Linear & logistic regression

1. Introduction

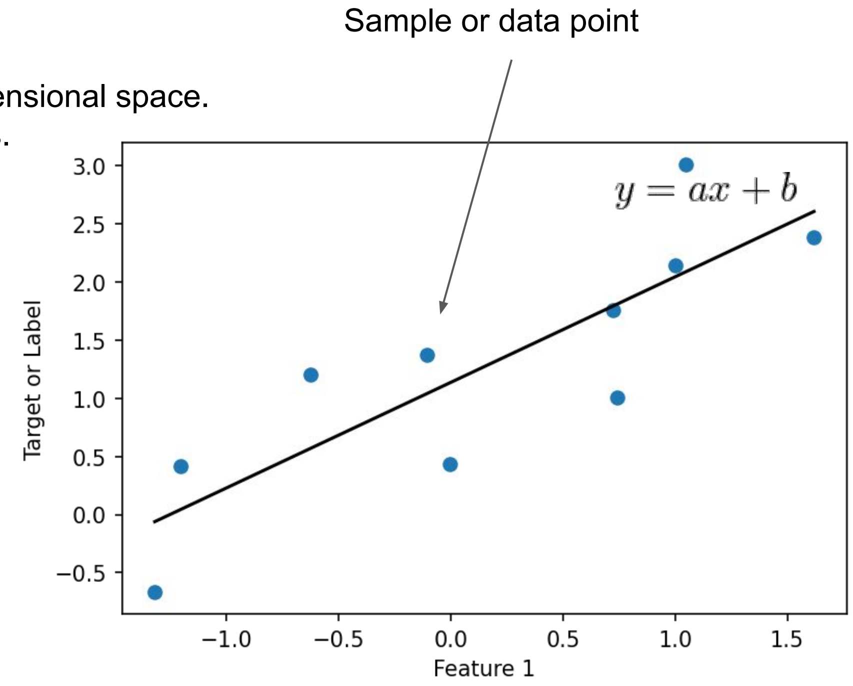
- Supervised machine learning:
    - Dataset with samples and associated labels
    - The goal in supervised machine learning is making predictions for new, previously unseen, test inputs
    - The task can be classification or regression
        - classification: discrete labels (a finite number of classes)
        - regression: continuous labels (value to guess for each sample)
    - Regression: fit a curve to the data

Linear and logistic regression: simple but

- powerful
- contain the most important concepts in machine learning

Linear regression

- Feature(s): value(s) describing the sample.
- Feature vector: vector of N real values. coordinate in a N dimensional space.
- Target or label: real value we want to predict from the features.

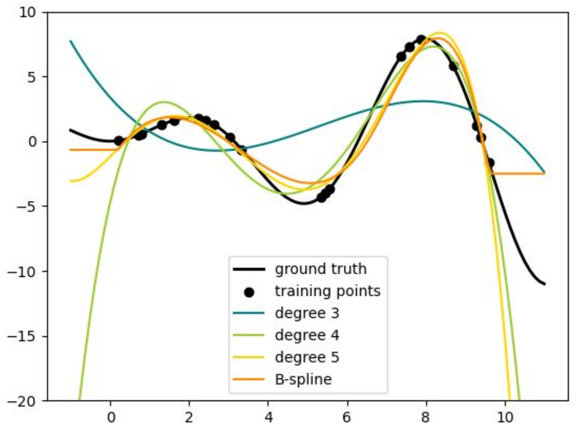


- Linear regression assumes linear relationship between labels and features
- Linear regression: find coefficient a and intercept b.

Label y can be predicted for a new datapoint x when a and b are known →generalization → **relationship learned!**

Regression beyond linear

Polynomial fit



$$y = \sum_{k=0}^{K} a_k x^k$$

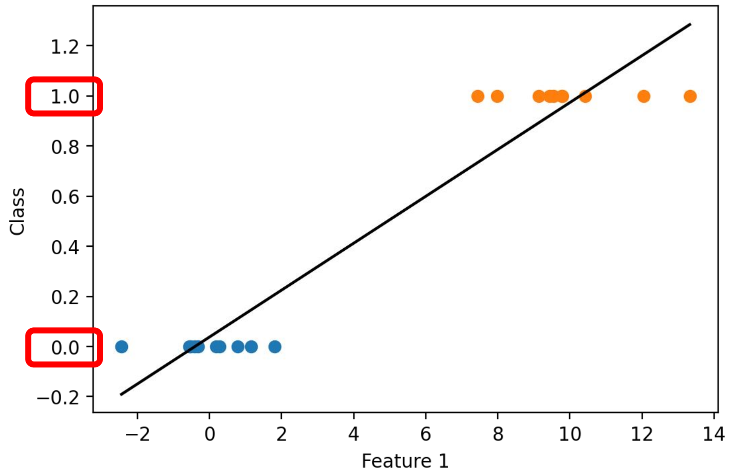
Stone–Weierstrass theorem:
Any continuous function on a bounded interval can be approximated as closely as desired by a polynomial

- Difficult to choose K
- choose the K with smallest error on the training data
  $\longrightarrow$ risk of overfitting

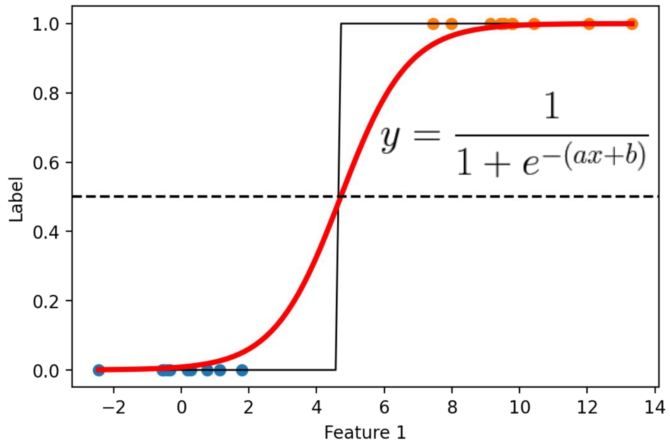
From sklearn “polynomial interpolation”
https://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html

