TP1 : Régression polynomiale

Objectifs

Dans le TP précédent nous avons vu comment créer un modèle linaire. Ce type de modèles ont l'avantage d'être très simple mais exigent d'avoir une relation linéaire entre les données et l'objectif, ce qui est rarement le cas.

Dans ce TP nous allons complexifier notre modèle et faire de la régression polynomiale.

Comme pour le premier TP, nous nous plaçons volontairement dans un cadre académique (c'est à dire avec de "fausses données") afin de bien comprendre tous les mécanismes.

Le but de ce TP est de retrouver la fonction $10 * \sin(x)/x$.

Génération des données

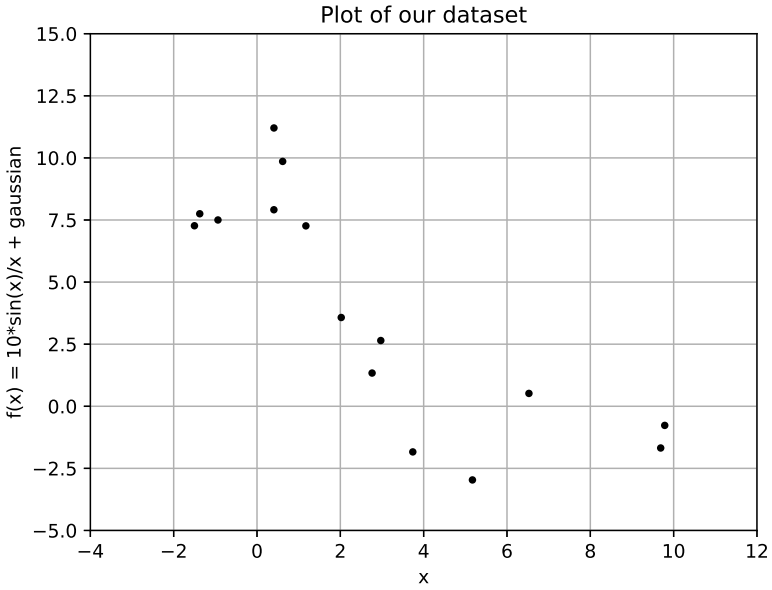
Nous allons tout d'abord générer un ensemble de points bruités pour l'ensemble d'apprentissage.

Il nous faut tout d'abord créer une première fonction qui va retourner *nb* valeurs égales à $10 * \sin(x_k)/x_k$ + une variable aléatoire gaussienne d'espérance 0 et de variance 1.

Nous allons créer cette fonction par étape :

- Commencer par générer nb points aléatoires uniformément dans [-3,10].
- A partir de ces points calculer les nb valeurs selon $10 * \sin(x)/x$.
- Ajouter un bruit gaussien de moyenne 0 et de variance 1 aux valeurs calculées.

Afficher les données dans un graphe.
Pour 15 points j'obtiens :



Quelques explications du modèle

Nous allons maintenant essayer de créer un modèle pour notre problème.

Comme stipulé en début de sujet nous allons nous concentrer sur des modèles polynomiaux de différents dégrés.

Soit un modèle polynomial $g(x, w)$, avec x un vecteur correspondant aux variables, et w ses paramètres. Pour un modèle de degré d, nous avons donc : $g(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$.

Avec $d = 0$, quel type de modèle avons-nous ?

Avec $d = 1$, quel type de modèle avons-nous ?

Supposons que nous n'ayons aucune information, une démarche naturelle consiste à générer des polynomes avec des dégrés de plus en plus grands.

Un premier polynome en sklearn

Nous allons commencer simplement avec un polynome de degré 1.

