Tables des matières

In [74]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn import model_selection
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
```

Chargement du fichier

In [75]:
```
df = pd.read_csv('advertising.csv', index_col=0).dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 7.8 KB
```

Obtenir le data et Cible

In [76]:
```
y = df.sales

X = df.drop(['sales'], axis=1)

X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
```

```
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
dtypes: float64(3)
memory usage: 6.2 KB
```

Ensemble de valeurs de alpha (lambda) à utiliser avant de determiner une valeur optimale

In [77]:
```
#Ridge
alphas = 10**np.linspace(10,-2,100)*0.5
alphas
```

Out[77]:
```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

In [78]:
```
# Obtentions des modeles selon les valeurs de alpha des coefficient
# from sklearn.pipeline import make_pipeline
# from sklearn.preprocessing import StandardScaler
ridge =  Ridge()
coefs = []
for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X, y)
    coefs.append(ridge.coef_)
np.shape(coefs)
```

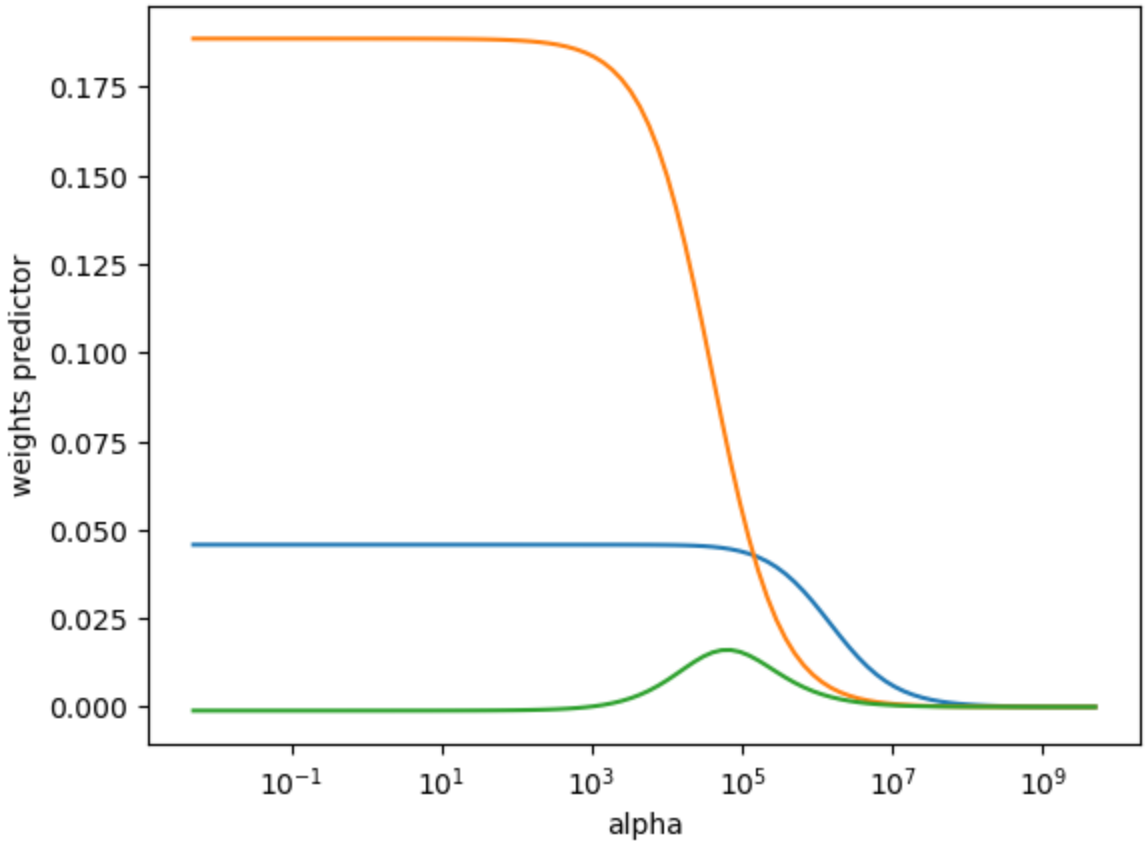
Out[78]:
```
(100, 3)
```

Variation des wieghts selon alpha

In [79]:
```
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
```

```
plt.xlabel('alpha')
plt.ylabel('weights predictor')
```

Out[79]:
```
Text(0, 0.5, 'weights predictor')
```



Split du data

In [80]:
```
# split data into training and test sets
X_train, X_test , y_train, y_test = model_selection.train_test_split(X, y,
                                                                     test_size=0.2,
                                                                     random_state=44)
```

Test avec alpha = 4

In [81]:
```
ridge2 = Ridge(alpha=4)
# Fit a ridge regression on the training data
ridge2.fit(X_train, y_train)

# Use this model to predict the test data
pred2 = ridge2.predict(X_test)

# Print coefficients
print(pd.Series(ridge2.coef_, index=X.columns))
# Calculate the test MSE
print(mean_squared_error(y_test, pred2))
```

```
TV           0.046843
radio        0.178521
newspaper    0.002593
dtype: float64
1.9923299190641555
```

Test avec alpha = 10**10

In [82]:
```
ridge3 = Ridge(alpha=10**10 )
# Fit a ridge regression on the training data
ridge3.fit(X_train, y_train)
# Use this model to predict the test data
pred3 = ridge3.predict(X_test)
# Print coefficients
print(pd.Series(ridge3.coef_, index=X.columns))
# Calculate the test MSE
print(mean_squared_error(y_test, pred3))
```

```
TV           6.018096e-06
radio        7.412761e-07
newspaper    4.967663e-07
dtype: float64
19.68686735293694
```

