Statistical Methods for Data Science

Semester 1 2024

DATA7202

Regression Models

Slava Vaisman

The University of Queensland

r.vaisman@uq.edu.au

1. Linear Regression

2. Model selection
   - Subset Selection
   - Regularization
   - PCA

3. Generalized linear models

4. The Empirical CDF and the Bootstrap method

Regression (1)

- Regression analysis is used to investigate how one variable (a *response* variable) depends on other variables (*explanatory* variables), usually in the mean of the response variable.
- The regression analysis is widely used for prediction and forecasting.
- Specifying models: equations linking the response and explanatory variables.
- Estimating parameters used in the models.
- Checking how well the models fit data.
- Making inferences: calculating confidence intervals and testing hypotheses about the parameters.

Regression (2)

- Let the response variable to be Y, and explanatory variable(s) by $\boldsymbol{x}$.
- The objective of regression analysis is to estimate the expectation of Y for given values of $\boldsymbol{x}$.
- Specifically, we aim to develop a regression function $\mu(\boldsymbol{x})$

$$\mu(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{x}].$$

Regression (3)

- Given a data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, we would like to build a function $g(\boldsymbol{x})$ that predicts the response variable y. Suppose that $y \in \mathbb{R}$.
- Numerous functions are of course possible, but for now, let us consider a (restricted) class of linear functions, namely:

$$\mathcal{H} = \{\boldsymbol{x}^\top \boldsymbol{\beta}\}.$$

- We already know what is the reason for introducing a restricted class of functions (inductive bias).
- In addition, such model is highly interpretable as we saw earlier.

> - The class $\mathcal{H}$ is very structured.
> - Specifically, we need to estimate the set of parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)$ (suppose that $\boldsymbol{x}$ is d-dimensional).
> - Make inference about $\boldsymbol{\beta}$ (confidence intervals), and perform hypothesis testing ($\beta_i = 0$?).

The many faces of regression

- Linear regression: $\mu(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{d} x_j\beta_j$;
- Generalized linear regression: $g(\mu(\boldsymbol{x})) = \beta_0 + \sum_{j=1}^{d} x_j\beta_j$, where $g(\cdot)$ is known and is called a link function;
- Parametric regression: $\mathbb{E}[\mu(\boldsymbol{x})] = h(\boldsymbol{x};\boldsymbol{\theta})$, with $h(\cdot;\cdot)$ is of a known form;
- Nonparametric regression: $\mathbb{E}[\mu(\boldsymbol{x})] = h(\boldsymbol{x})$, with $h(\boldsymbol{x})$ unknown, but satisfies smoothness conditions;
- Semiparametric regression: combines nonparametric and parametric techniques.

Linear models (1)

Given data (x_i, y_i), $i = 1, \ldots, n$, simple linear model is:

$$Y_i = \beta_0 + x_i\beta_1 + \varepsilon_i$$

where

- β_0 and β_1 are unknown parameters;
- x_i is explanatory or independent variable;
- Y_i is dependent or response variable;
- ε_i are random *additive* “errors”.

Linear models (2)

Let

$$\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Thus, $Y_i = \beta_0 + x_i\beta_1 + \varepsilon_i$ for $i = 1, \ldots, n$, can be written as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{1}$$

Moreover, if we consider a d-dimensional data vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^\top$, the multiple linear model

$$Y_i = \beta_0 + \sum_{j=1}^{d} \beta_j x_{ij} + \varepsilon_i,$$

will also take the matrix form (1).

Linear models (3)

Specifically, for $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^\top$:

$$\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & x_1^\top \\ 1 & x_2^\top \\ \vdots & \vdots \\ 1 & x_n^\top \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{pmatrix} \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

and again, $Y_i = \beta_0 + \sum_{j=1}^{d} \beta_j x_{ij} + \varepsilon_i$ for $i = 1, \ldots, n$, can be written as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

Model assumptions

1. Relationship between response and explanatory variables is linear.
2. ε_i's are independent.
3. ε_i's have normal distribution (a strong assumption...);.
4. ε_i's have constant variance σ^2 (homoscedasticity).
5. Usually, number of observations n is larger than number of unknown parameters $p = (d+1)$.

So: $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, where $\mathbf{I}_n$ is $n \times n$ identity matrix . Thus, a more succinct and transparent way of writing $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ is

$$\boldsymbol{Y} \sim \mathrm{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n).$$

We can estimate parameter vector $\boldsymbol{\beta}$ in a general linear model

- using maximum likelihood estimation, or
- least squares.

Maximum likelihood estimation (1)

Since, $\boldsymbol{Y} \sim \mathrm{N}(\boldsymbol{X}\beta, \sigma^2 \mathbf{I}_n)$, the likelihood function is

$$L(\boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-\frac{1}{2}n} \mathrm{e}^{-\frac{1}{2\sigma^2}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})}$$

and thus,

$$\ln L(\boldsymbol{\beta}, \sigma^2) = -\frac{n}{2} ln(2\pi) - \frac{n}{2} ln(\sigma^2) - \frac{1}{2\sigma^2} (\boldsymbol{Y} - \boldsymbol{X}\beta)^\top (\boldsymbol{Y} - \boldsymbol{X}\beta).$$

1. The MLE for $\boldsymbol{\beta}$ is

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}.$$

2. The MLE for σ^2 is

$$\frac{\partial L}{\partial \sigma^2} = 0 \Rightarrow \widehat{\sigma^2} = \frac{1}{n} \left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\beta} \right)^\top \left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\beta} \right).$$

Maximum likelihood estimation (2)

- The MLE of σ^2 is biased, therefore, we can obtain an unbiased estimator via

$$S^2 = \widehat{\sigma^2} = \frac{1}{n-p}\left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right)^\top \left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right).$$

- $\frac{(n-p)S^2}{\sigma^2}$ is distributed χ^2 with $(n-p)$ degrees of freedom $(p = d+1)$.
- $\boldsymbol{X}^\top\boldsymbol{X}$ should be nonsingular or, equivalently, the design matrix $\boldsymbol{X}$ should have full rank p.

It is interesting to note that the linear model can be viewed as first-order approximation of the general model $Y_i = b(\boldsymbol{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$ for $i = 1, \ldots, n$, and $b(\cdot)$ is a function of $\boldsymbol{x}$. Specifically, replace $\boldsymbol{x}$ with Taylor's expansion:

$$\begin{aligned} b(\boldsymbol{x}) &\approx b(\boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla b(\boldsymbol{x}_0) = \underbrace{b(\boldsymbol{x}_0) - \boldsymbol{x}_0^\top \nabla b(\boldsymbol{x}_0)}_{\beta_0} + \boldsymbol{x}^\top \underbrace{\nabla b(\boldsymbol{x}_0)}_{\boldsymbol{\beta}} \\ &= \beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}. \end{aligned}$$

Ordinary least squares (OLS) estimation

- OLS is an alternative to the Maximum likelihood method.
- OLS minimizes the sum of squares of the differences between Y_i's and their expected values.
- Specifically, minimize (with respect to $\boldsymbol{\beta}$) the expression:

$$\sum_{i=1}^{n}(Y_i - X_i^\top \boldsymbol{\beta})^2 = (\boldsymbol{Y} - \boldsymbol{X}\beta)^\top(\boldsymbol{Y} - \boldsymbol{X}\beta)$$

- The optimal solution is:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{Y}.$$

- Note that it is the same solution as the maximum likelihood estimator assuming normal errors with constant variance.

