Deep Learning Project – DNA Sequencing

Part 2 : Presentation of the theorical methods

LBRTI2101(B) – Data Science in Bioscience Engineering

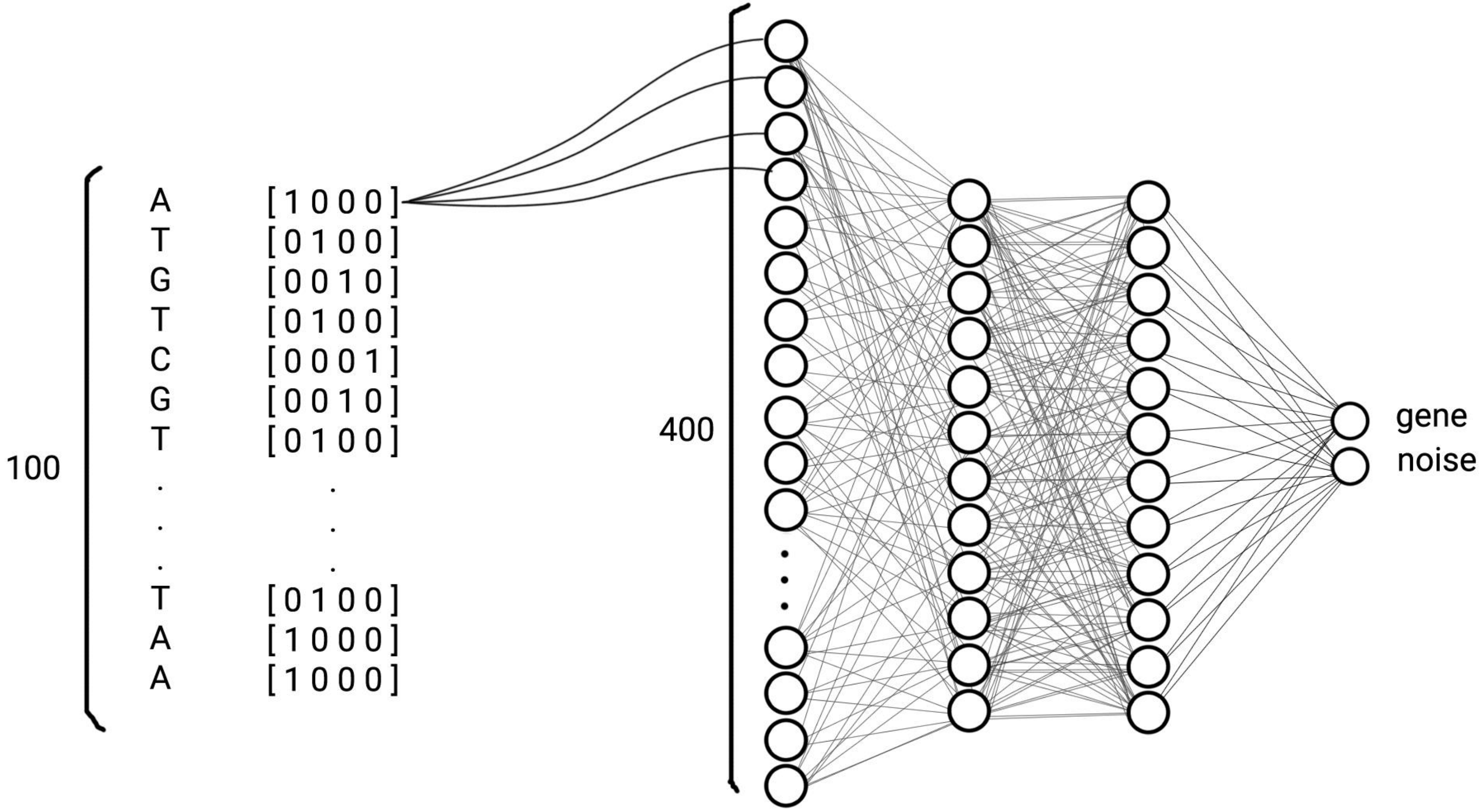
Groupe n°6

Alexis Franco
Noé Laloux
Tom Kenda

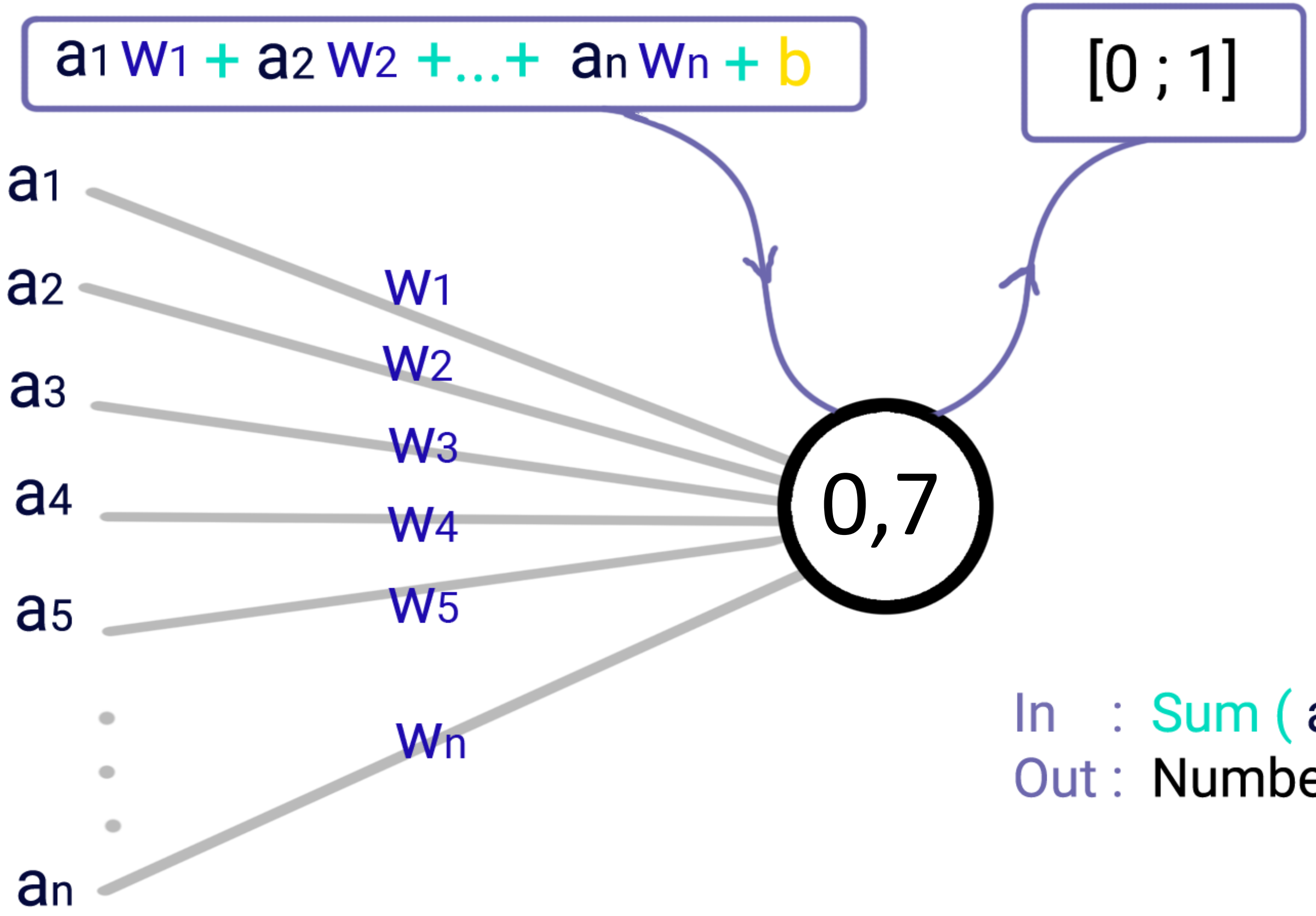
Titulaires : E. Hanert , P. Bogaert

Année académique 2020 - 2021

What is a Neural Network?