Regression and categorical data

Label are discrete: data points belong to different classes



Linear regression is not adapted



$$y = \frac{1}{1 + e^{-(ax+b)}}$$

Logistic regression : fit a logistic function, find a and b

Logistic regression: learning for categorical data

Multidimensional data

Usually we have samples with several features $x \in \mathbb{R}^N$ $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$

$$y = a_1 x_1 + a_2 x_2 + \cdots + a_N x_N + b$$

$$y = a^T x + b$$

$$a^T = (a_1, a_2, \cdots, a_N)$$

Sometimes more compactly:

$$y = a^T x$$

$$a^T = (a_1, a_2, \cdots, a_N, b) \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ 1 \end{pmatrix}$$

We can also have $y = x^T a$

Error and Loss function

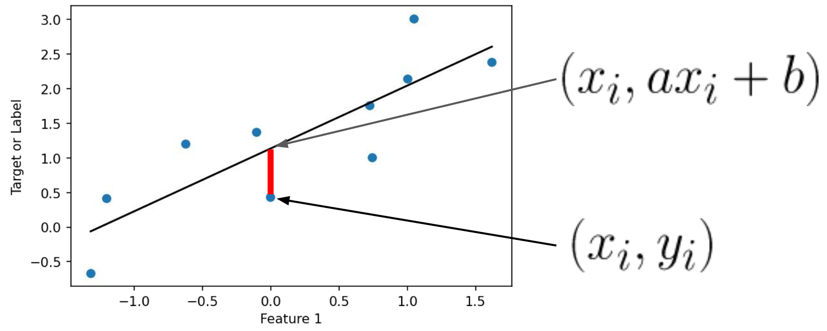
Learning is reducing the error between the prediction and the correct answer

Learning is minimizing a loss function

$$E = \sum_i (y_i - (ax_i + b))^2$$

For linear regression, we want to minimize this

Minimize with respect to *a* and *b*



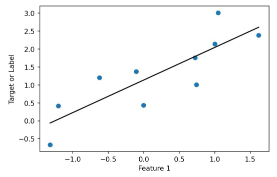
Argument minimum “argmin”: values of the variables for which the function is minimum

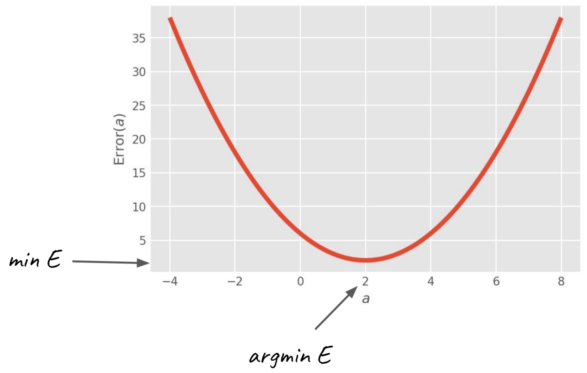
$$(a^*, b^*) = \underset{a,b}{\operatorname{argmin}}\ E(a, b)$$

We want the best parameters (a*, b*)

Best solution (linear regression):

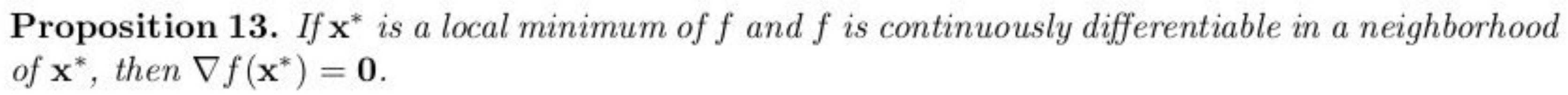
$$y = a^*x + b^*$$



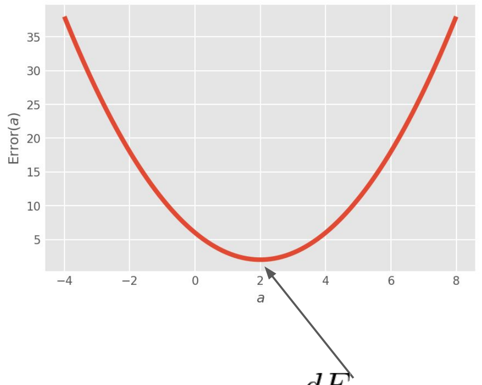


Minimizing the error

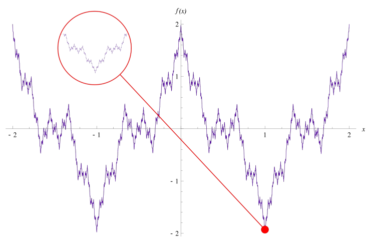
- The error is a function of some parameters E(a,b)
- The extrema of a function f are found where the derivative of f vanishes (see Fermat's theorem on stationary points + assume f is continuously differentiable)