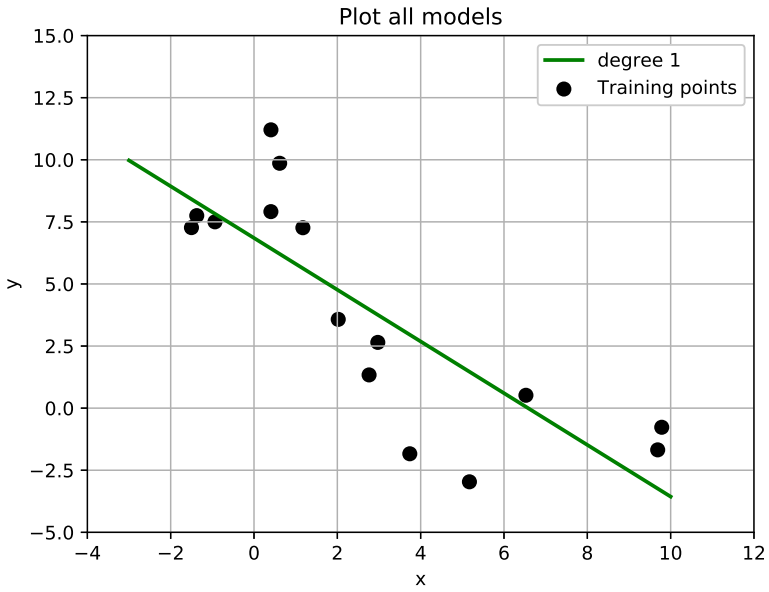
Pour créer un modèle polynomial en python il faut tout d'abord `PolynomialFeatures` qui permet de facilement ajouter des caractéristiques polynomiales ; on rappelle que l'on souhaite trouver

$$y = \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_n x^n p + \beta$$

Il faut ensuite créer un pipeline afin d'appliquer une liste de transformations à un

estimateur avec la fonction `make_pipeline`. L'estimateur que l'on va utiliser est un *Linear Least Squares* avec `Ridge()` en python.

Créer le modèle et l'afficher. Vous devez obtenir quelque chose comme :

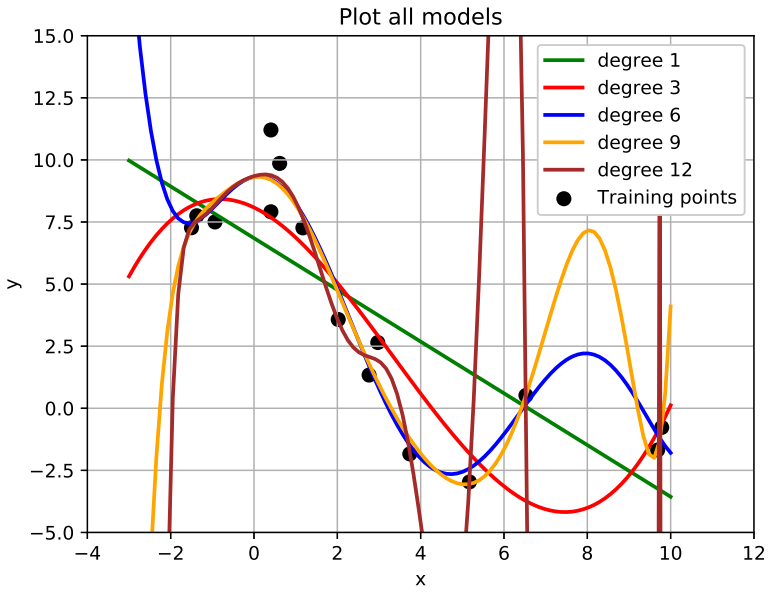


Comme pour le premier TP, nous pouvons mesurer l'erreur résiduelle de notre modèle.

```
Residual sum of squares : 104.15
```

Régression polynomiale d'ordres supérieurs

Calculer maintenant des polynomes d'ordre 1, 3, 6, 9, 12. Vous devriez obtenir ceci :