Properties of $\widehat{\boldsymbol{\beta}}$

1. $\mathbb{E}[\widehat{\boldsymbol{\beta}}] = \mathbb{E}\left[(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{Y}\right] = \boldsymbol{\beta}$.
2. $\text{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$.
3. If $\varepsilon_i \overset{iid}{\sim} \text{N}(0, \sigma^2)$, then $\widehat{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\right)$.
4. If ε_i's distribution is non-normal, then $\widehat{\boldsymbol{\beta}}$ has approximate distribution $N\left(\boldsymbol{\beta}, \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\right)$ for large n.

Distribution of $\widehat{\boldsymbol{\beta}}$ is used for statistical inference for $\boldsymbol{\beta}$.

```
==============================================================
          coef    std err   t       P>|t|   [0.025      0.975]
--------------------------------------------------------------
radio     0.0857   0.131   0.656   0.512   -0.171      0.342
tv        4.7897   0.126   38.128  0.000    4.543      5.036
internet 7.0184   0.133   52.641  0.000    6.757      7.280
==============================================================
```

Fitted values and residuals

- **Fitted values**: $\widehat{Y}_i = \widehat{\mu}_i := X_i^\top \widehat{\beta}$

$$\widehat{\boldsymbol{Y}} := \begin{pmatrix} \widehat{Y}_1 \\ \vdots \\ \widehat{Y}_n \end{pmatrix} = \boldsymbol{X}\widehat{\beta} = \underbrace{\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top}_{=:\mathbf{H}(\text{ the hat matrix})} \boldsymbol{Y}$$

- **Residuals**: $e_i = Y_i - \widehat{Y}_i$

$$\mathbf{e} := \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} = \boldsymbol{Y} - \widehat{\boldsymbol{Y}} = (\mathbf{I}_n - \mathbf{H})\boldsymbol{Y}.$$

$$\mathrm{Var}(\mathbf{e}) = (\mathbf{I}_n - \mathbf{H})\left[\mathrm{Var}\left(\boldsymbol{Y}\right)\right](\mathbf{I}_n - \mathbf{H})^\top = \sigma^2(\mathbf{I}_n - \mathbf{H}).$$

We are going to discuss the *hat matrix H* later.

Multiple Outputs

- Suppose we would like to predict multiple outputs $Y_1, Y_2, \ldots, Y_K$ from inputs $X_0, X_1, X_2, \ldots, X_p$.
- Then, we just construct k models:

$$Y_i = \beta_{0,i} + \sum_{j=1}^{p} X_j \beta_{j,k} + \varepsilon_i = f_k(\boldsymbol{X}) + \varepsilon_i, \quad i = 1, \ldots, k.$$

Why OLS is such a powerful idea?

- We saw that the optimal solution for normal linear model is:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}.$$

- We saw that minimizing (with respect to $\boldsymbol{\beta}$) the expression:

$$\sum_{i=1}^{n} (Y_i - X_i^\top \boldsymbol{\beta})^2 = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}),$$

  yields the same solution:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}.$$

- Please note however, that for large datasets, the multiplication and inversion can be quite expensive.
- This brings us to the idea of using other optimization procedures.

Gradient descent (1)

- We want to choose θ so as to minimize some function $J(\theta)$.
- The gradient descent algorithm, which starts with some initial θ, and repeatedly performs the update:

  $$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta),$$

  where the update is simultaneously performed for all values of $j = 0, \ldots, d$.
- The α parameter is called the learning rate.
- This is a very natural algorithm that repeatedly takes a step in the direction of steepest decrease of $J(\theta)$.

Gradient descent (2)

Recall that we are dealing with linear model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

- Set $h(x) = \sum_{i=0}^{d} \beta_i x_i$.
- Given a training set, how do we pick, or learn, the parameters $\beta_0, \ldots, \beta_d$?
- Define a function that measures, for each value of the β's, how close the $h(x^{(i)})$'s are to the corresponding $y^{(i)}$'s via

$$J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_\beta(x^{(i)}) - y^{(i)} \right)^2$$

- This gives rise to the ordinary least squares regression model.

Gradient descent (3)

- For the linear regression, it can be shown that (we omit $1/n$ for convenience):

$$\frac{\partial}{\partial \beta_j} J(\beta) = \sum_{j=0}^{d} (h_\beta(x^{(j)}) - y^{(j)})x^{(j)}.$$

- So, the GD algorithm is:
  Set $\beta_0 = (0, \ldots, 0)^\top$, and repeat until convergence:

$$\beta_{t+1} = \beta_t + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)} \quad \text{for every } j.$$

- This method looks at every example in the entire training set on every step, and is called batch gradient descent.

Stochastic gradient descent (1)

- There is an alternative to batch gradient descent that also works very well.

```
Loop:
    For $i = 1$ to $m$:
        $\beta_{t+1} = \beta_t + \alpha(y^{(i)} - h_\beta(x^{(i)}))x_j^{(i)}$   for every $j$.
```

- In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only.

Stochastic gradient descent (2)

- While batch gradient descent has to scan through the entire training set before taking a single step, a costly operation if n — the number of training points is large
- stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at.
- Often, stochastic gradient descent gets β "close" to the minimum much faster than batch gradient descent.
- Note however that it may never "converge" to the minimum, and the parameters β will keep oscillating around the minimum of $J(\beta)$;
- but in practice most of the values near the minimum will be reasonably good approximations to the true minimum.

It will be generally a good idea to scale the x data-points. Specifically, it is common to set:

$$x = \frac{x - max(x)}{max(x) - min(x)}.$$

Stochastic gradient descent (3)

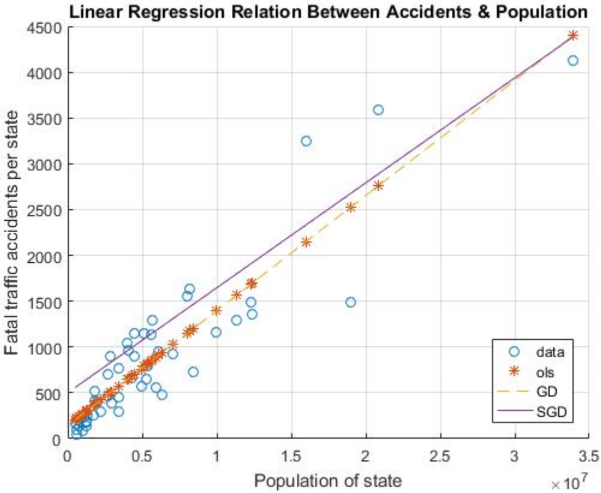


Figure 1: The SGD and GD are very close, but SGD is more manageable, especially for large data.

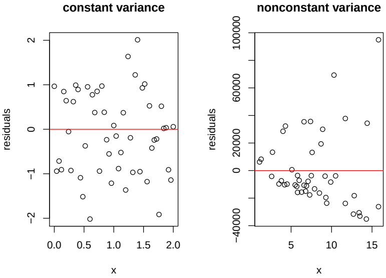
Normal linear model assumptions

- Normal linear model is very attractive since the inference is easy, and we can handle large datasets with SGD.
- However, one should justify the usage of such a model.
- In particular, residuals, e_i's, are important for checking the model assumptions, as they should satisfy the following properties.

