Week 9: Regression

M P Gururajan, Hina Gokhale and N N Viswanathan

Department of Metallurgical Engineering & Materials Science
Indian Institute of Technology Bombay

October, 2024

Outline

Linear regression: least squares method

Vegard's law

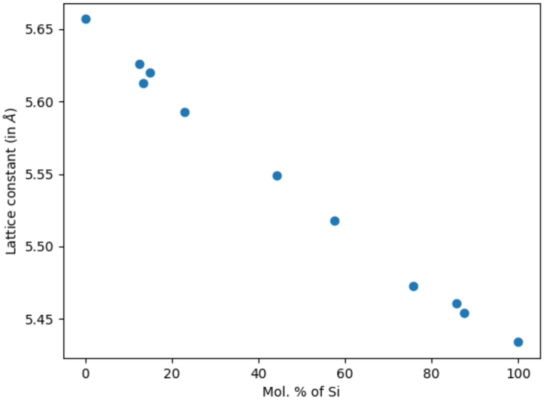

Figure: The variation of lattice parameter with composition in SiGe alloys. Taken from Some properties of Germanium-Silicon alloys, E. R. Johnson and S. M. Christian, Phys. Rev., 95, pp. 560-561, 1954.

Vegard's law

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("SiGeLatticeParameter.csv")
C = data[["Mol. % of silicon"]]
a = data["Lattice constant"]
reg = LinearRegression(fit_intercept=True).fit(C,a)
print(reg.coef_)
print(reg.intercept_)
```

Vegard's law

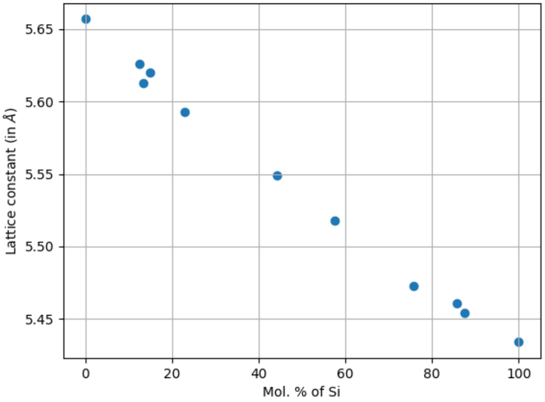

Figure: The fit gives Vegard's law coefficient of 0.0022 (that is, the slope) and the pure Germanium lattice parameter of 5.65 (that is, intercept). These are close enough as is clear from the figure.

Fit and data

```
a_fit = reg.coef_*C + reg.intercept_
plt.scatter(C,a,label="Data")
plt.plot(C,a_fit,color='red',label="Fitted line")
plt.xlabel("Mol. % Si")
plt.ylabel("Lattice constant (in $\AA$)")
plt.legend()
plt.grid()
plt.show()
```

Fit and data: plot

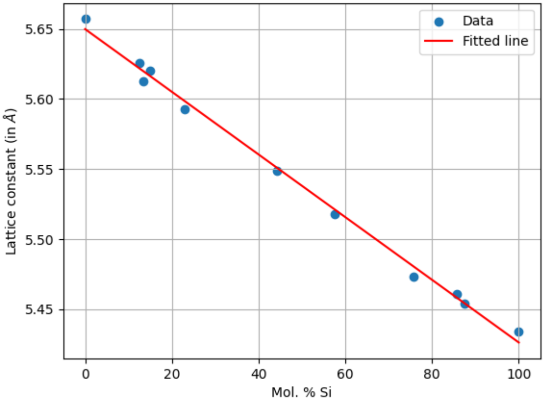

Figure: It is always a good practice to visualise the data and the fit. But, for large and multi-dimensional data set, this is not feasible.

Plotting residuals and printing score