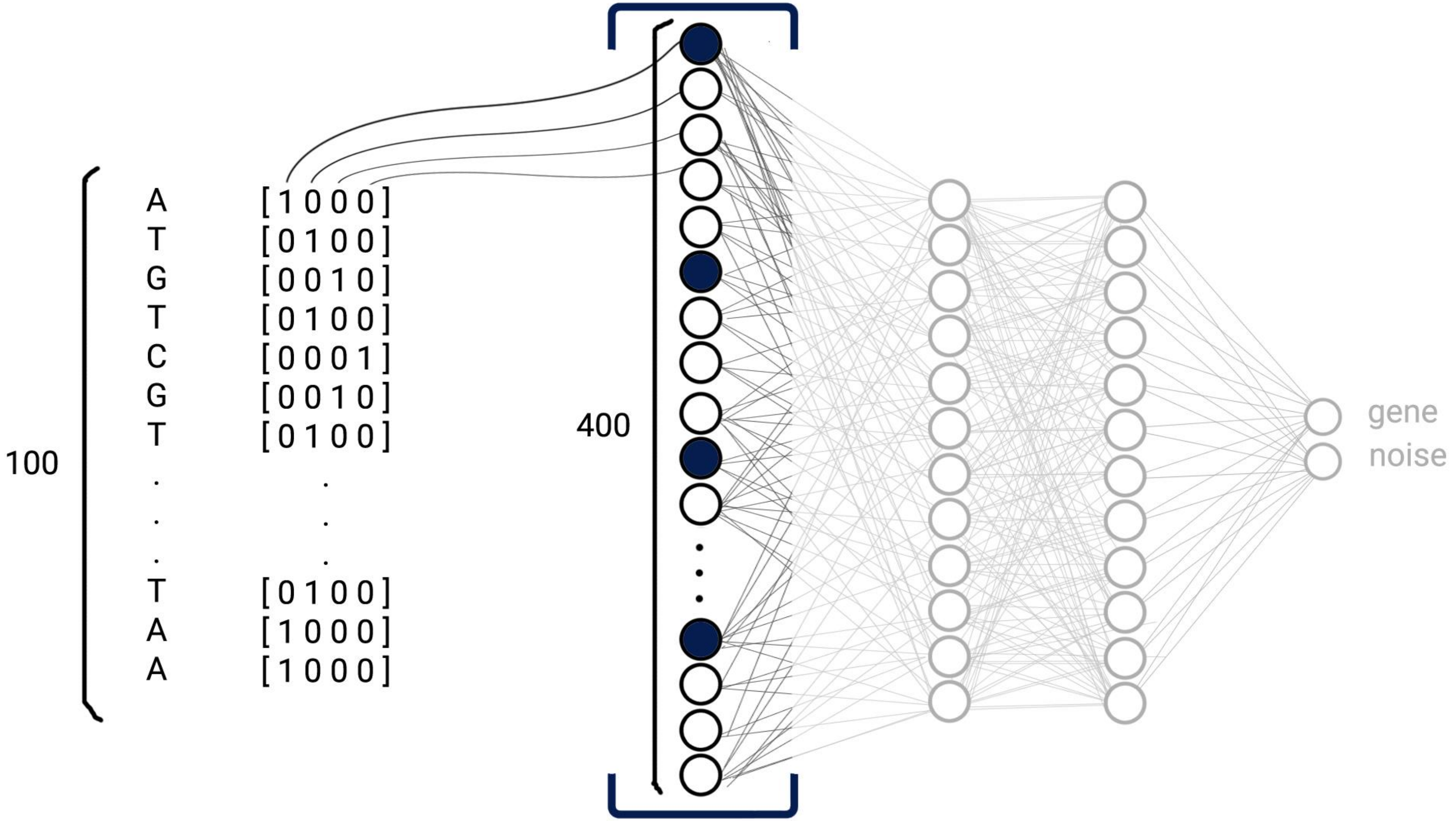


The Neuron

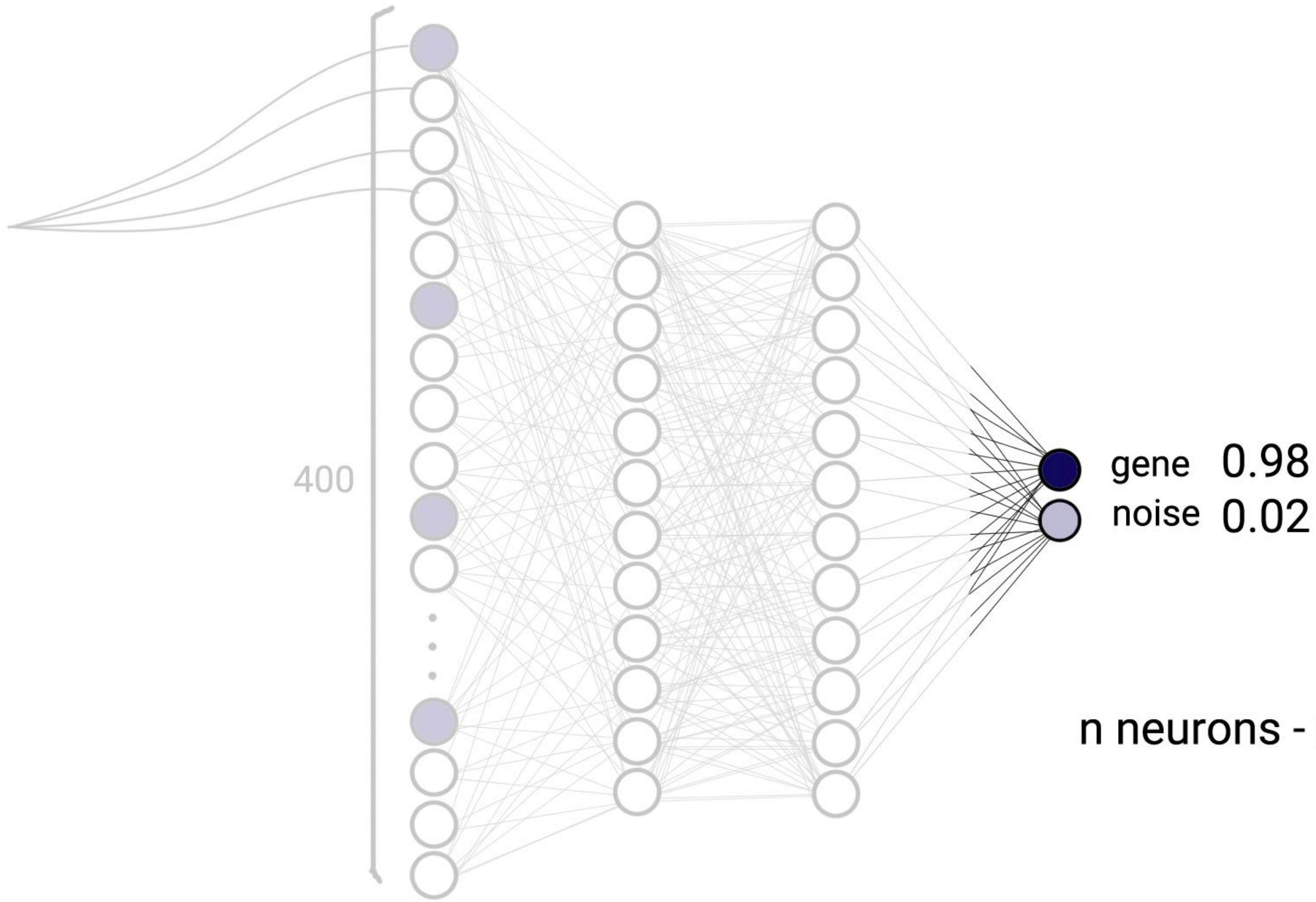


