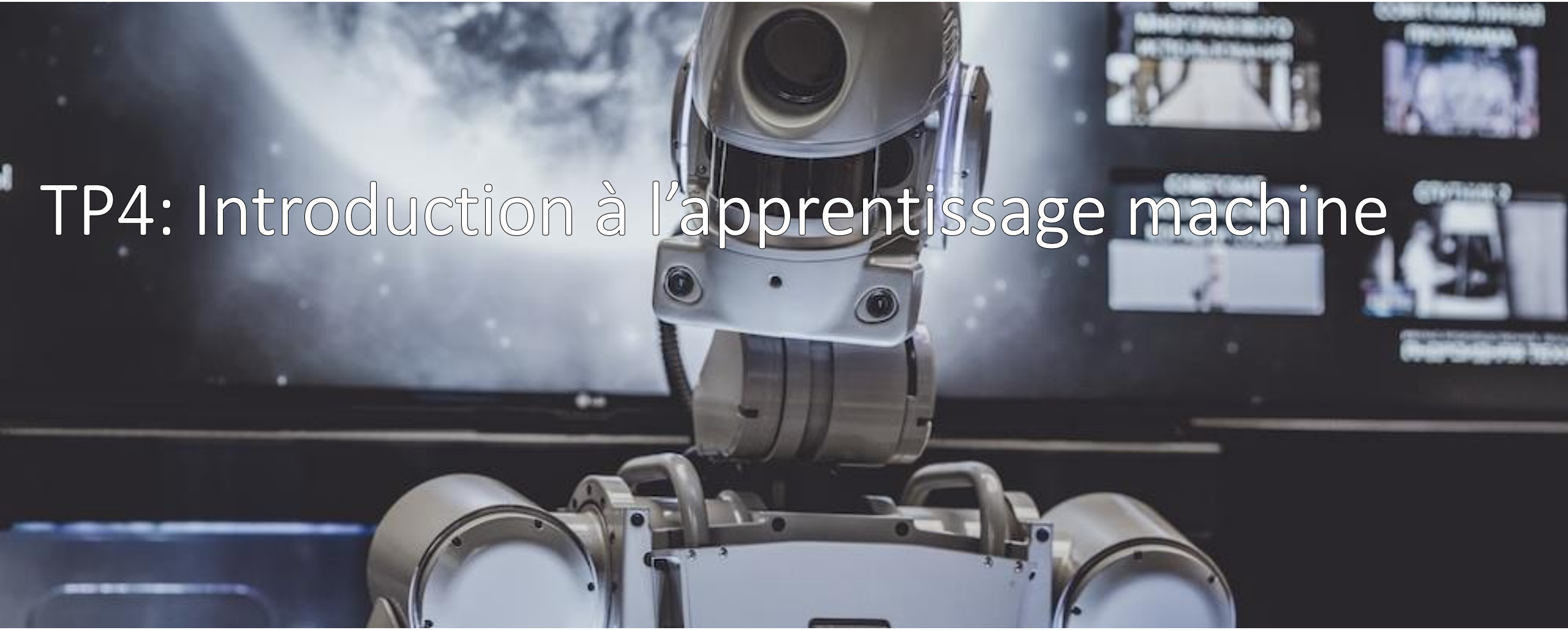
ECE
ÉCOLE D'INGÉNIEURS
ENGINEERING SCHOOL

ING3 – **Calcul embarqué**



TP4: Introduction à l'apprentissage machine

PÔLE ÉLECTRONIQUE

Dr Foued DERRAZ
fderraz@ece.fr

Régression linéaire…

Algorithme de regression linéaire

avec hypothèse $\mathrm{y} = \mathrm{h}_\theta(x) = \theta_1 x + \theta_0$

$$J = \sum_{i=1}^{m} (\mathrm{h}_\theta(x) - \mathrm{y})^2$$

descente de gradient

$$do \begin{cases} \theta_0^{n+1} := \theta_0^n - \frac{\alpha}{m} \sum_{i=1}^{m} \left(\mathrm{h}_\theta\left(\mathrm{x}^{(i)}\right) - \mathrm{y}^{(i)}\right); \\ \theta_1^{n+1} := \theta_1^n - \frac{\alpha}{m} \sum_{i=1}^{m} \left(\mathrm{h}_\theta\left(\mathrm{x}^{(i)}\right) - \mathrm{y}^{(i)}\right) \mathrm{x}^{(i)}; \\ n := n + 1; \\ erreur = erreur + \left(\mathrm{h}_\theta\left(\mathrm{x}^{(i)}\right) - \mathrm{y}^{(i)}\right)^2 \end{cases}$$

while $(erreur < 1e - 6)$

$$\theta_0 := \theta_0^{n+1}$$

$$\theta_1 := \theta_1^{n+1}$$

lire x(m+1)