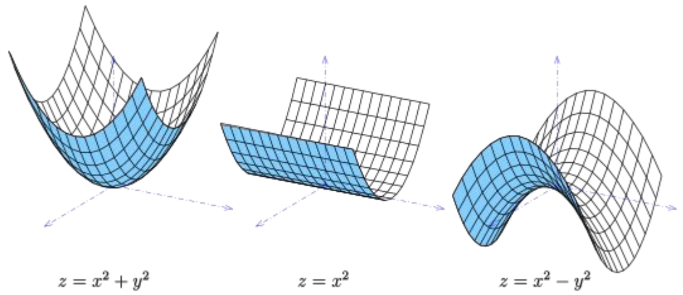
Proposition 13. *If* $\mathbf{x}^*$ *is a local minimum of* f *and* f *is continuously differentiable in a neighborhood of* $\mathbf{x}^*$*, then* $\nabla f(\mathbf{x}^*) = \mathbf{0}$.



$$\frac{dE}{da}(a_0) = 0$$



Weierstrass function, continuous but not differentiable

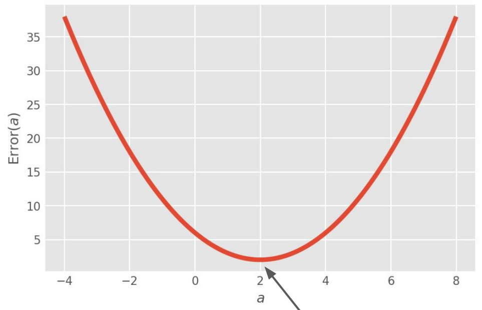


Different cases where the derivative vanishes

Image: Ag2gaeh, CC BY-SA 4.0, via Wikimedia Commons

There are different ways for finding the minimum

- Finding the minimum by solving directly “derivative = 0” (linear regression), for simple loss functions
- Finding the minimum iteratively
  - with gradient descent (logistic regression and most of the ML methods)
  - with the gradient but tailored for specific cases (ex: conjugate gradient)
  - with the second derivative (ex: newton’s method) converge faster but more difficult to compute



$$\frac{dE}{da}(a_0) = 0$$

Different ways of “learning”

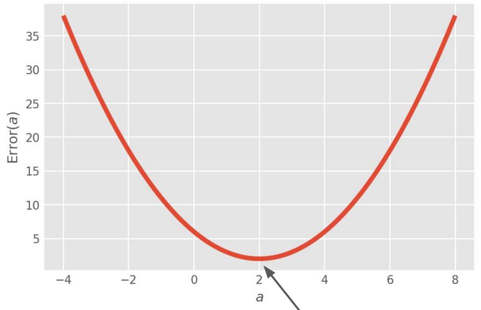
2. Linear regression

2.1 Linear regression loss

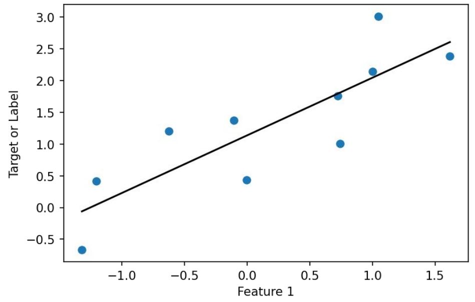
$$E = \sum_i (y_i - (ax_i + b))^2$$

Properties of the loss function:

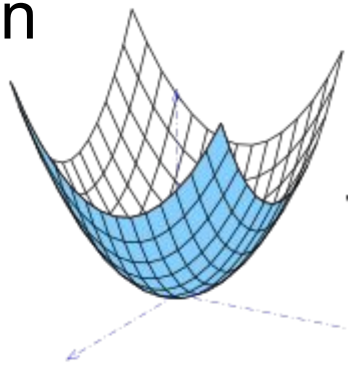
- E is continuously differentiable w.r.t. a and b
- E is convex (Euclidean norm is convex)



$$\frac{dE}{da}(a_0) = 0$$



2.2 Minimizing the loss for linear regression



At the minimum:

$$\frac{\partial E}{\partial a}(a^*, b^*) = 0 \qquad \frac{\partial E}{\partial b}(a^*, b^*) = 0$$

2 equations, 2 unknowns

In general in N dimensions:

$$E = \sum_i (y_i - (a_1 x_{1i} + a_2 x_{2i} + \cdots + a_N x_{Ni} + b))^2$$

$$\frac{\partial E}{\partial a_k}(a_1^*, a_2^*, \cdots, a_N^*, b^*) = 0 \quad \forall k$$

$$E = \sum_i (y_i - a^T x_i)^2$$

or

$$E = \sum_i (y_i - x_i^T a)^2$$

with

$$a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ b \end{pmatrix}$$

Matrix form

we have several samples, with several features (several dimensions)

Let us write it in matrix form

$$X = \begin{array}{cc} \xleftrightarrow{\text{Features}} & \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} & 1 \\ x_{21} & x_{22} & & & 1 \\ \vdots & & \ddots & & 1 \\ x_{M1} & & & x_{MN} & 1 \end{pmatrix} & \updownarrow \text{Samples} \end{array}$$

$$E = \|y - Xa\|^2$$

Euclidean norm

Minimum with respect to a ?

Norms are convex (see the math for ML book), we have a global minimum for linear regression

Vectors and derivatives

$$x = (x_1, x_2, \cdots, x_N)^T$$

Vector of derivatives
Gradient

$$\frac{\partial}{\partial x} = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_N})^T$$

Example: the 2-norm (Euclidean norm)

$$\|x\|_2^2 = (x, x) = \sum_i x_i^2$$

$$\frac{\partial}{\partial x}\|x\|_2^2 = 2x \longleftarrow$$

Exercise!