In : Sum (activation * weight) + bias
Out : Number from 0 to 1

The Input Layer



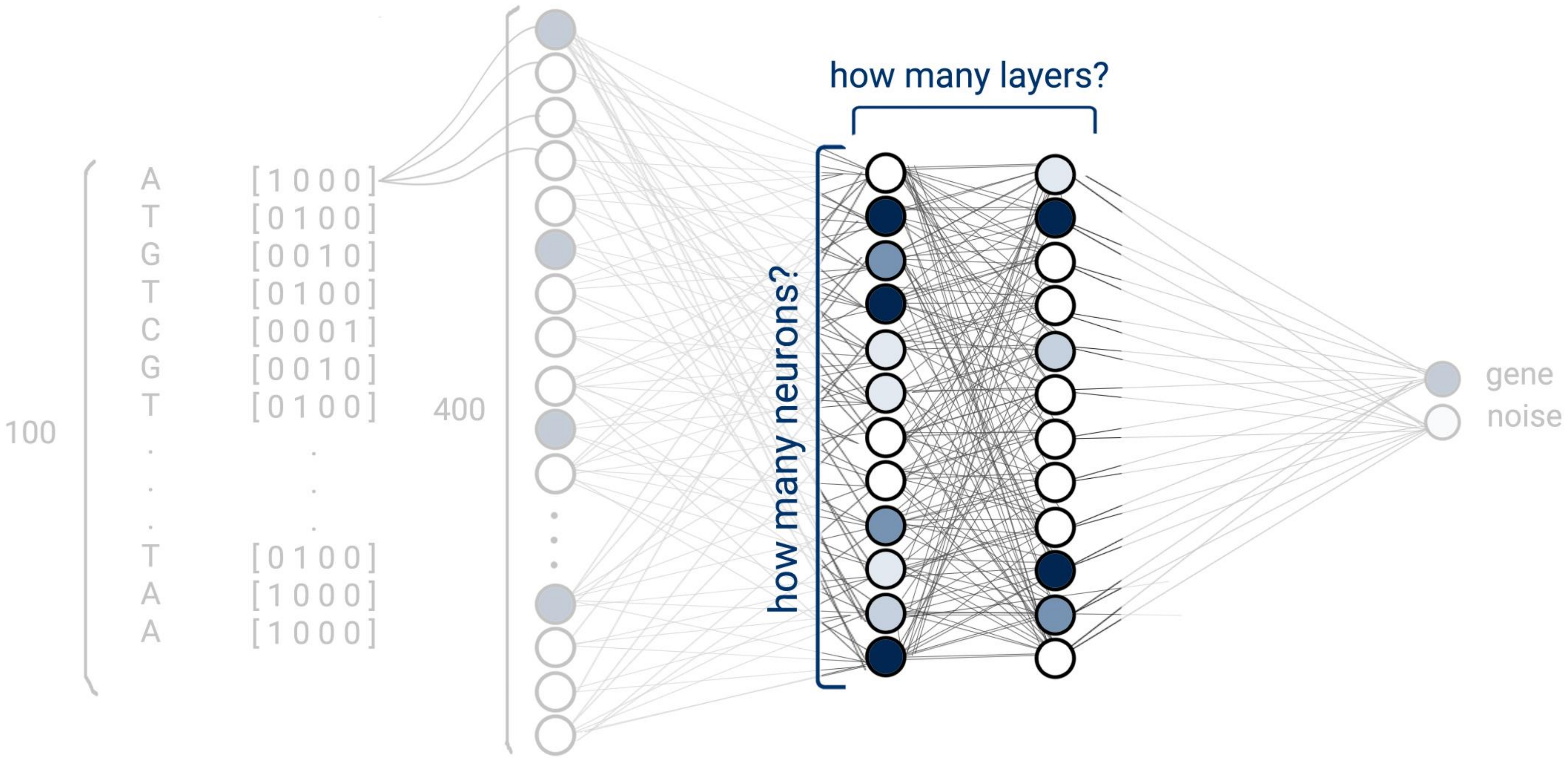
The Output Layer



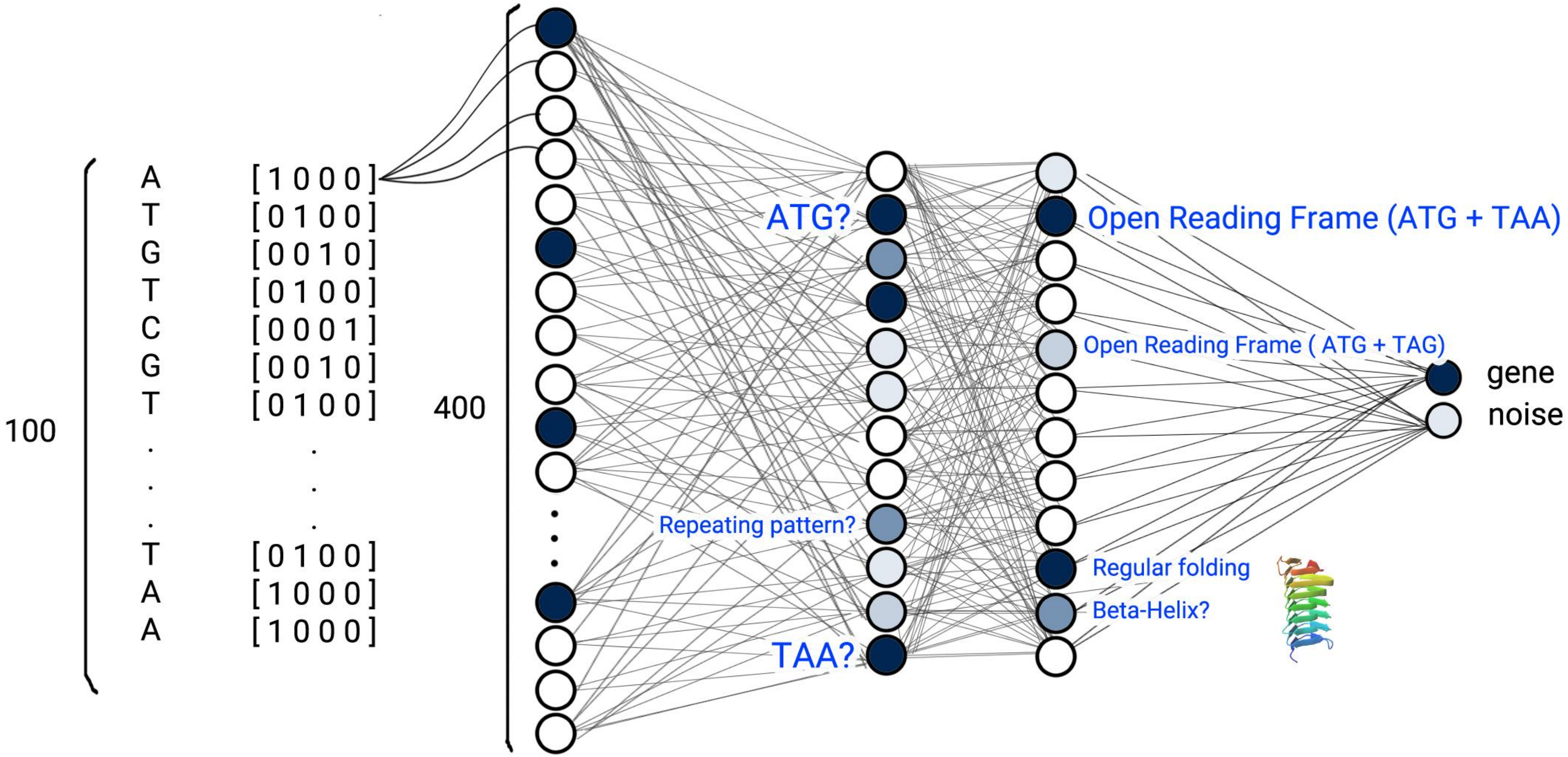
Choice : “gene”