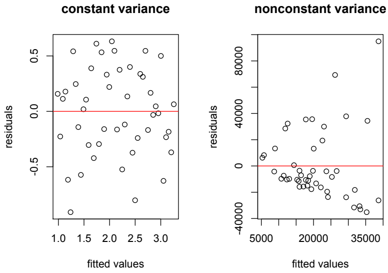
1. **Linearity**. The model should capture all the systematic variance present in the data, leaving nothing but random noise.
2. **Homoscedasticity** (homogeneity of variance). The residuals should have constant variance.
3. **Independence**. The residuals should be independent of each other.
4. **Normality**. The residuals should have a distribution which is approximately normal (with zero mean and constant variance, and be independent).

Linearity and Homoscedasticity

Plot residuals against fitted values and each of the explanatory variables (plus other plots); if the model adequately describes the effect of the explanatory variables, there should be no apparent pattern in the plots.



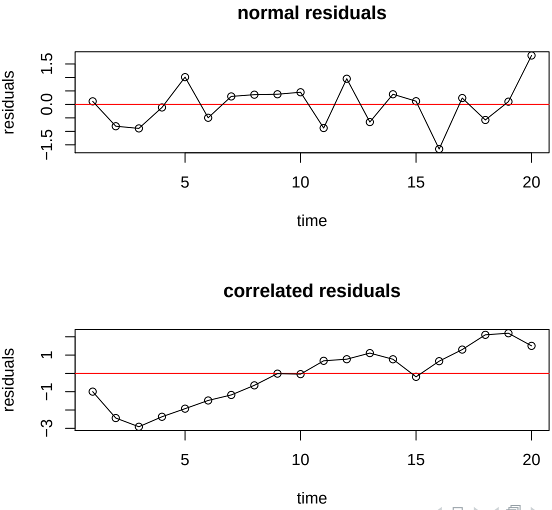
(a) Residuals vs explanatory variable



(b) Residuals vs fitted values

Independence

Plot the residuals using the order in which y_i were measured.

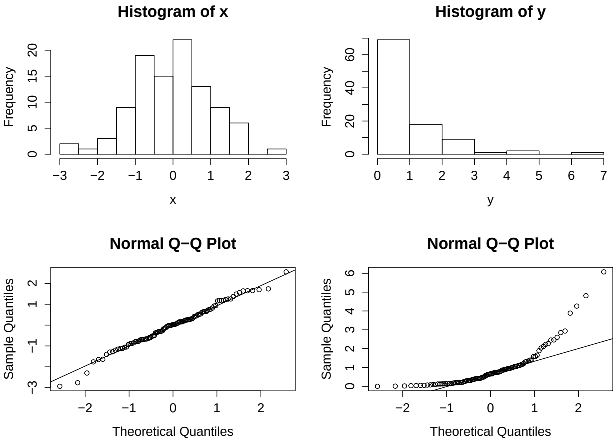


normal residuals

correlated residuals

Residuals normality

We use a histogram or a normal probability plot of the residuals. If we met the normality assumption, the points (standardized residuals) on the qq-plot should be on the straight 45-degree line.



Checking model assumptions — example (1)

`states.csv`

```
State Murder Population Illiteracy Income Frost
Alabama 15.1 3615 2.1 3624 20
Alaska 11.3 365 1.5 6315 152
Arizona 7.8 2212 1.8 4530 15
Arkansas 10.1 2110 1.9 3378 65
California 10.3 21198 1.1 5114 20
Colorado 6.8 2541 0.7 4884 166
....
```

We would like to predict `Murder` based on Population, Illiteracy, Income, and Frost.

Checking model assumptions — example (2)

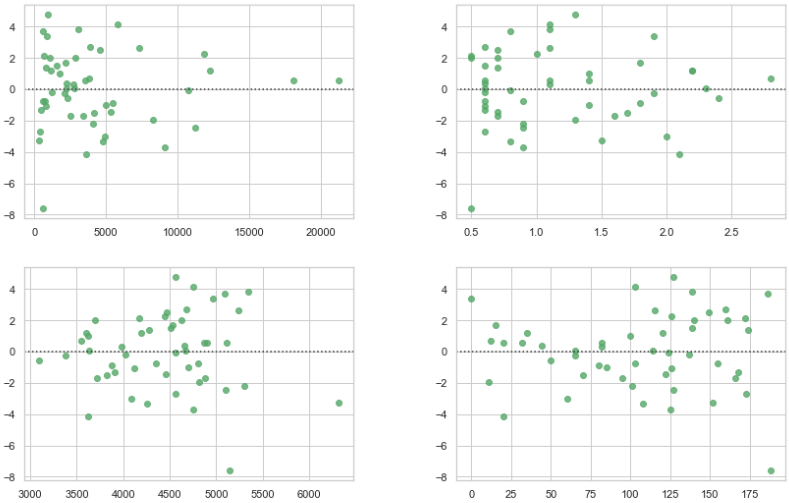
modelassumption.py

```
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import preprocessing
from scipy import stats

sns.set(style="whitegrid")
ds = pd.read_csv("states.csv",index_col=False)
X = ds.loc[:, ds.columns != "Murder"]
X_num = X.loc[:, X.columns != "State"].values
y = ds.loc[:, ds.columns == "Murder"].values

regr = linear_model.LinearRegression()
regr.fit(X_num, y)
y_pred = regr.predict(X_num)

residuals = y_pred - y
```

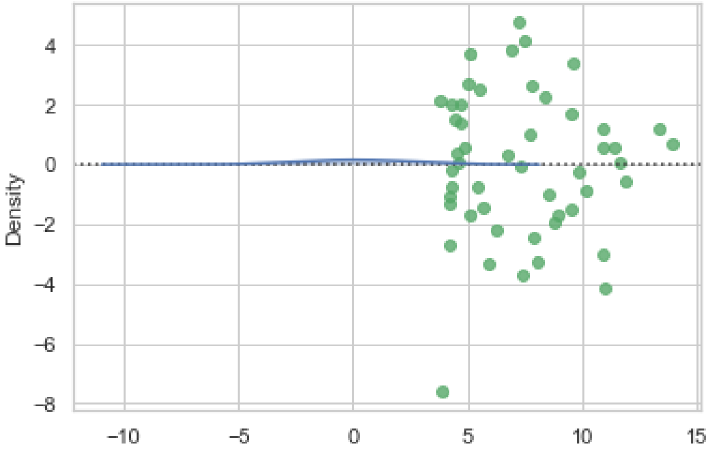
Checking model assumptions — example (3)

```
# plot x vs residuals
for i in range(4):
    sns.residplot(x=X_num[:,i], y=residuals, lowess=False, color="g",label="a")
    plt.show()

# plot predicted vs residuals
sns.residplot(x=y_pred, y=residuals, lowess=False, color="g")

# checking normality
sns.distplot(residuals)
plt.show()
fig = sm.qqplot(residuals[:,0])
plt.show()
```

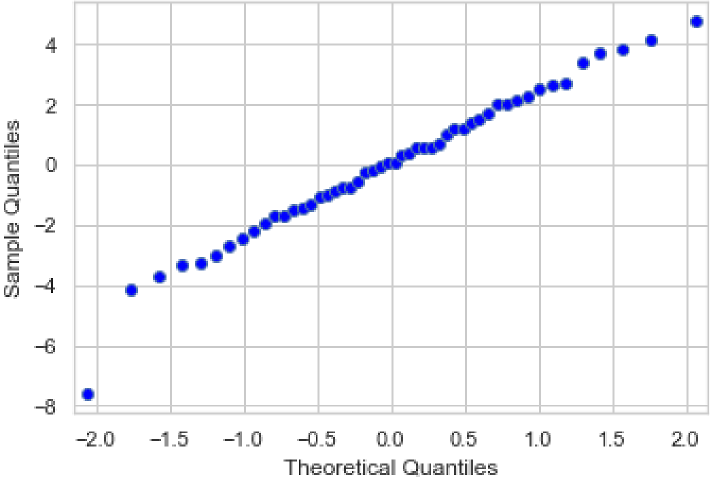
Checking model assumptions — example (4) explanatory vs residuals