Gradient and Jacobian

$$\frac{\partial}{\partial x} = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_N})^T$$

Note: the symbol ∇ (nabla) is used to denote the gradient. $\nabla = \frac{\partial}{\partial x}$

If the gradient is applied to a real-valued function: vector of derivatives
If the gradient is applied to a vector-valued function: matrix of derivatives, Jacobian

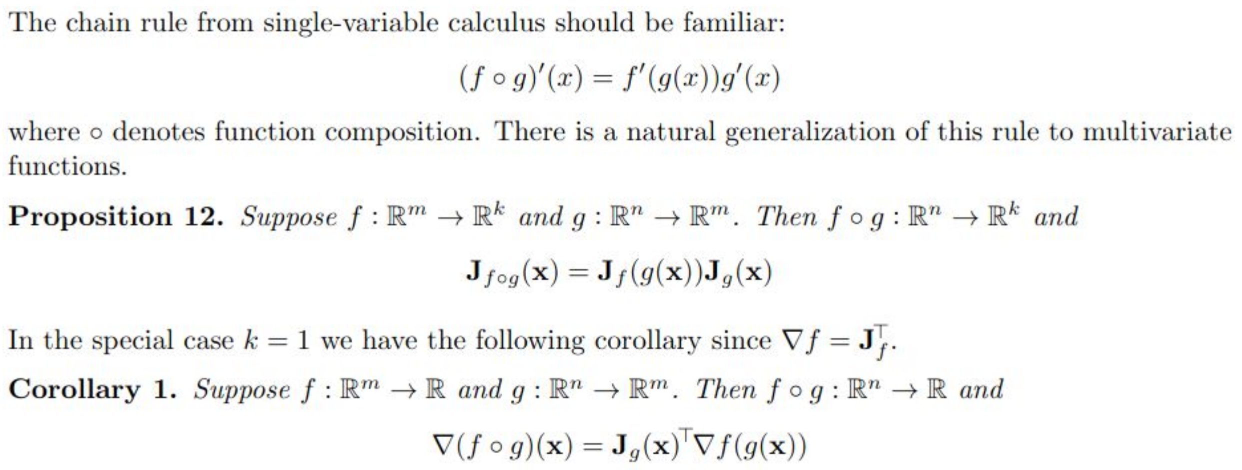
$$f : \mathbb{R}^N \to \mathbb{R}$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix}$$

Jacobian for $f : \mathbb{R}^N \to \mathbb{R}^M$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \\ \frac{\partial f_M}{\partial x_1} & & \frac{\partial f_M}{\partial x_N} \end{pmatrix}$$

The chain rule (Math for ML or matrix cookbook)



The chain rule from single-variable calculus should be familiar:

$$(f \circ g)'(x) = f'(g(x))g'(x)$$

where $\circ$ denotes function composition. There is a natural generalization of this rule to multivariate functions.

Proposition 12. *Suppose $f : \mathbb{R}^m \to \mathbb{R}^k$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. Then $f \circ g : \mathbb{R}^n \to \mathbb{R}^k$ and*

$$\mathbf{J}_{f \circ g}(\mathbf{x}) = \mathbf{J}_f(g(\mathbf{x}))\mathbf{J}_g(\mathbf{x})$$

In the special case $k = 1$ we have the following corollary since $\nabla f = \mathbf{J}_f^\top$.

Corollary 1. *Suppose $f : \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. Then $f \circ g : \mathbb{R}^n \to \mathbb{R}$ and*

$$\nabla(f \circ g)(\mathbf{x}) = \mathbf{J}_g(\mathbf{x})^\top \nabla f(g(\mathbf{x}))$$

Jacobian for $f : \mathbb{R}^N \to \mathbb{R}^M$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \\ \frac{\partial f_M}{\partial x_1} & & \frac{\partial f_M}{\partial x_N} \end{pmatrix}$$

- **Note 1:** in our case, the variable is **a** the linear regression vector of parameters
- **Note 2:** for linear regression we are in the case of Corollary 1

$$f(x) = \|x\|^2 \qquad g(a) = y - Xa \qquad E(a) = (f \circ g)(a) = \|y - Xa\|^2$$

- **Note 3:** the chain rule is used heavily in deep learning

Derivative of the loss function

$f(x) = \|x\|^2$ $\quad g(a) = y - Xa$ $\quad E(a) = (f \circ g)(a) = \|y - Xa\|^2$

- Derivatives:

$$\frac{\partial}{\partial x}\|x\|_2^2 = 2x \qquad J_g = -X$$

$$\nabla(f \circ g)(a) = -X^T 2(y - Xa)$$

Other way to compute the derivative (exercise): start with

$$\|y - Xa\|^2 = (y - Xa)^T (y - Xa)$$

- Minimum at zero E(a*)=0:

$$X^T y - X^T X a^* = 0 \Leftrightarrow a^* = (X^T X)^{-1} X^T y$$

If we can invert X^TX, (the columns must be linearly independent) !
we need as many equations as we have unknown (in one dimension, a and b: we need at least 2 points)

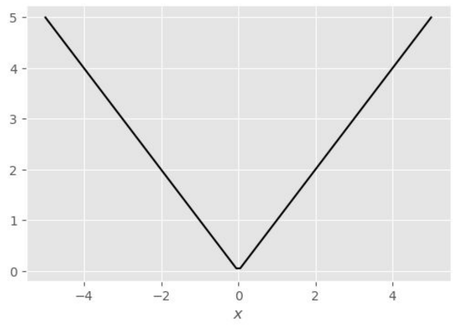
Why this loss function?