n neurons - n possible outputs

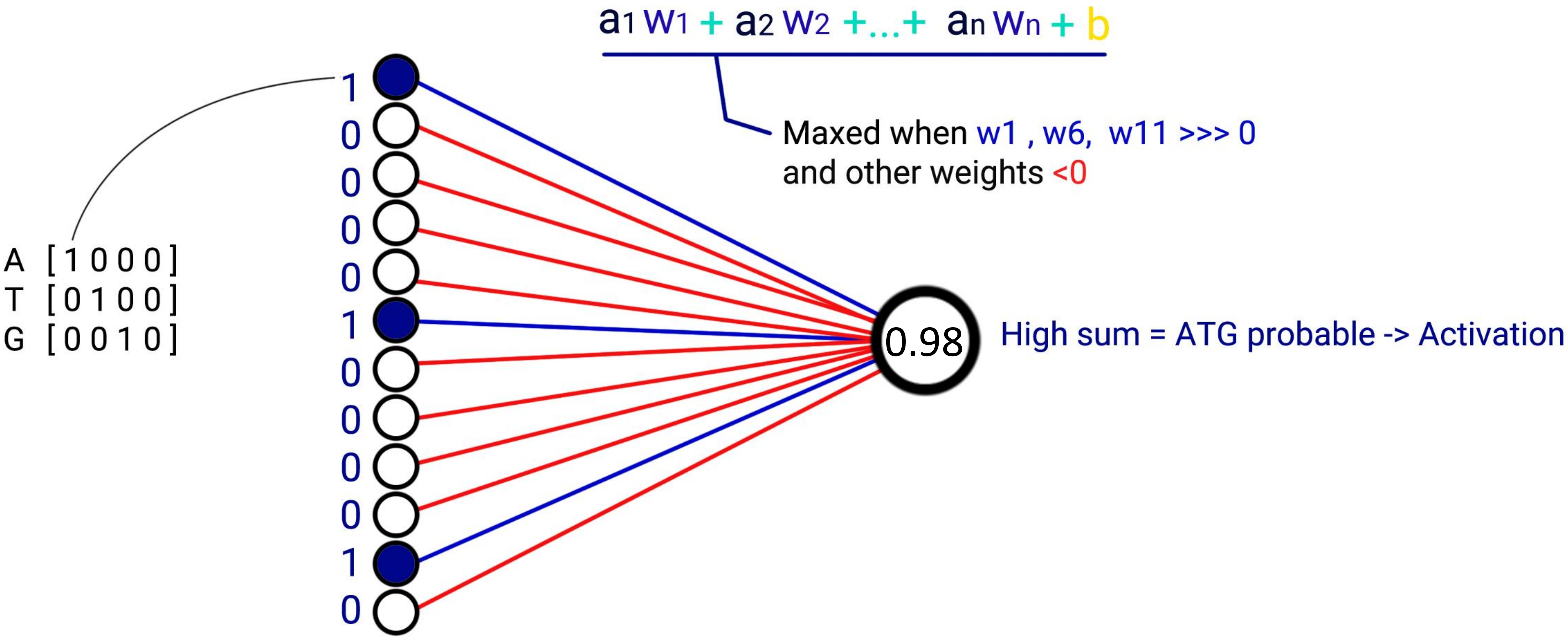
The Hidden Layers



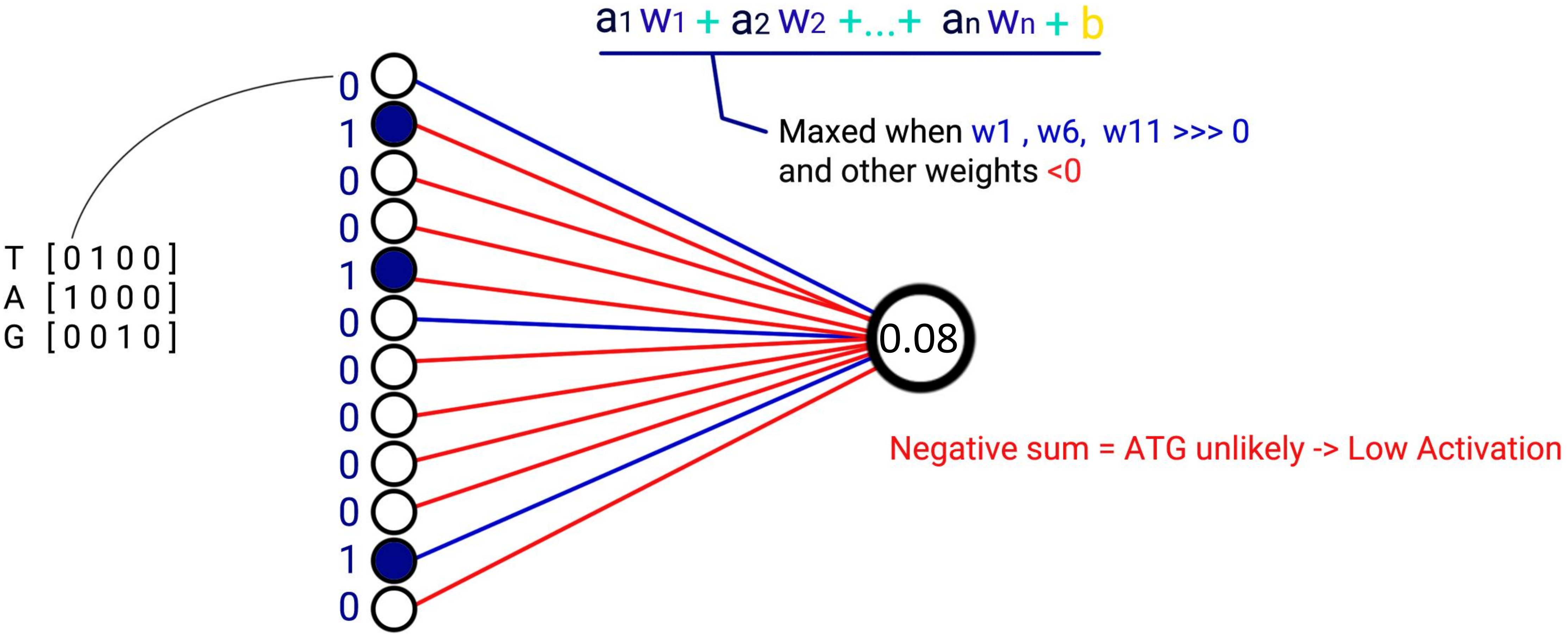
Pattern in layer activation determines activation of the next layer