Checking model assumptions — example (5) predicted vs residuals



Checking model assumptions — example (6) checking normality



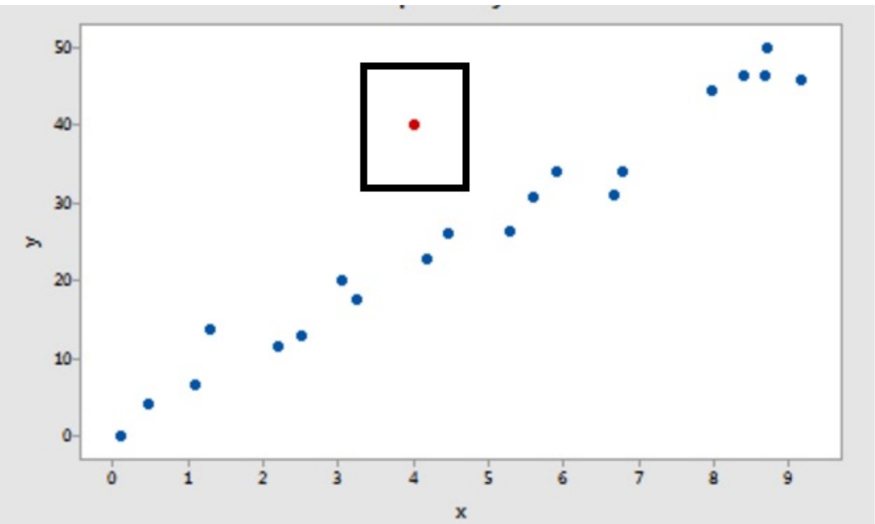
Non-constant variance

- If you'd violated the assumption, you'd expect to see a nonhorizontal line.
- The suggested power transformation in listing is the suggested power p (y^p) that would stabilize the nonconstant error variance.
- For example, if the plot showed a nonhorizontal trend (similar to square root), then using $\sqrt{y}$ rather than Y in the regression equation might lead to a model that satisfies homoscedasticity.

Unusual observations (1)

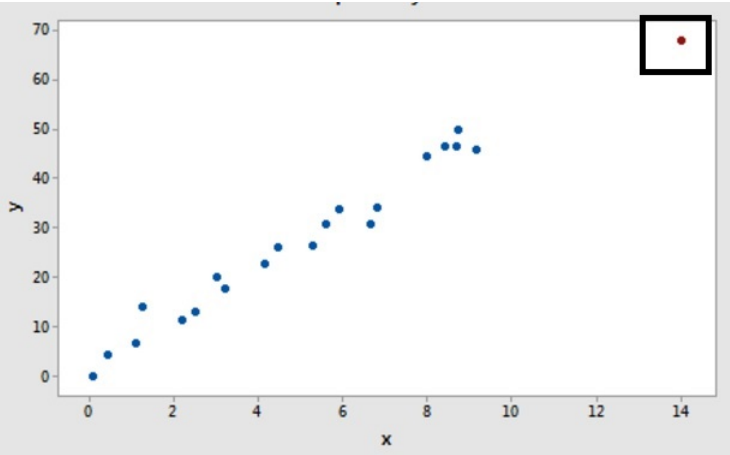
- Outliers are observations that are not been predicted in a good way by the proposed model.
- They have higher residual values $e_i = Y_i - \widehat{Y}_i$.
- $e_i > 0$ indicate that the model underestimates the response value.
- $e_i < 0$ indicate that the model overestimates the response value.

Unusual observations (2)



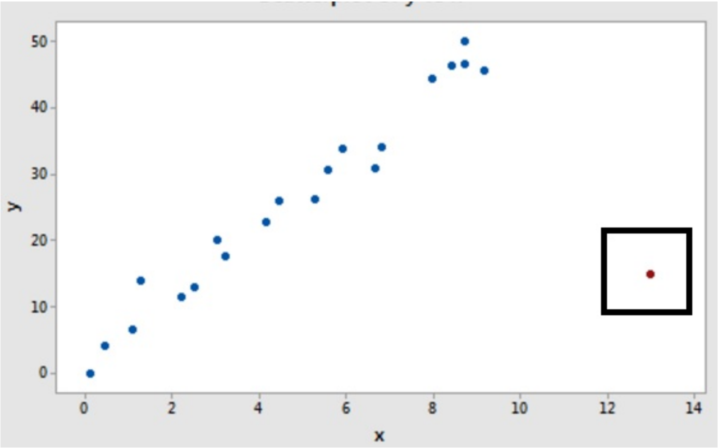
Unusual observations (3)

A data point has high leverage if it has "extreme" predictor x values. Recall that the response value is not important for determining the leverage.



Unusual observations (4)

Sometimes, the high leverage point can influence the regression model! In this case, it is also an outlier! This point is *Influential*, in the sense that if removed, it would heavily change the statistical analysis.



The hat matrix and the leverage

- $\mathbf{H} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top$ is called the hat matrix.
- $\mathbf{H}$ is symmetric and idempotent (idempotent matrix is a matrix which, when multiplied by itself, yields itself.).
- For each $1 \leq i \leq n$, the i-th element, h_{ii} on the diagonal of $\mathbf{H}$, is called the leverage of the i-th observation.
- An observation with high leverage can make a substantial difference to the fit of the model (may be an outlier or influential observation).
- A rule of thumb: it may be a concern if $h_{ii} > 2p/n$.
- **Standardized residuals**:

$$r_i = \frac{e_i}{\widehat{\sigma}(1 - h_{ii})},$$

  where h_{ii} is the leverage of the i-th observation.
- **Studentized residuals**:

$$r_i^* = \frac{e_i}{\widehat{\sigma}_{(-i)}(1 - h_{ii})},$$

  where $\widehat{\sigma}_{(-i)}$ is the estimator of σ after deleting the i-th observation.

Outliers and Influential Observations

- Leverages h_{ii}: Observations with $h_{ii} > \frac{2p}{n}$ have extreme $\boldsymbol{x}$ values: investigate to see if influential.
- Studentized residual: Observations with large absolute value (eg $|r_i^*| > 2$): investigate to see if outlier.
- DFFITS: $DFFITS_i = r_i^* \left(\frac{h_{ii}}{1-h_{ii}}\right)^{1/2}$; A rule of thumb: the i-th observation is influential if $|DFFITS_i| > 2\sqrt{\frac{p}{n}}$.
- Additional measures: DFBETA and Cook's distance.

High leverage points are harder to spot when there are multiple covariates.