```
reg = LinearRegression(fit_intercept=True).fit(C,a)
print(reg.score(C,a))
a_fit = reg.predict(C)
ResidualPlot = PredictionErrorDisplay(y_true=a,y_pred=a_fit)
ResidualPlot.plot()
plt.show()
```

Plotting residuals

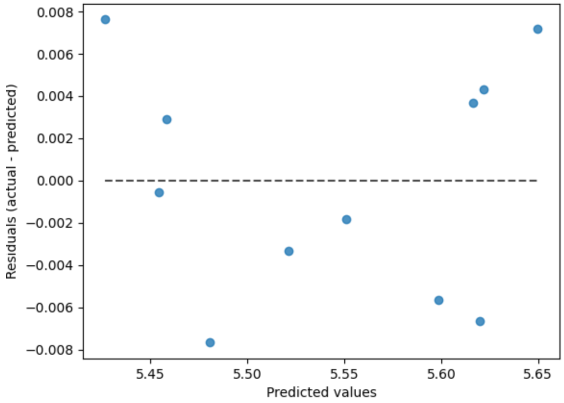

Figure: Plot of residuals. It is always a good practice to visualise the residuals. Do you think the error is normally distributed? The score of regression (coefficient of determination, R^2) is 0.995.

Design matrix for known functional form

Specific heat as a function of temperature

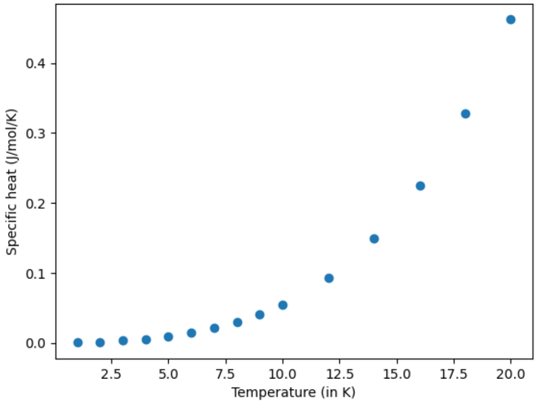

Figure: The variation of specific heat with temperature in copper in the low temperature regime (0 – 20K) – known to fit $C_p = aT + bT^3$. The data is taken from Heat capacity of reference materials: Cu and W, G. K. White and S. J. Collocott, J. Phys. Chem. Ref. Data, 13, pp. 1251-1257, 1984.

Fitting $C_p = aT + bT^3$

```
data = pd.read_csv("SpecificHeat.csv")
T = data["Temperature"]
Cp = data["Specific Heat"]
DM = np.zeros((T.shape[0],2))
for i in range(T.shape[0]):
    DM[i][0] = T[i]
    DM[i][1] = T[i]*T[i]*T[i]
coefficients = np.linalg.inv(DM.T @ DM) @ DM.T @ Cp
Cp_fit = coefficients[0]*T + coefficients[1]*T*T*T
```

Recall: $\hat{\beta} = (X'X)^{-1}X'Y$
Fitted curve: $C_p = -2.246 \times 10^{-4}T + 5.74 \times 10^{-5}T^3$

Low temperature specific heat of copper

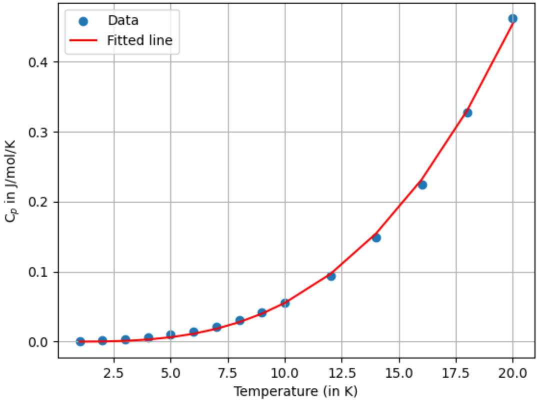

Figure: The fit to $C_p = aT + bT^3$. We obtain $a = -2.246 \times 10^{-4}$ and $b = 5.74 \times 10^{-5}$.

Plot of residuals

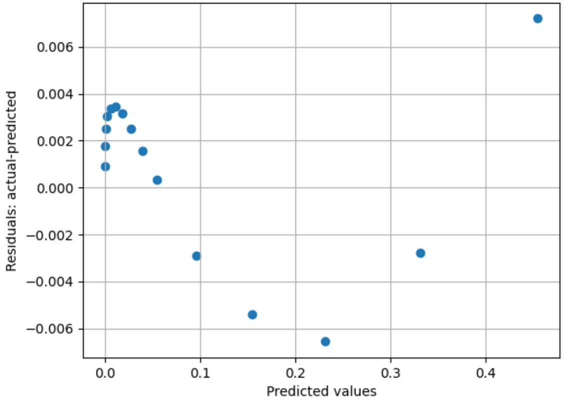

Figure: Plot of residuals. Do you think the error is normally distributed? Note: the coefficient of determination $R^2 = 0.9993$.

Multiple regression

ICI: complete data

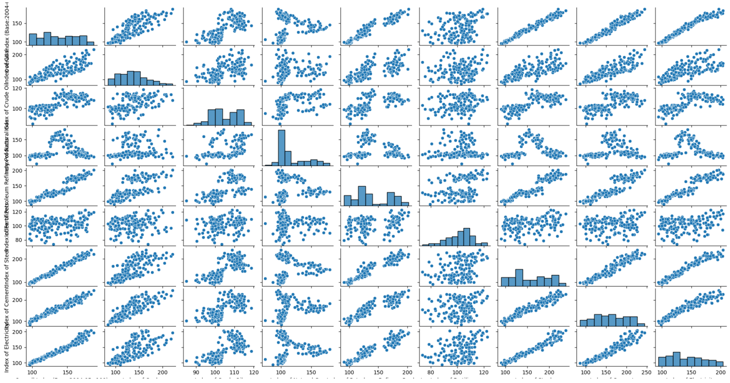

Figure: Is there a linear combination of indices which can be a good indicator of overall index?

ICI: potential for dimensional reduction

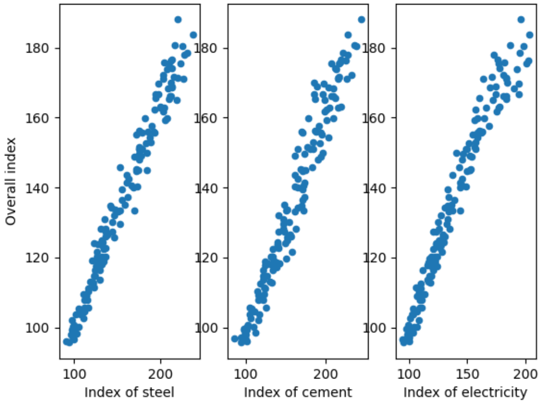

Figure: The steel, cement and electricity indices seem to have almost the same effect on overall index. May be dimensionality reduction is possible? Linearly dependent columns: not good for inversion!!

ICA data: data cleaning, scaling and PCA

