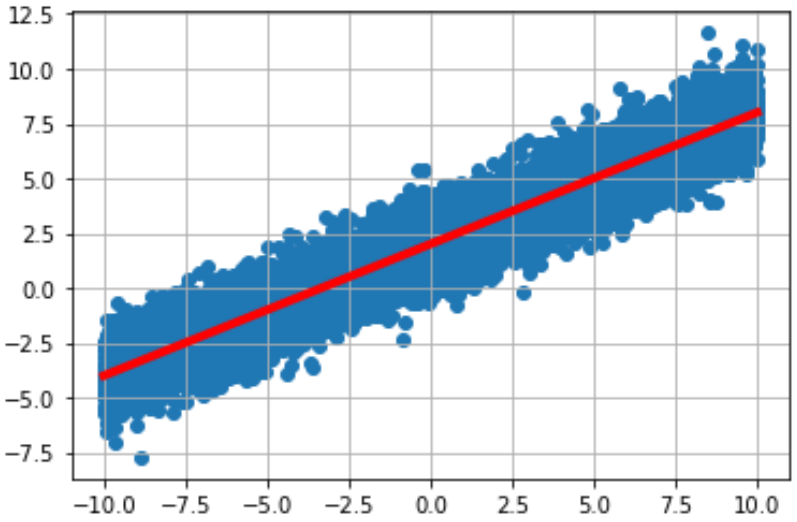
TP n°3-4: Les réseaux de neurones

♦ Exercice 1 Mon premier réseau de neurones

On va essayer de répondre au problème de la droite affine qui approche le mieux un nuage de points.
Vous devez effectuer les opérations suivantes :

1. Simuler un échantillon de 100 000 points autour d'une droite prédéfinie entre $[-10, 10]$.
2. Créer un modèle avec un seul neurone avec keras (une seule entrée et une seule sortie).
3. Entrainez le modèle sur 1, 2, 3, 4, 5 époques
4. Visualiser le résultat (la droite obtenue avec les points).



Memo keras

- Charger `keras`

```
from tensorflow import keras
```

- Créer un nuage de points autour d'une droite

  `Utilisez linspace(), numpy.random.normal()`

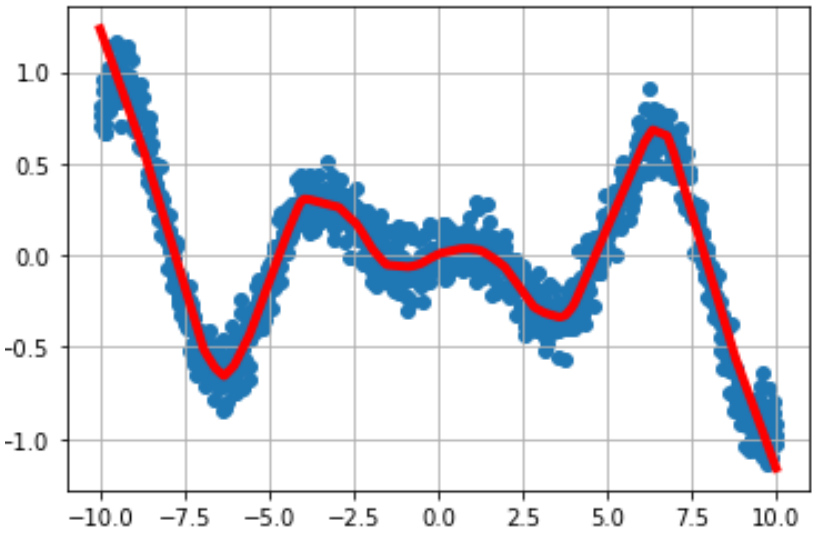
- Créer un réseau de neurones classique

  `Dans keras, Sequential(), layers, summary() seront utiles`

♦ Exercice 2

Même exercice que le précédent sauf que l'on choisit un nuage point autour d'une courbe (polynôme ou trigonométrique).
Mais c'est un peu plus compliqué ! On ne va pas approcher cela avec une droite. Donc un neurone ne sera pas suffisant.

TP n°3-4: Les réseaux de neurones

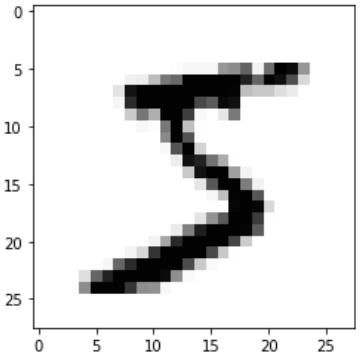


On va donc tester les modèles suivants :

1. Un modèle avec une seule couche (et un certain nombre de neurones !)
2. Un modèle à 2 couches
3. Un modèle à 3 couches

Réfléchissez bien à l'architecture du réseau ! (que donnez vous en entrée ? et que voulez vous en sortie ?) Conclure.