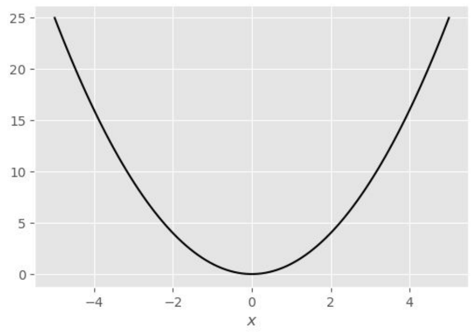
- Intuitive,
- simple derivative,
- analytic expression of the solution.

$$E = \sum_i (y_i - x_i^T a)^2$$

Could it be simpler?

- not the absolute value: not differentiable at 0.





2.3 Ridge regression

Regularization of the solution, we do not want the coefficients of a to be too big

$$E = \|y - Xa\|^2 + \lambda\|a\|^2$$

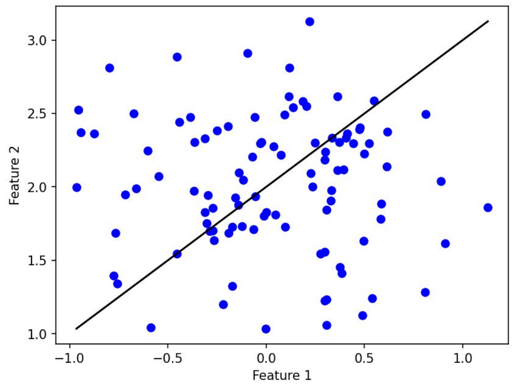
Answer (exercise):

$$a^* = (X^T X + \lambda I)^{-1} X^T y$$

It is also a way to solve a system with less points than unknowns
If there is several solution, choose the one where the coefficients a_i are small

2.4 Evaluation of the model

The R-squared score



Error between true values y and predicted ones ŷ

$$E = \sum_i (y_i - \hat{y}_i)^2$$

(loss function)

prediction of the model

Error between true values y and a mean prediction

$$E_m = \sum_i (y_i - \bar{y})^2$$

mean value of y

The R² score

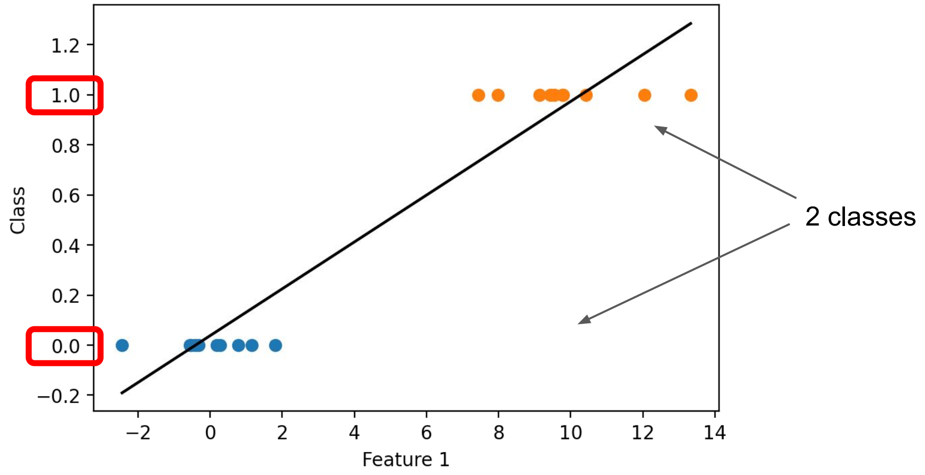
$$R^2 = 1 - \frac{E}{E_m}$$

- best value is 1 (E=0)
- value 0: as good as an average prediction
- negative value: worse that an average prediction

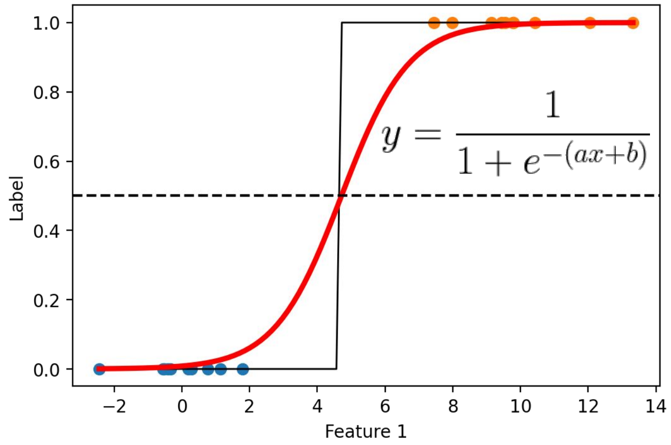
3. Logistic regression

3.1 Logistic regression, introduction

- Supervised learning
- Classification (regression used for classification)



Labels take 2 values
Linear regression is not adapted

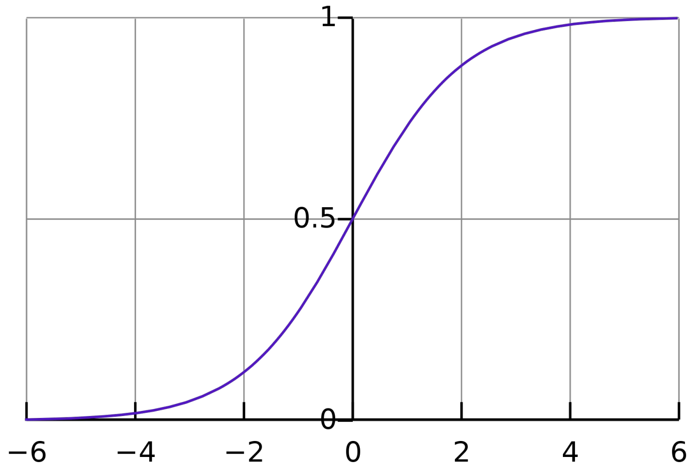


Logistic regression : fit a logistic function, find a and b

Logistic function: values from 0 to 1.