```
ICI = pd.read_csv("IndexCoreIndustry.csv")
ICI= ICI[ICI["Months/Years"].str.contains("(Apr-Mar)") ==
False]
ICI= ICI[ICI["Months/Years"].str.contains("(Apr-Jul)") ==
False]
FeaturesData = ICI[["Index of Coal","Index of
Crude Oil","Index of Natural Gas", "Index of
Petroleum Refinery Products","Index of
Fertilizers","Index of Steel","Index of
Cement","Index of Electricity"]]
scaling = MinMaxScaler()
scaling.fit(FeaturesData)
ScaledFeatures = scaling.transform(FeaturesData)
DimRed = PCA()
DimRed.fit(ScaledFeatures)
print(DimRed.explained_variance_ratio_)
```

Reduced dimensions

- Explained variance ratio:
  [0.69755592 0.12190102 0.0889402 0.0508548 0.02435557 0.00939227 0.00424941 0.00275081]
- 4 to 5 features can explain 95.9 to 98.4 % of the variance
- Multi-linear fit in terms of the first four or five variables: sufficient
- Note: these features are from PCA space; not the original features but a linear combination of them
- Note: we also have to split the data into training and test data set!

F-statistic and p-values

```
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LinearRegression
f_statistic, p_values =
f_regression(ScaledFeatures, ScaledResponse)
print(f_statistic)
print(p_values)
```

```
[2.32398669e+02 1.80808479e+02 3.95355751e+00 1.48618442e+03
1.18591675e+01 5.04714109e+03 2.84871550e+03 3.10886899e+03]
[5.37639278e-032 2.50937256e-027 4.86419099e-002 2.01690381e-078
7.49061211e-004 3.94899913e-115 1.13116211e-097 2.58023890e-100]
```

Another way to select features for multiple linear regression!

Multiple linear regression

```
DimRed = PCA(n_components=5)
DimRed.fit(ScaledFeatures)
SRF = DimRed.transform(ScaledFeatures)
#print(SRF)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    SRF, ScaledResponse, test_size=0.4, random_state=0)
from sklearn.linear_model import LinearRegression
mlfit = LinearRegression().fit(X_train,y_train)
print(mlfit.score(X_test,y_test))
```

Fit score

Random state	**Fit score**
0	0.9994
10	0.9992
100	0.9994
255	0.9993
1000	0.9993

How about splitting the data in several ways and checking for fit score in a more methodical way? Cross-validation.

Cross-validation

```
from sklearn.model_selection import cross_val_score
lm = LinearRegression()
scores = cross_val_score(lm,SRF,ScaledResponse,cv=5)
print(scores)
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=10,test_size=0.3,random_state=0)
scores = cross_val_score(lm,SRF,ScaledResponse,cv=cv)
print(scores)
```

```
[0.99353782 0.99561542 0.99400146 0.98249998 0.95971253] [0.99940411
0.9994602 0.99932453 0.99950819 0.99957825 0.99952639 0.99940202
0.99938085 0.99943883 0.99945557]
```

Summary

- Regression: a supervised learning method
- Linear and multi-linear regression
- Design matrix: regression when you know the functional form
- Workflow: EDA, Scaling, PCA, Regression
- Cross-validation: good idea
- Simple cases: plot data and fit, plot residuals
- F-scores for confidence levels or p-values

Thank You!

Questions, clarifications, comments?