parameters W?
- As the size of the error modifies the parameters using gradient descent, having different scales makes some parameters fit better than others.
- By rescaling all values to a similar scale, the parameters are modified with the same proportion.
- Too large or too small jumps (as with the learning rate) will make it difficult to find the correct weight that minimises the function.

Feature size and gradient descent



In the previous example, the domine of X_1 range from 300 to 2000 => 1700 and x_2 is from 0 to 5 (size 5)

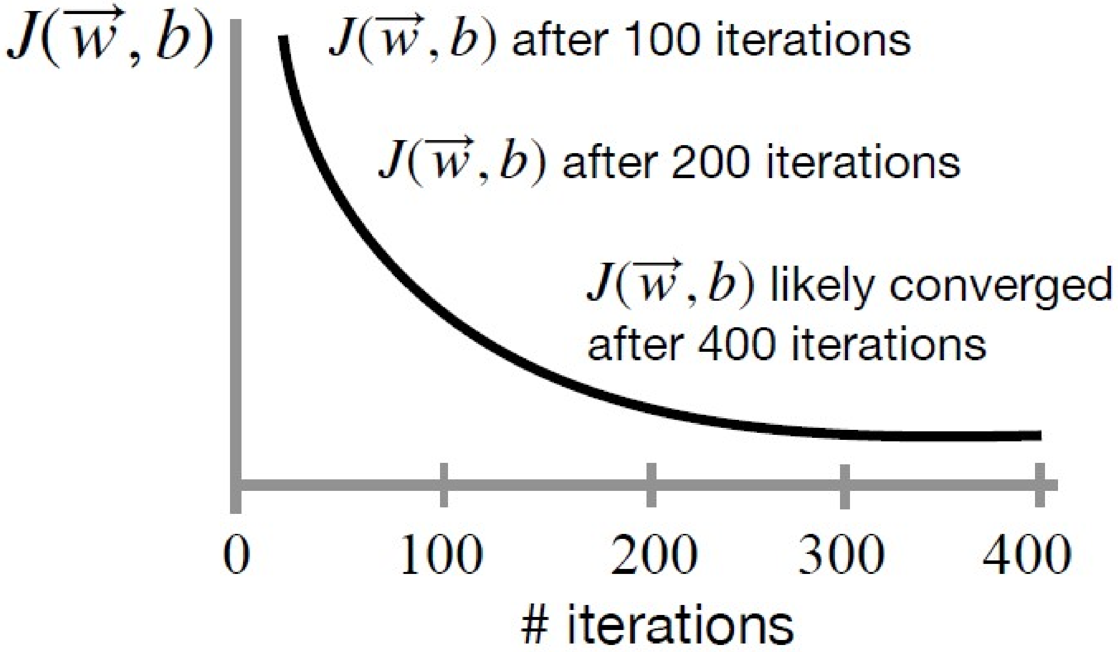
- We can transform date dividing by max (if we can adjust the min to 0, we can substract min/max). What we are looking for is that both variables move between the values 0..1.
- Also we can normalized substrating the mean and dividing by range (or MAX-MIN) which keeps the values between -1 and 1
- Finally we can normalize using Z-Score (substrating the mean and dividing by standar desviatión). Using z-Score, all features have mean 0 and desviation 1

2.3.5 How to establish the convergence criterion?

We can stablishing different convergence criteria, most common:

- That the error is less than a given one
- No significant improvement in model perdormance diriing n iterations of gradient descent