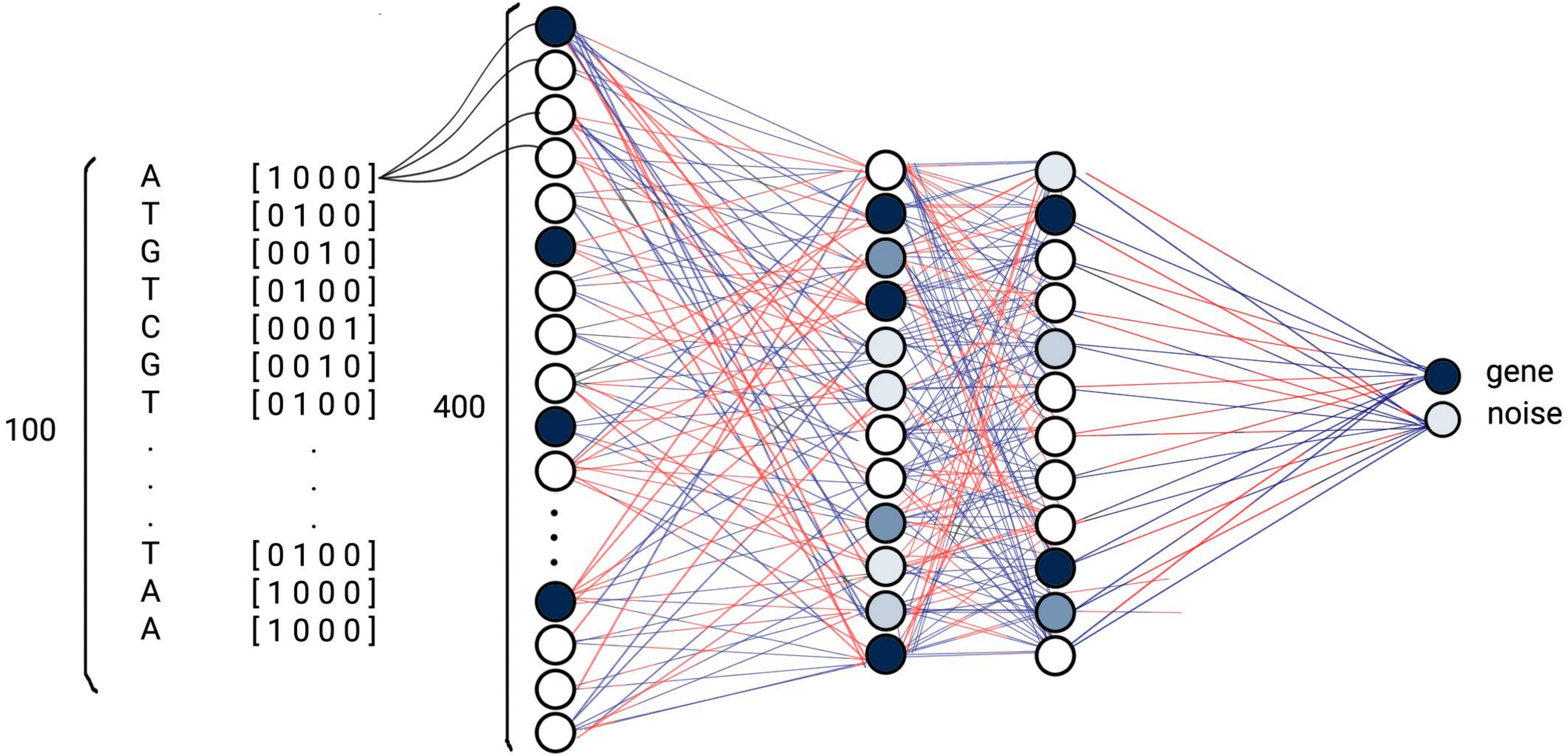
Pattern to activation : Presence of ATG



Pattern to activation : Absence of ATG



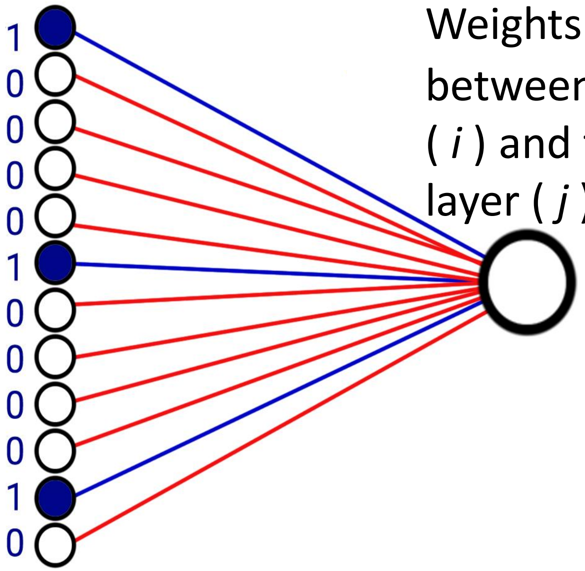
4994 Parameters !



Aggregate function (h_j)

Input (x) :	**Activation function (h) :**	**Output :**
Output values (a_i) of the neurons from the precedent layer	General expressions of h_j : $h_j(a_1, a_2, \dots a_n) = \sum_{i=1}^{n} (a_i * w_{ij})$ $h_j(a_1, a_2, \dots, a_n) = \vec{\boldsymbol{w}} \,.\, \vec{\boldsymbol{a}}$	Value that will be used by the activation function to calculate the output value of the current neuron

Output values (a_i) of the neurons from the precedent layer



Weights (w_{ij}) of the link between the precedent neuron (i) and the one from the next layer (j)

In this case :

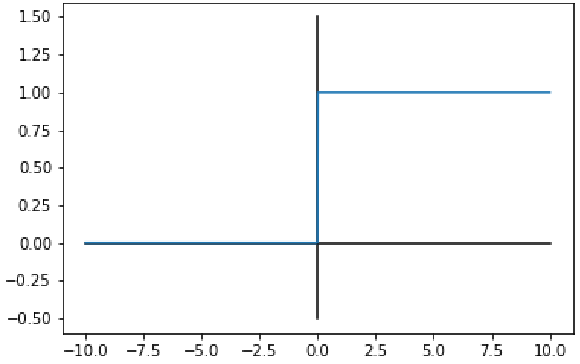
$$h_j(a_1, a_2, \dots a_n) = 1 * w_{1j} + 1 * w_{6j} + 1 \ * w_{11j}$$

Activation function

Input (x) :	**Activation function (g) :**	**Output ($a_j = g(x)$) :**
Sum of : - Aggregation function (h_j) - Bias (b_j) $x = h_j(a_1, a_2, \dots, a_n) + b_j$	Several types of activation functions are possible :	Output value (a_j) of the neuron : number between 0 and 1

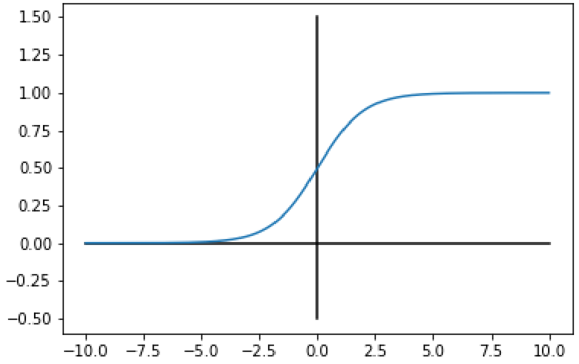
Heaviside (Step) :