Outliers and Influential Observations — example (1)

We consider the rat data and the corresponding unusualdata.rat.py code.

```
   BodyWt LiverWt Dose    y
1  176    6.5     0.88    0.42
2  176    9.5     0.88    0.25
3  190    9       1       0.56
4  176    8.9     0.88    0.23
5  200    7.2     1       0.23
6  167    8.9     0.83    0.32
...
```

Outliers and Influential Observations — example (2)

```
"""
# unusualdata.rat.py
"""
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy import stats

ds = pd.read_csv("rat.csv",index_col=0)
pd.plotting.scatter_matrix(ds, alpha=1.0)
```

Outliers and Influential Observations — example (3)

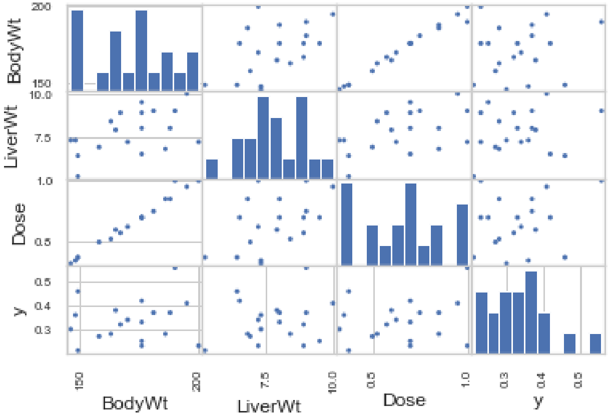


Figure 2: The rat dataset — we cannot see any apparent problems.

Outliers and Influential Observations — example (4)

```
# fit regression
import statsmodels.formula.api as smf
results = smf.ols(’y ˜ BodyWt + LiverWt+Dose’, data=ds).fit()
print(results.summary())

sns.residplot(x=ds.BodyWt, y=results.resid, lowess=False, color="g")
sns.residplot(x=ds.LiverWt, y=results.resid, lowess=False, color="g")
sns.residplot(x=ds.Dose, y=results.resid, lowess=False, color="g")


# residuals histogram
sns.distplot(results.resid)

from statsmodels.graphics.regressionplots import plot_leverage_resid2
fig, ax = plt.subplots(figsize=(8,6))
fig = plot_leverage_resid2(results, ax = ax)
```

Outliers and Influential Observations — example (5)

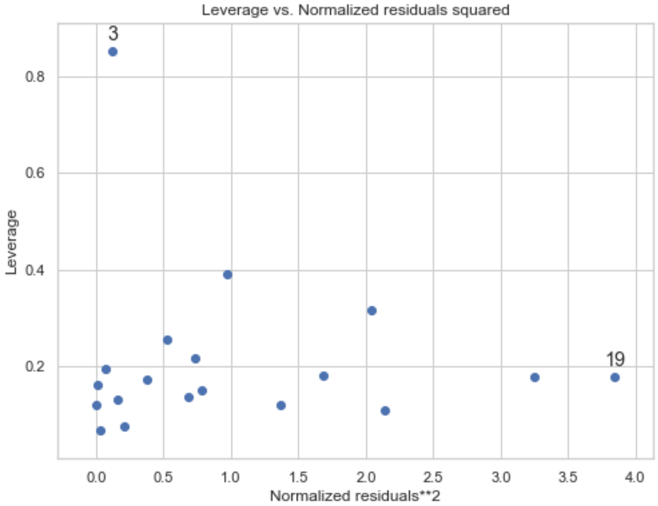


Figure 3: The rat dataset — observation #3 is problematic.

Outliers and Influential Observations — example (6)

```
   BodyWt LiverWt Dose    y
1  176    6.5     0.88    0.42
2  176    9.5     0.88    0.25
3  190    9       1       0.56
4  176    8.9     0.88    0.23
5  200    7.2     1       0.23
6  167    8.9     0.83    0.32
...
```

Careful inspection of the data shows that observation 3 had a body weight of 190g but was given the full dosage of the drug instead of following the protocol of dosage being proportional to body weight.

Outliers and Influential Observations — example (7)

We compare the original regression results with the results obtained after removing the third observation.

```
###############################################################
# removing the third observation
ds2 = ds[ds.index !=3]
results2 = smf.ols(’y ~ BodyWt + LiverWt+Dose’, data=ds2).fit()
print(results2.summary())

print(results.mse_model)
print(results2.mse_model)
```

The mean squared error of both models are:

1. Original model: 0.01708, and
2. After deleting observation 3: 0.00061.

Corrective measures

- It is important to remember that an outlier or influential observation cannot simply be erased.
- Unless the observation is clearly in error, then most that can be done is to report the results both with and without the outlier or influential observation.
- Also, not all outliers are bad. As Weisberg (2005) puts it, for example, a geologist searching for oil deposits may be looking for outliers, if the oil is in the places where a fitted model does not match the data".
- What we may deem as "outliers" are also relative to the underlying model that we are using. If we modify our model, then what we might consider an "outlier" might change.

Corrective measures — deleting observations

- Deleting influential observations and outliers can cause an improvement of a dataset's fit to the normality assumption.
- It will be generally a good idea to delete an observation, if we can determine that this observation is a data error or the experiment protocol failure.
- It is important to remember that outliers can be very interesting in the sense that they can provide an important insight about our data. Namely, maybe our model is false.

Corrective measures – variable transformation

- When our models doe not meet the linear regression assumptions, we can try to transform one or more variables.
- A variable X will be generally transformed to be X^{λ}.
- If a variable is a proportion, a logit transformation

$$\ln\left(\frac{X}{1-X}\right)$$

  can be used.
- Some transformations:

$$X^{-2}, \quad X^{-0.5}, \quad X^{0.5}, \quad X^{2}, \quad \log(X), \quad \cdots$$

- One should use transformations with care since it can make the result interpretation to be problematic. Specifically, how do we interpret the relationship between the frequency of employee complaints and the cube root of logarithm of employee's privileges.

Corrective measures — adding or deleting variables

- Changing the variables in our model will cause an impact on the fit of the model.
- Many times, adding an (important) variable will correct many of the model's problems.
- Deleting a variable that is not important for the model, can also improve the model fit.

The Bias - variance decomposition of squared error

Recall that

$$\mathbb{E}\left[(Y - g(\boldsymbol{X}))^2\right] = (\mathrm{Bias} g(\boldsymbol{X}))^2 + \mathrm{Var}(g(\boldsymbol{X})) + \sigma^2,$$

where

$$\mathrm{Bias}[g(\boldsymbol{X})] = \mathbb{E}\left[g(\boldsymbol{X}) - Y\right],$$

$$\mathrm{Var}(g(\boldsymbol{X})) = \mathbb{E}\left[(g(\boldsymbol{X}) - \mathbb{E}\left[g(\boldsymbol{X})\right])^2\right],$$

and σ^2 is the variance of the random error term.

- Note that the expectation is taken with respect to training sets $(\boldsymbol{X}, Y)$.
- The σ^2 term is irreducible.

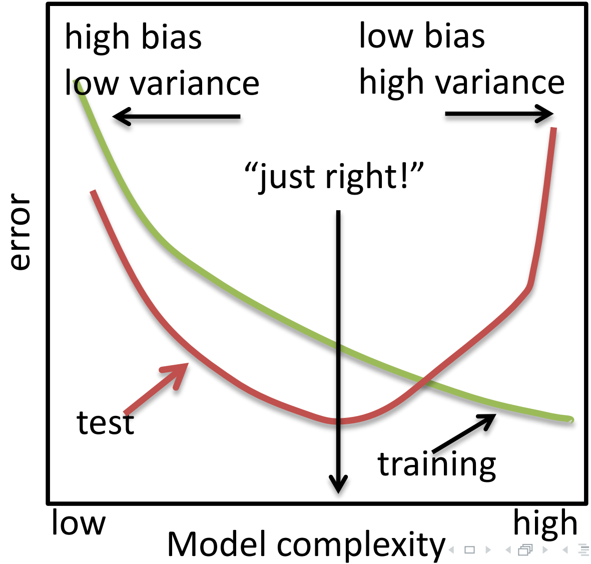
The bias and the variance

- The bias term $\mathrm{Bias}[g(\boldsymbol{X})] = \mathbb{E}\left[g(\boldsymbol{X}) - Y\right]$ measures the error caused by the simplifying assumptions built into the method.
- For example, if we model a polynomial of degree 7 with a polynomial of degree 1, we will encounter a high bias.