Logistic function

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$



Derivative

$$\frac{\mathrm{d}}{\mathrm{d}x}f(x) = \frac{e^x}{(1+e^x)^2} = f(x)\big(1-f(x)\big)$$

Bounded between 0 and 1, with a sharp transition.

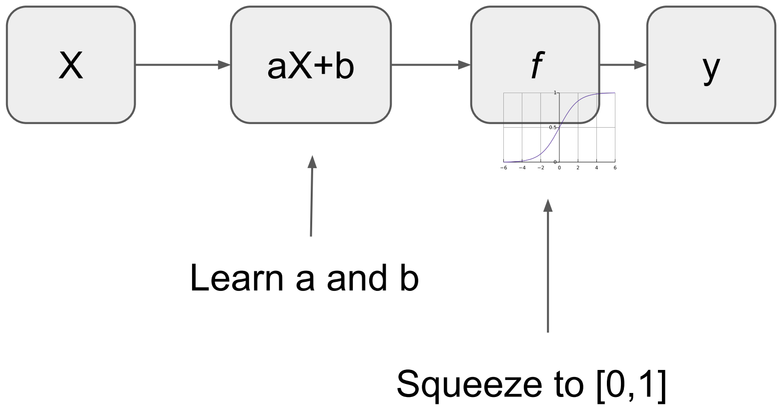
Integral

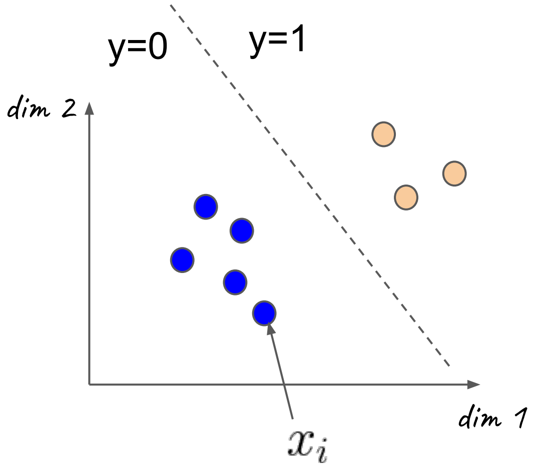
$$\int \frac{e^x}{1+e^x}\,dx = \int \frac{1}{u}\,du = \ln u = \ln(1+e^x).$$

3.2 A first intuition for why the logistic function

“Compress” the linear regression in the range [0,1]

$$\hat{y}(x) = \frac{1}{1 + e^{-(ax+b)}}$$





$$\hat{y}_i = f(a_1 x_i[1] + a_2 x_i[2] + b)$$

3.3 Explanation from a statistical point of view

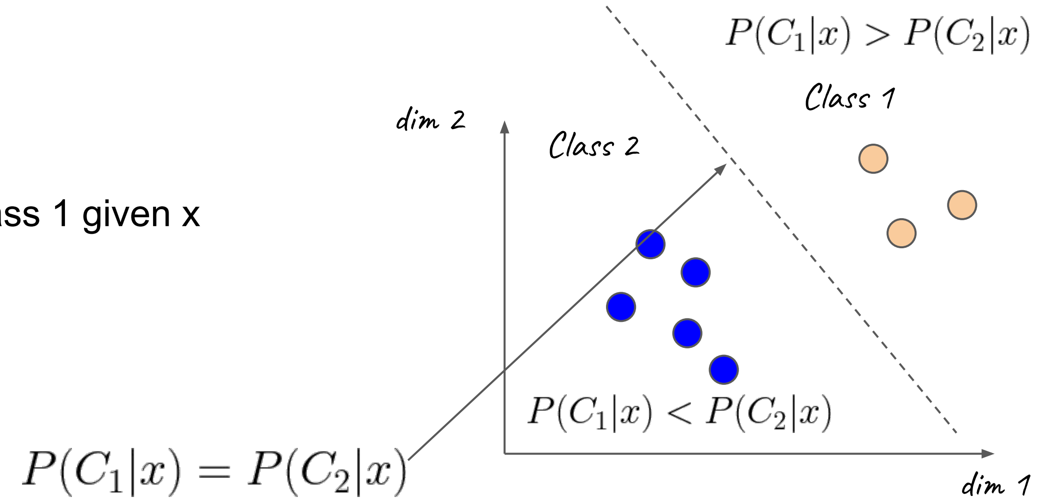
2 classes, we choose:

- sample in class 1: y = 1
- sample in class 2: y = 0

$y = P(C_1|x)$ Probability to be in class 1 given x

$$P(C_2|x) = 1 - P(C_1|x)$$

The label becomes a probability



or $\frac{P(C_1|x)}{1 - P(C_1|x)} = 1$ or $\ln \frac{P(C_1|x)}{1 - P(C_1|x)} = 0$

If we choose:

$\ln \frac{P(C_1|x)}{1 - P(C_1|x)} = a^T x + b$ the solution is $y = f(a^T x + b)$ ok, but why this choice?

With a Gaussian assumption

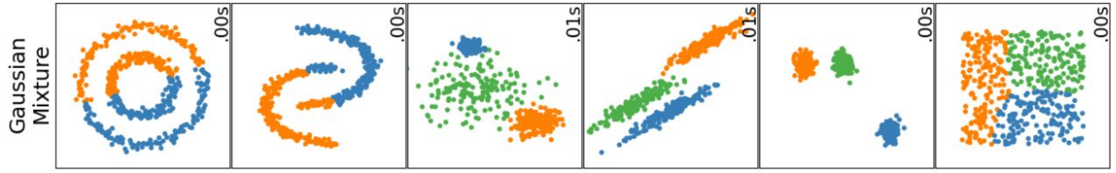
$$P(C_1|x) = ce^{-\frac{(x-\mu_1)^2}{\sigma}}$$

(1 dim, 1 feature)

$$\ln \frac{P(C_1|x)}{P(C_2|x)} = \frac{(x-\mu_2)^2 - (x-\mu_1)^2}{\sigma} = ax + b \longleftarrow$$

Linear in x
if same variance, quadratic term cancels out

Logistic regression may not work if the data distribution is far from a Gaussian!