Predire Y(m+1):=$\theta_1\ x(m+1) + \theta_0$

Régression linéaire…

Variables et initialisation

```
#define NNmax 5 // size of the vector to be
learned
#define Nmax 200 // max iteration of your
algorithm
double  error[Nmax];
double x[NNmax] = {20.,50.,42.,25.,70.};
double y[NNmax] = {80.,60.,50.,30.,90.};

double theta0(0.); // Slope
double theta1(0.); // Intercept
```

$n \leftarrow 0;$

$\theta_0^0 \leftarrow 0$

$\theta_1^0 \leftarrow 0;$

$m \coloneqq taille\ de\ vecteur\ d'entrainement:$

erreur:=0;

lire le vecteur x(1:m);

Lire le vecteur y (1:m);

Régression linéaire…

Fonction hypothèse

```
double  hypoLR(double t0, double t1,double x)
{
  return(t0+t1*x);
}
```

$$y = h_\theta(x) = \theta_1 x + \theta_0$$

Régression linéaire…

Fonction d'apprentissage

```
for(int i=0;i<Nmax; i++)
{
    err0=0.;
    err1=0.;
    for(int j=0;j< NNmax;j++){
    index =j % NNmax;
    Y_predict = hypoLR(theta0, theta1,x[index]);
    tmperr0 = Y_predict - y[index];
    tmperr1 = (Y_predict - y[index])*x[index];
    err0+=tmperr0;
    err1+=tmperr1;
    }
    theta0 -= alpha * err0/NNmax;
    theta1 -= alpha * err1/NNmax;
}
```

Algorithme

$$\begin{cases} for(n = 0\,;\ n < Nmax; n++) \\ \{ \\ \theta_0^{n+1} := \theta_0^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right); \\ \theta_1^{n+1} := \theta_1^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}; \\ n := n + 1; \\ \} \end{cases}$$

Régression linéaire…

Fonction d'apprentissage

```
do
{ //code arduino
    err0=0.;
    err1=0.;
    for(int j=0;j< NNmax;j++){
    index =j % NNmax;
    Y_predict = hypoLR(theta0, theta1,x[index]);
    tmperr0 = Y_predict - y[index];
    tmperr1 = (Y_predict - y[index])*x[index];
    err0+=tmperr0;
    err1+=tmperr1;
    }
    theta0 -= alpha * err0/NNmax;
    theta1 -= alpha * err1/NNmax;
    error=err0*err0/NNmax;
    error=err0*err0;
    erreur-=error;
}while(fabs(error)>1.e-3 );
```

Algorithme

$$do\begin{cases}\theta_0^{n+1} := \theta_0^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right); \\ \theta_1^{n+1} := \theta_1^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}; \\ n := n+1; \\ erreur = erreur + \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2\end{cases}$$

$$while\ (erreur > 1e-3)$$

Régression linéaire…

```
do
{
    err0=0.;
    err1=0.;
    for(int j=0;j< NNmax;j++){
    index =j % NNmax;
    Y_predict = hypoLR(theta0, theta1,x[index]);
    tmperr0 = Y_predict - y[index];
    tmperr1 = (Y_predict - y[index])*x[index];
    err0+=tmperr0;
    err1+=tmperr1;
    }
    theta0 -= alpha * err0/NNmax;
    theta1 -= alpha * err1/NNmax;
    error=err0*err0/NNmax;
    error=err0*err0;
    erreurs-=error;
}while(fabs(erreurs)>1.e-3 );
```

$$do\begin{cases}\theta_0^{n+1} := \theta_0^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right); \\ \theta_1^{n+1} := \theta_1^n - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}; \\ n := n+1; \\ erreur = erreur + \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2\end{cases}$$

$$while\ (erreur > 1e-3)$$

Régression linéaire…

Fonction d'apprentissage

```
void learnerMLRLinear() {
  int i(0) ;
  int index(0.);
  double err0,err1;
  double tmperr0(0.), tmperr1(0.);
    do
    {
        //int index = i % NNmax;
        err0=0.;
        err1=0.;
        for(int j=0;j< NNmax;j++){
        index =j % NNmax;
        Y_predict = hypoLR(theta0, theta1,x[index]);
        tmperr0 = Y_predict - y[index];
        tmperr1 = (Y_predict - y[index])*x[index];
        err0+=tmperr0;
        err1+=tmperr1;
        }
        theta0 -= alpha * err0/NNmax;
        theta1 -= alpha * err1/NNmax;
    //theta0 = theta0 - alpha * err;
        //theta1 = theta1 - alpha * err * x[index];
    }while(i++<Nmax );
}
```

Régression linéaire…

Fonction de prédiction

```
void prediction(double X) {
    double predict = theta0 + theta1* X;
  Serial.print("Y Prediction = \t");
  Serial.print(theta0);
  Serial.print("+\t");
  Serial.print(theta1);
  Serial.println("*x");


  }
```

Régression Logistique et SVM…