



École des Ponts
ParisTech

DATA SCIENCE FOR INDUSTRY

Semaine d'Ouverture

Intelligence Économique – 03INT

```
import numpy as np
class Network(object):
    def __init__(self, sizes):
        self.num_layers = len(sizes)
        self.sizes = sizes
        self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
        self.weights = [np.random.randn(y, x)
                        for x, y in zip(sizes[:-1], sizes[1:])]
    def feedforward(self, a):
        for b, w in zip(self.biases, self.weights):
            a = sigmoid(np.dot(w, a)+b)
        return a
    def SGD(self, training_data, epochs, mini_batch_size, eta,
            test_data=None):
        if test_data: n_test = len(test_data)
        n = len(training_data)
        for j in xrange(epochs):
            random.shuffle(training_data)
            batches = [
                training_data[k:k+mini_batch_size]
                for k in xrange(0, n, mini_batch_size)]
            for mini_batch in batches:
                self.update_mini_batch(mini_batch, eta)
            if test_data:
                print "Epoch (%d): (%d) / (%d)" % (j+1,
                                                    self.evaluate(test_data, n_test),
                                                    n)
            else:
                print "Epoch (%d): complete" % (j+1)
```

1,5 ECTS

Modules Obligatoires

Architecture Big Data – ABDAT

Apprentissage Automatique pour la Recherche Opérationnelle – ROAPA

IA pour la Collaboration Homme-Machine – IACHM

Ingénierie Système – INSYS

Management Stratégique des Entreprises – MSTR

Machine Learning et Applications (Master MFD) – MSAMA*

Introduction à la Vision Artificielle (ENS Ulm) – ENSVI*

Deep Learning – DEEPL*

```
1,5 ECTS
2 ECTS
3 ECTS
2 ECTS
3 ECTS
6 ECTS
3 ECTS
2 ECTS
3 ECTS
4,5 ECTS
3 ECTS
4 ECTS
4 ECTS
```

Modules Électifs

(À choisir parmi la liste ci-dessous)

Évaluation Environnementale et Conception (GMM) – COECO

Science du Changement Climatique (SEGF) – SGFCH

Projet Data Science

1 Module parmi les cours de l'ESSEC

1 Module parmi les Masters MVA, MPRO ou MFD

```
def update_mini_batch(self, mini_batch, eta):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]
    for x, y in mini_batch:
        delta_nabla_b, delta_nabla_w = self.backprop(x, y)
        nabla_b = [nb+dnb for nb, dnb in zip(nabla_b, delta_nabla_b)]
        nabla_w = [nw+dnw for nw, dnw in zip(nabla_w, delta_nabla_w)]
    self.weights = [w-(eta/len(mini_batch))*nw
                    for w, nw in zip(self.weights, nabla_w)]
    self.biases = [b-(eta/len(mini_batch))*nb
                   for b, nb in zip(self.biases, nabla_b)]
def backprop(self, x, y):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]
    # feedforward
    activation = x
    activations = [x] # list to store all the activations, layer by layer
    z = [] # list to store all the z vectors, layer by layer
    for b, w in zip(self.biases, self.weights):
        z = np.dot(w, activation)+b
        activations.append(activation)
        activation = sigmoid(z)
    # backward pass
    delta = self.cost_derivative(activations[-1], y) * \
            sigmoid_prime(zs[-1])
    nabla_b[-1] = delta
    nabla_w[-1] = np.dot(delta, activations[-2].transpose())
    for l in xrange(2, self.num_layers):
        z = zs[-l]
        sp = sigmoid_prime(z)
        delta = np.dot(self.weights[-l+1].transpose(), delta)
        nabla_b[-l] = delta
        nabla_w[-l] = np.dot(delta, activations[-l-1].transpose())
    return (nabla_b, nabla_w)
def evaluate(self, test_data):
    test_results = [(np.argmax(self.feedforward(x)), y)
                    for (x, y) in test_data]
    return sum(int(x==y) for (x, y) in test_results)
def cost_derivative(self, output_activations, y):
    return (output_activations-y)
def sigmoid(z):
    return 1.0/(1.0+np.exp(-z))
def sigmoid_prime(z):
    return sigmoid(z)*(1-sigmoid(z))
```