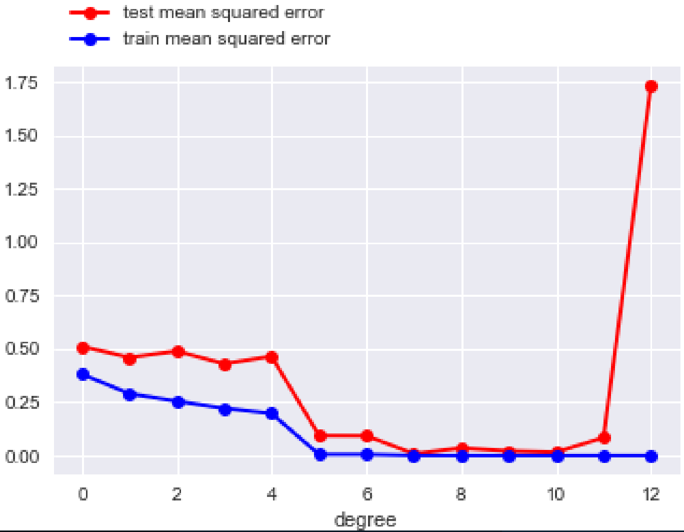
- The variance term $\mathrm{Var}(g(\boldsymbol{X})) = \mathbb{E}\left[\left(g(\boldsymbol{X}) - \mathbb{E}\left[g(\boldsymbol{X})\right]\right)^2\right]$, tells us about how (much) the model $g(\boldsymbol{x})$ will change if we will estimate it using a different dataset.
- For example, suppose that we have a dataset of with 10 points. Using a polynomial of degree 10, we can capture all the points. However, for a different dataset, the model will fail to generalize.

The bias and the variance problems are called under and over fitting. respectively. Our main objective is to find a model that will have both low bias and low variance.

A general overview on bias-variance trade-of



Compare with our polynomial regression train vs. test errors



Cross-Validation Alternatives

- Akaike Information Criterion (AIC)

$$\mathrm{AIC} = -\underbrace{2 \ln f(y \mid \hat{\theta})}_{\text{goodness of fit}} + \underbrace{2p}_{\text{model complexity}},$$

  where p is the number of parameters and $f(y \mid \hat{\theta})$ is the likelihood.

- Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = -\underbrace{2 \ln f(y \mid \hat{\theta})}_{\text{goodness of fit}} + \underbrace{p \ln n}_{\text{model complexity}},$$

  where n is the number of observations.

The model that minimizes AIC or BIC should be preferred.

What AIC and BIC actually measure?

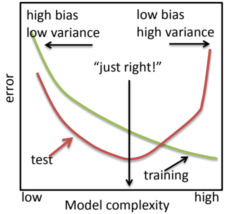
- They estimates a model's **generalizability** — the ability to fit unseen samples from the same process, not just the current data sample.

- When we say generalizability, we mean:

  1. "closeness" to the observed process, and
  2. predictive accuracy

- We next fir two models for the stockmarket data and examine the corresponding AIC and BIC. The results are as in `stockaicbic.py`.

The bias-variance trade-of (again)



Id	variable 1	variable 2	$\cdots$	variable i	$\cdots$	variable n
1	.	.	.	.	.	.
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: Data representation by the table

An idea — control the model complexity by variable elimination.

The problem

We will consider the linear model $Y = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p + \varepsilon$.

1. Prediction Accuracy.
   - Provided that the relationship between the response and the predictors is (approximately) linear, the least squares estimates will have low bias.
   - If the number of samples n satisfies $n \gg p$ we will also have low variance.
   - If n is not much larger than p — variance grows.
   - If $n < p$ — no unique least square estimate and the variance is infinite.
2. Model Interpretability.
   - Often, some or many of the variables used in a multiple regression model are not associated with the response.
   - Including such variables leads to unnecessary complexity.
   - By removing these variables (by setting the corresponding coefficient estimates to zero), we can obtain a model that is more easily interpreted.

What can be done?

- *Subset Selection*. Identify a subset of the p predictors that is related to the response. Then, fit a model using the reduced set of variables.
- *Shrinkage*. This approach fit a model involving all (p) the predictors, but, the estimated coefficients are shrunken towards zero.
- *Dimension Reduction*. Here, we construct a projection of all the p predictors into a K-dimensional subspace, where $K < p$ (or better $K \ll p$).

Best Subset Selection

- To perform best subset selection, we fit a separate least squares regression best subset for each possible combination of the p predictors.
- That is, we fit all p models selection that contain exactly one predictor

$$\binom{p}{1} \text{ models.}$$

- Next, we fit all p models selection that contain exactly two predictor

$$\binom{p}{2} \text{ models.}$$

- Overall, we will need to consider 2^p possibilities.

Best Subset Selection Algorithm

1. Let M_0 denote the null model , which contains no predictors. This model simply predicts the sample mean for each observation.
2. For k $= 1, 2, \ldots, p$:
    1. Fit all $\binom{p}{k}$ models that contain exactly k predictors.
    2. Pick the best among these $\binom{p}{k}$ models, and call it M_k. Here best is defined as having the smallest residual square error.
3. Select a single best model from among $M_0, \ldots, M_p$ using crossvalidated prediction error, or AIC, BIC, e.t.c.

Due to computational costs, the best subset selection cannot be applied with very large p.

Forward stepwise selection

1. Let M_0 denote the null model , which contains no predictors. This model simply predicts the sample mean for each observation.
2. For k $= 1, 2, \ldots, p$:
    1. Consider all $p - k$ models that augment the predictors in M_k with one additional predictor.
    2. Choose the best among these $p - k$ models, and call it M_{k+1}. Here best is defined as having smallest RSS.
3. Select a single best model from among $M_0, \ldots, M_p$ using crossvalidated prediction error, or AIC, BIC, e.t.c.

The forward stepwise selection gives a great computational value. Note that at each step, we perform $p - k$ fits, so the overall complexity is

$$\sum_{k=0}^{p-1} p - k = \mathcal{O}(p^2) \quad \ll \quad \underbrace{\mathcal{O}(2^p)}_{\text{best subset selection}} \quad .$$

Backward stepwise selection

1. Let M_p denote the full model, which contains all predictors.
2. For k $= p, p-1, \ldots, 1$:
    1. Consider all k models that contain all but one of the predictors in M_k, for a total of $k-1$ predictors.
    2. Choose the best among these k models, and call it M_{k-1}. Here best is defined as having smallest RSS.
3. Select a single best model from among $M_0, \ldots, M_p$ using crossvalidated prediction error, or AIC, BIC, e.t.c.