Summary

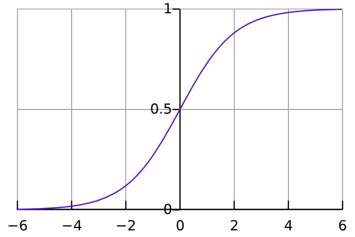
$$y = P(C_1|x)$$ Probability to be in class 1 given x

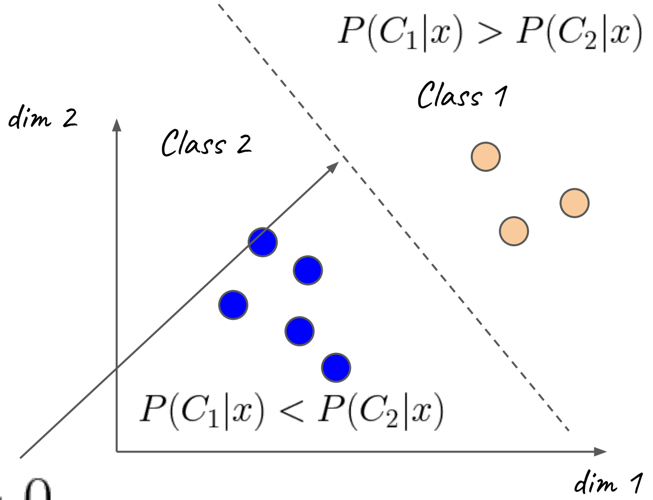
Introducing the logit function:

$$\ln \frac{P(C_1|x)}{1 - P(C_1|x)} = a^T x + b \longleftarrow$$ logit: values before applying the sigmoid

leads to the logistic function

$$y = f(a^T x + b)$$





$$a^T x + b = 0$$

It is a line that separates the 2 classes (a hyperplane in higher dimensions)

Logistic regression separates the classes with a straight line.

We do not fit the data (regression), we find the separation (classification)

3.4 Learning with logistic regression

The squared error is not so good, as it is not convex anymore and the derivative is not straight forward.

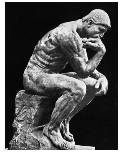
Let us introduce the **cross entropy loss**:

$$L = -\sum_i (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

y_i	True label for sample *i* (0 or 1)
$\hat{y}_i$	Prediction in range [0, 1]

Why this loss function?

Several explanations. One intuitive is here:
https://www.youtube.com/watch?v=KHVR587oW8I



Several interpretations for it. One is by computing the probability to have such a configuration of points.

- For points in class 1, $y_i = 1$
- For points in class 2, $y_i = 0$

We assume independence between points.
How likely is it to have such a configuration:

$$P(C) = \prod_{i=1}^{N} P(C_1|x_i)^{y_i} P(C_2|x_i)^{1-y_i}$$

$$P(C) = \prod_{i=1}^{N} \hat{y}_i^{y_i} \; (1-\hat{y}_i)^{(1-y_i)}$$

Exponent with true value

Prediction of the model for sample *i*

This quantity is maximum when the prediction is correct

- We take the log of this quantity (standard trick: maximizing x or log(x) is equivalent)
- we add a - sign to turn the maximization into a minimization problem (standard trick too)

We get our loss function

$$L = -\sum_i \left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

Minimizing the cross entropy

$$L = -\sum_i \left(y_i \log\left(f(a^T x_i)\right) + (1-y_i)\log\left(1 - f(a^T x_i)\right)\right)$$

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

$$a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ b \end{pmatrix}$$

Can we proceed similarly to the linear regression? using derivative = 0 ?
Let us compute the derivative!

Remember:

$$f'(x) = f(x)\left(1 - f(x)\right)$$

For one of the weights we arrive at:

$$\frac{\partial}{\partial a_1} L = \sum_{i=1}^{N} x_{1i}(f(a^T x_i) - y_i)$$

And in general (matrix form):

$$\frac{\partial}{\partial a} L = X^T(\hat{y} - y)$$

No analytic solution for a ! We need to use **gradient descent**

Remark on the derivative / gradient

$$\frac{\partial}{\partial a} L = X^T \underbrace{(\hat{y} - y)}_{}$$

Remark 1. The derivative is small if the prediction is good

Remark 2. remember the gradient for the linear regression:

$$\nabla (f \circ g)(a) = -X^T 2(y - Xa)$$

Very similar!