Test avec alpha = 0 RL

In [83]:
```
ridge2 = Ridge(alpha=0)
# Fit a ridge regression on the training data
ridge2.fit(X_train, y_train)
# Use this model to predict the test data
pred = ridge2.predict(X_test)
# Print coefficients
print(pd.Series(ridge2.coef_, index=X.columns))
# Calculate the test MSE
print(mean_squared_error(y_test, pred))
```

```
TV           0.046843
radio        0.178544
newspaper    0.002586
dtype: float64
1.9918855518287906
```

Valeur ideale de lambda

In [84]:
```
#
ridgecv = RidgeCV(alphas=alphas, scoring='r2')
ridgecv.fit(X_train, y_train)
ridgecv.alpha_
```

Out[84]:
```
266.8349615603151
```

In [85]:
```
ridge4 = Ridge(alpha=ridgecv.alpha_)
ridge4.fit(X_train, y_train)
mean_squared_error(y_test, ridge4.predict(X_test))
```

Out[85]:
```
2.021697415770955
```

Valeur des coefficients avec valeur ideale de alpha

In [86]:
```
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index=X.columns)
```

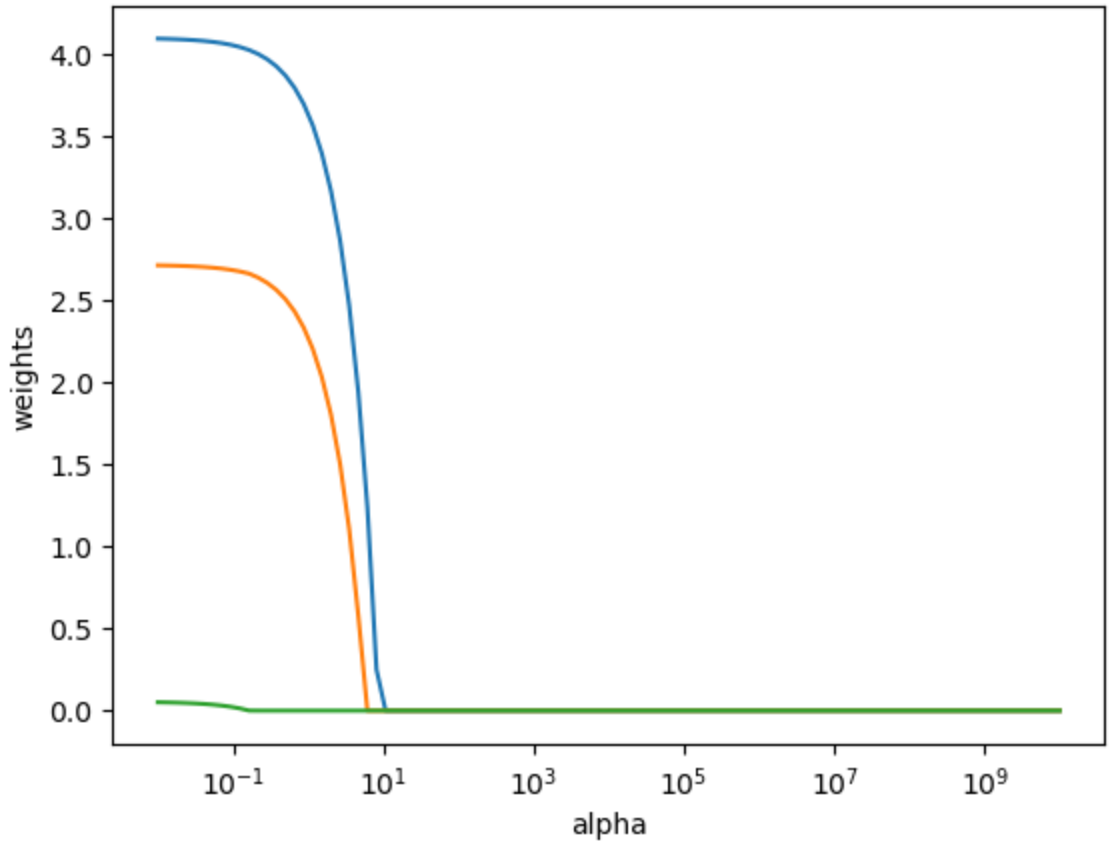
Out[86]:
```
TV           0.045764
```

```
radio        0.187227
newspaper   -0.000721
dtype: float64
```

Lasso

In [87]:
```
lasso = Lasso(max_iter=10000)
coefs = []
for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(scale(X_train), y_train)
    coefs.append(lasso.coef_)
ax = plt.gca()
ax.plot(alphas*2, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

Out[87]:
```
Text(0, 0.5, 'weights')
```



In [88]:
```
lassocv = LassoCV(alphas=None, cv=10, max_iter=100000)
lassocv.fit(X_train, y_train)
lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(X_train, y_train)
mean_squared_error(y_test, lasso.predict(X_test))
```

Out[88]:
```
2.0257820702956333
```

In [89]:
```
#The coefficients
pd.Series(lasso.coef_, index=X.columns)
```

Out[89]:
```
TV           0.046803
radio        0.176151
newspaper    0.001823
dtype: float64
```