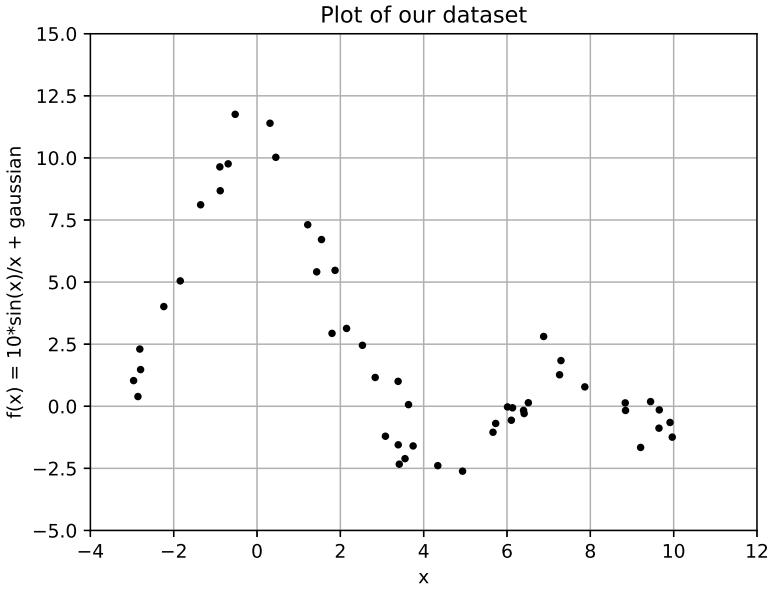
Là encore il nous faut calculer les erreurs résduelles.

```
Degree 1.00 : Residual sum of squares : 104.15
Degree 3.00 : Residual sum of squares : 56.07
Degree 6.00 : Residual sum of squares : 12.95
Degree 9.00 : Residual sum of squares : 11.80
Degree 12.00 : Residual sum of squares : 7.82
```

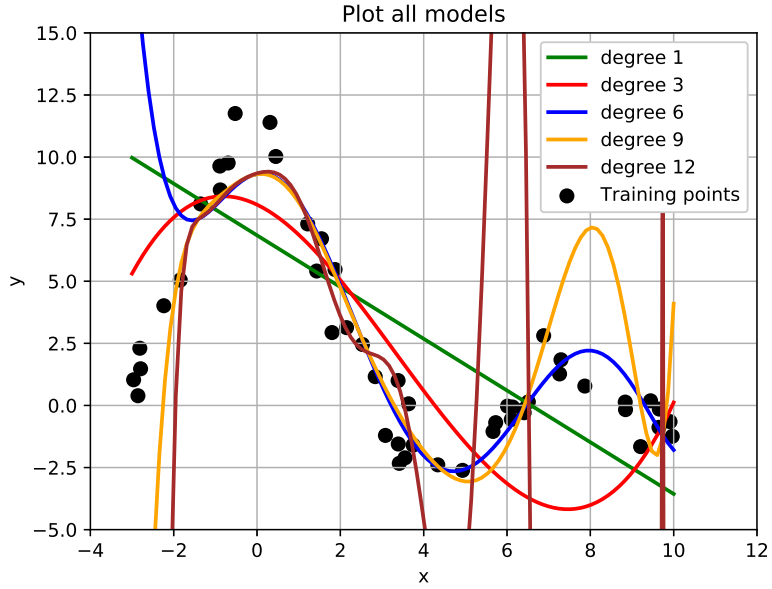
Que pouvez vous en conclure ? Quel modèle colle le mieux aux données ?

Généralisation : test sur une base indépendante

Nous allons maintenant généraliser (comme vu en cours). Créons un ensemble de test à l'aide de la fonction définie dans la première question.



Effectuer une prédiction sur ces points afin de mesurer la généralisation de notre modèle.




```
Degree 1.00 : Residual sum of squares : 658.45
Degree 3.00 : Residual sum of squares : 408.69
```

```
Degree 6.00 : Residual sum of squares : 1039.21
Degree 9.00 : Residual sum of squares : 3321.54
Degree 12.00 : Residual sum of squares : 3956332.86

 Degree 1.00 : R-squared : -0.03
 Degree 3.00 : R-squared : 0.36
 Degree 6.00 : R-squared : -0.63
 Degree 9.00 : R-squared : -4.21
 Degree 12.00 : R-squared : -9520.68
```

Conclusion

- Quel(s) algo(s) a(ont) "sous-appris" ?
- Quel(s) algo(s) ont "sur-appris" ?
- Quel est le modèle qui semble le plus pertinent ?