But not analytic solution for the minimum

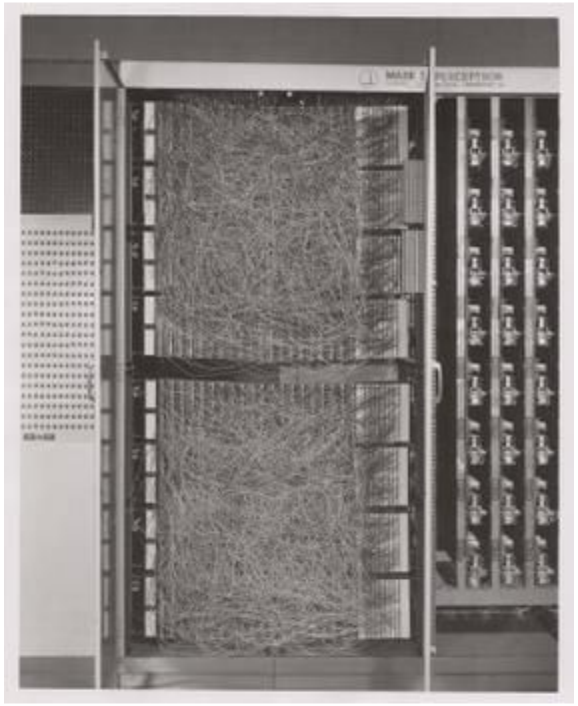
$$\frac{\partial}{\partial a} L = X^T(\hat{y} - y)$$

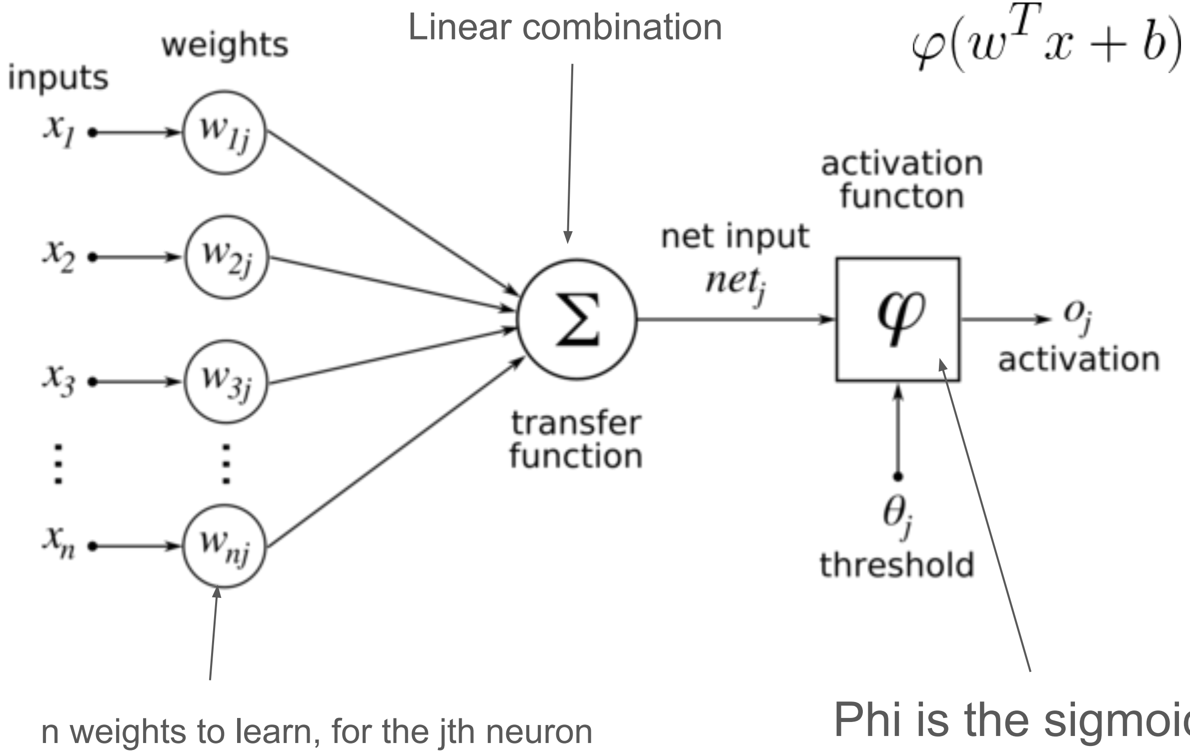
$$\uparrow$$

$$\hat{y}(x) = \frac{1}{1 + e^{-(ax+b)}}$$

What is a when derivative = 0 ?

Logistic regression is one neuron!

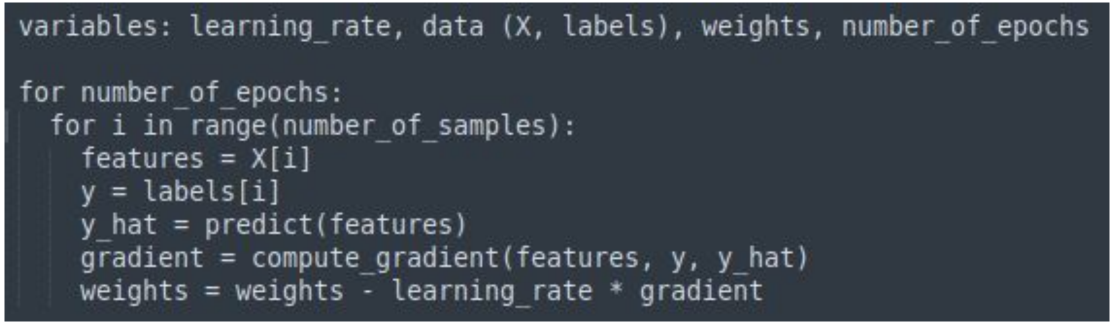




Conclusion

Pseudo-code for logistic regression

We will come back to that after we see gradient descent!



```
variables: learning_rate, data (X, labels), weights, number_of_epochs

for number_of_epochs:
  for i in range(number_of_samples):
    features = X[i]
    y = labels[i]
    y_hat = predict(features)
    gradient = compute_gradient(features, y, y_hat)
    weights = weights - learning_rate * gradient
```

- The learning is parametrized by a **learning rate**
- We show one sample at a time.
- We show the full dataset multiple time: “number_of_epochs”. **Epoch:** one complete pass of the training data