Stock-market example

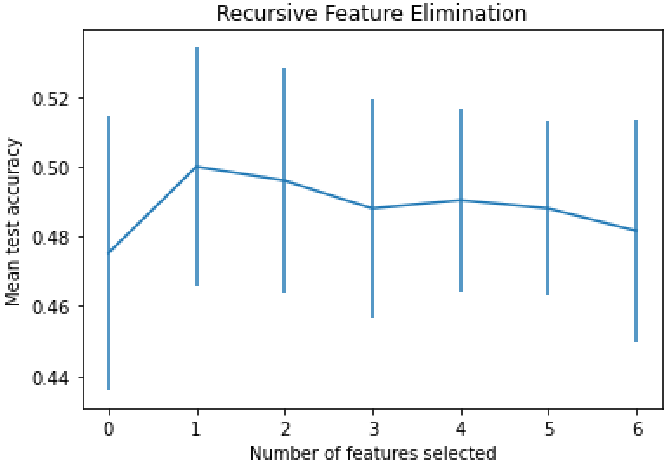
```
Year Lag1 Lag2 Lag3 Lag4 Lag5 Volume Today Direction
1 2001 0.381 -0.192 -2.624 -1.055 5.01 1.1913 0.959 Up
2 2001 0.959 0.381 -0.192 -2.624 -1.055 1.2965 1.032 Up
3 2001 1.032 0.959 0.381 -0.192 -2.624 1.4112 -0.623 Down
4 2001 -0.623 1.032 0.959 0.381 -0.192 1.276 0.614 Up
....
```

Stock-market example

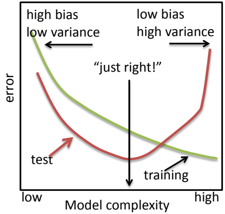
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFECV
from sklearn.model_selection import KFold
df = pd.read_csv("stock.csv",index_col=0)
df.Direction = df.Direction.astype(’category’)
df["DirectionNum"] = df.Year
df.loc[df[’Direction’] =="Up", ’DirectionNum’] = 1
df.loc[df[’Direction’] =="Down", ’DirectionNum’] = 0
# construct X,Y
Y = df.DirectionNum
X = df.drop(["Direction","DirectionNum","Today"],axis=1)
logreg = LogisticRegression(solver=’lbfgs’)
rfecv = RFECV(estimator=logreg, step=1, cv=KFold(5), scoring=’accuracy’)
rfecv.fit(X, Y)
print("Optimal number of features : %d" % rfecv.n_features_)
n_scores = len(rfecv.cv_results_["mean_test_score"])
plt.xlabel("Number of features selected")
plt.ylabel("Mean test accuracy")
plt.errorbar(
    range(0, n_scores ),
    rfecv.cv_results_["mean_test_score"],
    yerr=rfecv.cv_results_["std_test_score"],
)
plt.title("Recursive Feature Elimination")
plt.show()
# we can see that Lag1 + Volume are the best features
X_new = rfecv.transform(X)
```

Stock-market — Greedy feature selection in Python (stockfselect.py)

The procedure finds that 2 features should be used, Lag1 and Volume.



The bias-variance trade-of (again)



- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.
- Shrinking the coefficient estimates can significantly reduce their variance.
- The two best-known techniques for shrinking the regression coefficients towards zero are ridge regression and the lasso.

Ridge Regression (1)

- Recall that the least squares fitting procedure estimates

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- Ridge regression is very similar to least squares. Specifically, the ridge regression coefficient estimates β^R are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 ,$$

  where $\lambda \geq 0$ is a tuning parameter, that will be determined separately.
- The second term $\lambda \sum_{j=1}^{p} \beta_j^2$ called a shrinkage penalty, is small when the coefficients are close to zero, and so it has the effect of shrinking penalty the estimates of β_j towards zero.

Ridge Regression (2)

- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates.
- When $\lambda = 0$, the penalty term has no effect, and ridge regression will produce the least squares estimates.
- When $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero.
- Selecting a good value for λ is critical; this can be obtained via cross-validation.
- A very nice property of the ridge regression — there exists the following analytical solution:

$$\hat{\beta}^R = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda I)^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

The Lasso

- The lasso is a shrinkage method like ridge, with subtle but important differences — the L_2 penalty $\lambda \sum_{j=1}^{p} \beta_j^2$, is replaced by L_1 penalty $\lambda \sum_{j=1}^{p} |\beta_j|$.
- The lasso penalty makes the solutions nonlinear in the y_i, and there is no closed form expression as in ridge regression.
- The lasso estimate is defined by:

$$\hat{\beta}^{l} = \mathrm{argmin}_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the L_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.

Shrinkage example (shrinkage.py) (1)

```
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn import datasets, linear_model
import matplotlib.pyplot as plt
from sklearn.datasets import make_friedman1


n_points = 80 # points
X, Y =  make_friedman1(n_samples=n_points, n_features=5,
                       noise=0.5, random_state=100)

linreg = linear_model.LinearRegression()
linreg.fit(X,Y)
linreg.intercept_
linreg.coef_
>>>  6.94626636, 6.2524142 , 2.56615609, 9.08680857, 4.81193082
```

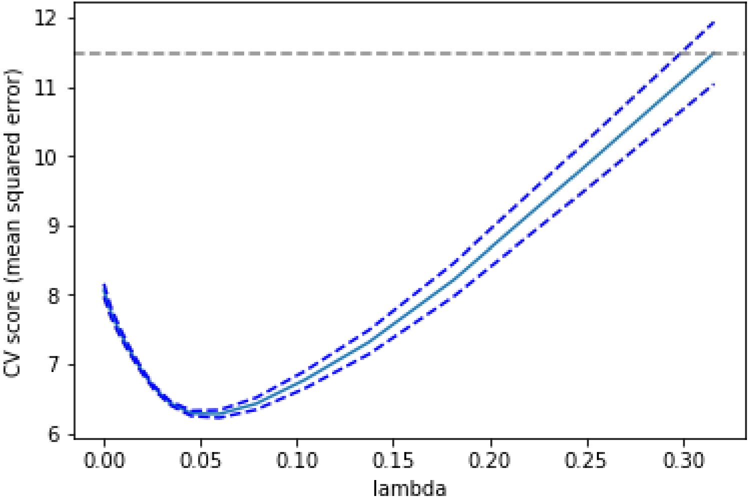
Shrinkage example (shrinkage.py) (2)

```
# add some redundant features
X_r = np.random.uniform(0,1,(n_points,14))
df_r = pd.DataFrame(X_r)

X = pd.DataFrame(X)
X =  pd.concat([X,df_r], axis=1)

scores = list()
scores_std = list()

lasso = linear_model.Lasso(fit_intercept=True)
alphas = np.logspace(-4, -.5, 30)

for alpha in alphas:
     lasso.alpha = alpha
     this_scores = -cross_val_score(lasso, X, Y,scoring=
          ’neg_mean_squared_error’,cv=5)
     scores.append(np.mean(this_scores))
     scores_std.append(np.std(this_scores))
```

Shrinkage example (shrinkage.py) (3)