$$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



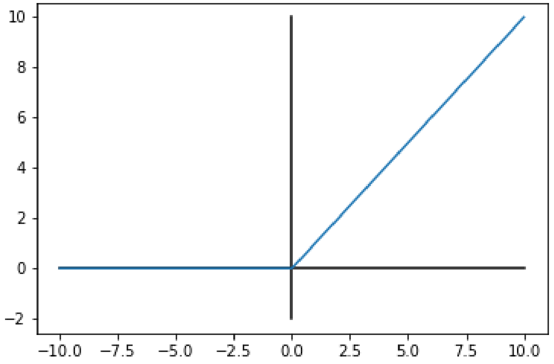
Sigmoid / Logistic :

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



ReLU (Rectified Linear Unit) :

$$g(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



Loss function

Input :	Loss function :	Output :
- Output value *y* predicted by the model for the given data *x* - Output value *t* attended for the data *x*	Several different loss function are possible :	A value that gives error done by the prediction as a number

Mean Squared Error :

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (y_i - t_i)^2$$

Mean Squared Logarithmic Error :

$$MSLE = \frac{1}{n} * \sum_{i=1}^{n} (\log(y_i + 1) - \log(t_i + 1))$$

→ In the model, we want to minimize the value of this loss function

Backpropagation and parameters optimization

Error on the prediction of the neurons of the output layer :

$$\varepsilon_i^{out} = g'\left(h_i^{out} + b_i\right) * (y_i - t_i)$$

Backpropagation of the error of the layer *n* to the neuron *j* from the *n-1* layer :

$$\varepsilon_j^{n-1} = g'^{(n-1)}\left(h_j^{n-1} + b_j\right) * \sum_{i=1}^{m} w_{ij}^n * \varepsilon_i^n$$

Derivate of the activation function evaluated at the value of the input of the neuron *j* in the *n-1* layer