Make sure gradient descent is working correctly

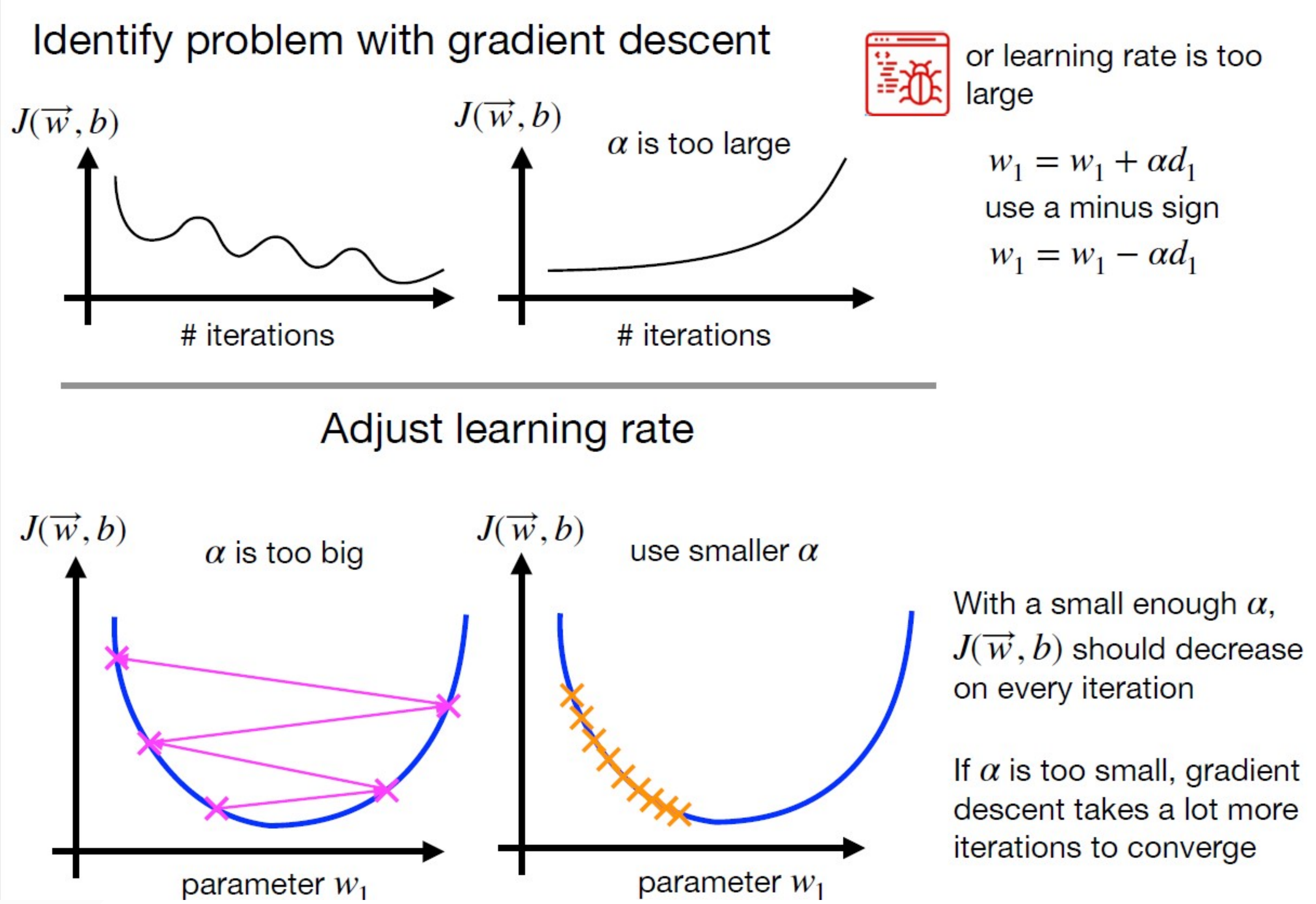


Automatic convergence test
Let ϵ "epsilon" be 10^{-3}

If $J(\vec{w}, b)$ decreases by $\leq \epsilon$ in one iteration, declare convergence.

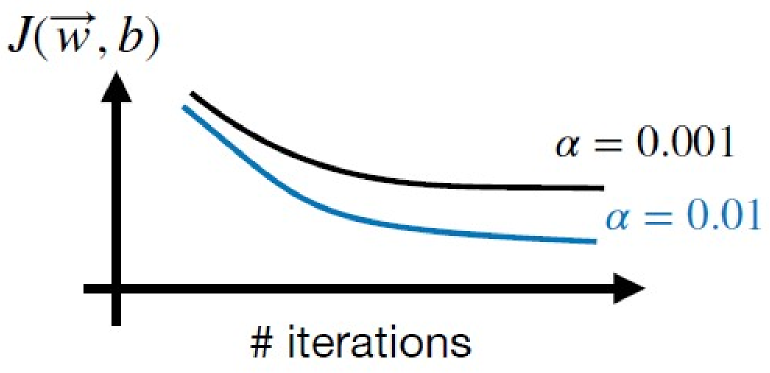
(found parameters $\vec{w}, b$ to get close to global minimum)

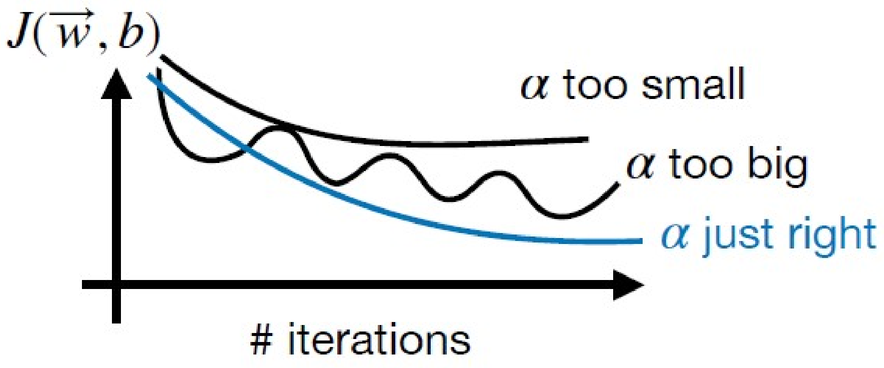
2.3.6 Convergence and choosing learning rate



Values of α to try:

... 0.001 0.003 0.01 0.03 0.1 0.3 1...





2.3.7 Polinomial regression

You can adjust the linear regressión to a poligon vector but...
Which one is the best fit?