♦ **Exercice 3** On va maintenant travailler à reconnaître des chiffres. Pour cela, il vous faudra utiliser l'ensemble de données **mnist** de la bibliothèque keras.
Première étape : Visualiser les données
Le premier script aura pour but d'obtenir une visualisation des éléments de notre ensemble de données :



On remarque que les données d'entrée sont des images carrées de 26 pixels de coté. L'idée consiste à donner un vecteur au réseau de neurones. Il nous faut donc convertir cette image en un vecteur de taille 26×26. On pensera à les convertir en réels et à normer le vecteur. Le but du jeu est concevoir un neurone capable de reconnaître un chiffre donné. Par exemple, on essayera de reconnaître le 0. Ce qui signifie que la sortie sera à 1 si c'est un 0 et à 0 si ce n'est pas 0...
Pour ce faire, suivez les étapes decrites ci-dessous :

1. Mélangez le jeu de données. (shuffle)
2. Séparer le jeu de données en deux : un jeu d'entrainement (80% des données) et un jeu de test (le reste)
3. créer la fonction sigmoïde et la visualiser

4. définir une fonction d'erreur (celle du cours)
5. créer un programme principal qui permet l'apprentissage avec les formules vues en cours/TD.
6. lancer la résolution pour plusieurs époques.

Ce qui constitue la résolution « à la main » de ce problème.
On continue avec la vérification de notre modèle. Pour ce faire, suivez ce qui suit :

1. Créer une image de 28x28 pixels avec un chiffre à la main
2. L'importer dans votre script python précédent
3. Tester la prédiction de votre modèle dessus
4. Conclure
5. Selectionner une image de l'ensemble de données initial
6. Prenez le négatif de l'image
7. Tester la prédiction de votre modèle dessus
8. Conclure

Memo `mnist`

- Charger `mnist`

```
from keras.datasets import mnist
```

- Quelques informations utiles sur mnist `https://keras.io/api/datasets/mnist/`
- Créer un réseau de neurones classique

  `Dans keras, Sequential(), layers, summary() seront utiles`

♦ **Exercice 4** Création d'un environnement de développement. L'idée consiste en une décomposition de la résolution passant par les réseaux de neurones pour éventuellement pouvoir rétroagir sur le modèle. Donc, on ne passera pas par les fonctions toutes faites de keras pour résoudre ce problème. On l'a [normalement] déjà fait à l'exercice précédent.Pour ce faire, on va décomposer le problème de manière classique :

1. Une fonction de construction du modèle
2. Une fonction d'évolution du modèle (Avec un **GradientTape**)
3. Le programme principal

Le synopsis des fonctions peut être le suivant :

— Pour la fonction **model** : Les paramètres classiques sont la taille de l'entrée et le nombre de sortie désirée de notre classifieur. Par conséquent, on aura *largeur*, *hauteur*, *profondeur* pour la taille de l'entrée et *nbClasses* pour le nombre de sorties.
— Pour la fonction d'évolution, on a juste besoin des images d'apprentissage et des étiquettes correspondantes.
— le programme principal sera chargé de définir le reste des paramètres d'entrainement (optimiseur, le nombre d'entrainements, la précision voulue...)

Avec cette approche, vous serez capable de tester différents modèles avec différentes fonction de perte sur le problème de classification de chiffres (mnist).
Maintenant quel modèle pour analyser un jeu de données tel que mnist ?
Voici quelques élements de réponse :

1. un réseau à une seule couche completement connecté
2. Avec un modèle convolutif, les structures sont légions. J'en demande une précise ci-dessous. Mais testez ensuite avec d'autres ! (comme VGG16)

TP n°3-4: Les réseaux de neurones

Voici la description d'un réseau de neurones convolutif qui intégre bon nombre de fonctions utiles :

1. Une couche de convolution 2d avec 30 filtres de convolution ainsi qu'un noyaux d'un pas de c(5,5) et une fonction d'activation relu.
2. Une couche de max-pooling 2d avec un noyau de pooling d'un pas de c(2,2).
3. Une couche de convolution 2d avec 15 filtres de convolution ainsi qu'un noyaux d'un pas de c(3,3) et une fonction d'activation relu.
4. Une couche de max-pooling 2d avec un noyau de pooling d'un pas de c(2,2).
5. Une couche dropout avec un taux d'extinction de 0,3.
6. Une couche d'aplatissement *flatten* pour aplatir les sorties de la couche précédente.
7. Une couche cachée composée de 128 neurones et une fonction d'activation relu.
8. Une couche cachée composée de 50 neurones et une fonction d'activation relu.
9. La dernière couche de sortie composée de 10 neurones et une fonction d'activation softmax

Ceci est un réseau de TP et ne prétend pas être le meilleur !

♦ Exercice 5

on installe tensorboard :

```
$ pip install –U tensorboard
```

L'idée est d'utiliser cette bibliothèque pour obtenir des rapports d'évolution de nos indicateurs. Ce qui servira à faire évoluer notre modèle.
Ensuite, on va créer un modèle pour étudier notre modèle mnist.
https://www.tensorflow.org/tensorboard/get_started
Il y a sur cette page des exemples d'utilisation de cette bibliothèque. Au passage, il y a une solution de l'exercice 3.

PARTIE I : utilisation avec un modèle simple
Reprenez le code que vous avez normalement obtenu à l'exercice 3. Intégrez tensorboard pour obtenir les tableaux récapitulatifs de votre entrainement.

PARTIE II : utilisation avec un modèle simple Même exercice avec le code obtenu dans l'exercice 4. L'intérêt est de pouvoir agir sur le modèle, la fonction de perte... Et d'en constater les évolutions directement sur les tableaux tensorboard.