Sum of the products of the weigths (w_{ij}) and the errors on the predictions of the neurons from the *n* layer

With the error related to each neuron, we can now adjust the weights to make better predictions with the next inputs :

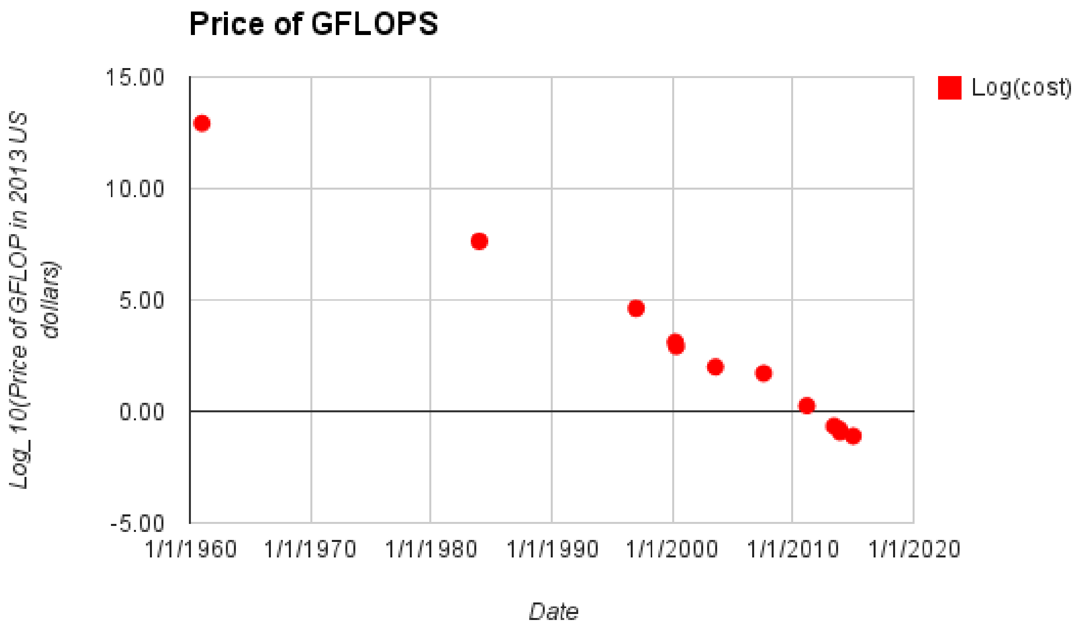
$$w_{ij}^n = w_{ij}^n \; - \; \lambda * \varepsilon_i^n * a_j^{n-1}$$

with λ = learning rate (value between 0 and 1)

General considerations

1. Limits of Deep Learning

- Amount of data needed
  - Memory
  - Time
- Technical limit
  - Computing power $(10^{14} \rightarrow 10^{28}\ GFLOPS)$
  - Technical and environmental cost
- Non modellable data :
  - Reasoning problem : programming, applying scientist method
  - Long term planning, algorithmic-like data manipulation, …
  - No continuous geometric morphing



Trends in the cost of computing

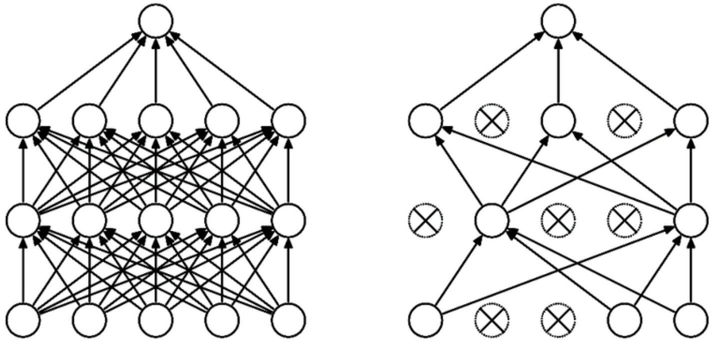
2. Overfitting and underfitting

- Importance of noise ? $y = \int(x) + \boldsymbol{\varepsilon}$
- High variance
- High degree polynomial ~ too complex model

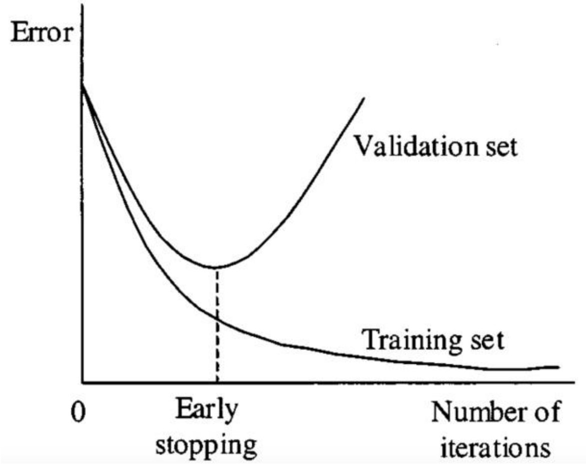
→ Problem : loses of its predictive power on new samples

→ Solution : regularization – imposing constraint *(find out more)*

Example of methods: Dropout, Early Stopping, Euclidian regulation



Dropout (Research Gate)



Early Stopping (source)

Today’s presentation

→ Understanding Deep Learning

What’s next ?

→ Apply the techniques for the detection of protein coding genes

Question from last presentation :

- ✓ Can we find some data ?
- ❑ Any existing libraries/algorithm doing this ? → see you in 2 weeks
- ❑ Is Deep Learning relevant for this application ? → see you in 6 weeks

Bibliography

[Data analytic post – overfitting]

[The limitations of deep learning]

[The Computational Limits of Deep Learning Are Closer Than You Think]

[Trends in the cost of computing]