```
plt.plot(alphas, scores)
# plot error lines showing +/- std. errors of the scores
plt.plot(alphas, np.array(scores) + np.array(scores_std) / np.sqrt(len(X)),
             'b--')
plt.plot(alphas, np.array(scores) - np.array(scores_std) / np.sqrt(len(X)),
             'b--')
plt.ylabel('CV score (mean squared error)')
plt.xlabel('lambda')
plt.axhline(np.max(scores), linestyle='--', color='.5')

lasso = linear_model.Lasso(fit_intercept=True,alpha=0.9)
lasso.fit(X,Y)
lasso.coef_
lasso.intercept_
>>> lasso.coef_
Out[11]:
array([ 0.        ,  0.        ,  0.        ,  1.15602723,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -0.        ,
        0.        ,  0.        ,  0.        , -0.        ,  0.        ,
        0.        ,  0.        , -0.        ,  0.        ])
```

The Lasso CV example — (shrinkage.py)



Understanding the PCA – the general setting

Our $m \times n$ data matrix X is given by

$$X = \begin{pmatrix} X_{1,1} & \dots & \dots & X_{1,n} \\ X_{2,1} & \dots & \dots & X_{2,n} \\ \vdots & \dots & \dots & \vdots \\ X_{m,1} & \dots & \dots & X_{m,n} \end{pmatrix},$$

where

- m — the number of measurement,
- n — the number of observations.

Note that, each data sample is a column vector of X. Each sample is in m-dimensional space.

An important assumption

Our data comes from multivariate normal distribution.

Understanding the PCA – our objective

$$X = \begin{pmatrix} X_{1,1} & \dots & \dots & X_{1,n} \\ X_{2,1} & \dots & \dots & X_{2,n} \\ \vdots & \dots & \dots & \vdots \\ X_{m,1} & \dots & \dots & X_{m,n} \end{pmatrix}$$

Redundancy: It might happen, that our system has $k << m$ degrees of freedom (the number of independent ways by which a dynamic system can move), but is taking the entire $m-$**dimensional** space in our original data set X.

Data Redundancy

1. Essentially, we would like to know if the rows of X are correlated.
2. If they are, we might be able to perform the desired dimensionality reduction. **Namely, we would like to remove the redundancy!**

A reminder: The Variance and the Covariance

First, let us formalize the concept of **redundancy**.
Suppose that we are given 2 data vectors (let us suppose that their mean is zero)

$$\boldsymbol{x} = (x_1, \ldots, x_n), \quad \boldsymbol{y} = (y_1, \ldots, y_n).$$

Then, the variance is given by (inner product):

$$\sigma_{\boldsymbol{x}}^2 = \frac{1}{n-1}\sum_{i=1}^{n} x_i \cdot x_i = \frac{1}{n-1}\boldsymbol{x} \cdot \boldsymbol{x}^T, \quad \sigma_{\boldsymbol{y}}^2 = \frac{1}{n-1}\boldsymbol{y} \cdot \boldsymbol{y}^T.$$

The covariance is measuring the statistical relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\sigma_{\boldsymbol{xy}}^2 = \frac{1}{n-1}\boldsymbol{x} \cdot \boldsymbol{y}^T = \frac{1}{n-1}\boldsymbol{y} \cdot \boldsymbol{x}^T = \sigma_{\boldsymbol{yx}}^2.$$

If I observe $\boldsymbol{x}$, can I say something about $\boldsymbol{y}$?

Things to remember about the covariance

$$\begin{cases} \sigma^2_{\boldsymbol{xy}} \approx 0 \;\Rightarrow\; \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ are (almost) statistically independent} \\ \sigma^2_{\boldsymbol{xy}} \neq 0 \;\Rightarrow\; \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ share some information } \;\Rightarrow\; \text{REDUNDANCY} \\ \sigma^2_{\boldsymbol{xy}} = \sigma^2_{\boldsymbol{yx}} \end{cases}$$

Constructing a covariance matrix from our data

Recall that X is given by

$$X = \begin{pmatrix} X_{1,1} & \dots & \dots & X_{1,n} \\ X_{2,1} & \dots & \dots & X_{2,n} \\ \vdots & \dots & \dots & \vdots \\ X_{m,1} & \dots & \dots & X_{m,n} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_m \end{pmatrix}.$$

The covariance matrix is given by $C_X = \frac{1}{n-1} X\, X^T$. In particular,

$$C_X = \begin{pmatrix} \sigma^2_{\boldsymbol{X}_1} & \sigma^2_{\boldsymbol{X}_1 \boldsymbol{X}_2} & \dots & \sigma^2_{\boldsymbol{X}_1 \boldsymbol{X}_m} \\ \sigma^2_{\boldsymbol{X}_2 \boldsymbol{X}_1} & \sigma^2_{\boldsymbol{X}_2} & \dots & \sigma^2_{\boldsymbol{X}_2 \boldsymbol{X}_m} \\ \vdots & \dots & \dots & \vdots \\ \sigma^2_{\boldsymbol{X}_m \boldsymbol{X}_1} & \dots & \dots & \sigma^2_{\boldsymbol{X}_m} \end{pmatrix}.$$

Properties of covariance matrix

$$C_X = \begin{pmatrix} \sigma^2_{\boldsymbol{X}_1} & \sigma^2_{\boldsymbol{X}_1\boldsymbol{X}_2} & \cdots & \sigma^2_{\boldsymbol{X}_1\boldsymbol{X}_m} \\ \sigma^2_{\boldsymbol{X}_2\boldsymbol{X}_1} & \sigma^2_{\boldsymbol{X}_2} & \cdots & \sigma^2_{\boldsymbol{X}_2\boldsymbol{X}_m} \\ \vdots & \cdots & \cdots & \vdots \\ \sigma^2_{\boldsymbol{X}_m\boldsymbol{X}_1} & \cdots & \cdots & \sigma^2_{\boldsymbol{X}_m} \end{pmatrix}.$$

- The diagonal elements are the variances of the rows of our data matrix X.
- The off-diagonal elements are the corresponding covariances

$$C_X(i,j) = \sigma^2_{\boldsymbol{X}_i\boldsymbol{X}_j} = \sigma^2_{\boldsymbol{X}_j\boldsymbol{X}_i} = C_X(j,i).$$

- $C_X = X\,X^T$ is **symmetric**.
- Intuitively: small off-diagonal entries ⇒ statistical independence.
- Intuitively: not so small off-diagonal entries ⇒ REDUNDANCY!

Intuition

Non-zero off-diagonal entries $\Rightarrow$ REDUNDANCY!

So, what do we want to achieve?

We want the covariance matrix to look like this, why?

$$C_X = \begin{pmatrix} \sigma^2_{\mathbf{X}_1} & 0 & \dots & 0 \\ 0 & \sigma^2_{\mathbf{X}_2} & \dots & 0 \\ \vdots & \dots & \dots & \vdots \\ 0 & \dots & 0 & \sigma^2_{\mathbf{X}_m} \end{pmatrix}.$$

Because, in this case we will have

$$\textbf{NO CORRELATION} = \textbf{NO REDUNDANCY!}$$

What is this? DIAGONALIZATION!

DIAGONALIZATION OF C_X = NO REDUNDANCY

More intuition

- Basically, we want to find a new way to look at my system (change of basis – linear transformation), such that C_X becomes diagonal.
- Suppose that we achieved the desired diagonalization:

$$C_X = \begin{pmatrix} \sigma^2_{\boldsymbol{X}_1} & 0 & \dots & 0 \\ 0 & \sigma^2_{\boldsymbol{X}_2} & \dots & 0 \\ \vdots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \sigma^2_{\boldsymbol{X}_m} \end{pmatrix}.$$

Now, we make an assumption.

An assumption

Larger values of $\sigma^2_{\boldsymbol{X}_i}$ are much more interesting than the smaller ones. (Namely, most of the system dynamic is happening in places, where the variance is relatively big.)

Even more intuition

$$C_X = \begin{pmatrix} \sigma^2_{\boldsymbol{X}_1} & 0 & \dots & 0 \\ 0 & \sigma^2_{\boldsymbol{X}_2} & \dots & 0 \\ \vdots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \sigma^2_{\boldsymbol{X}_m} \end{pmatrix}.$$

Suppose that I order the variances, such that

$$\sigma^2_{\boldsymbol{X}_1} \geq \sigma^2_{\boldsymbol{X}_2} \geq \cdots \geq \sigma^2_{\boldsymbol{X}_m}.$$

- In this case $\sigma^2_{\boldsymbol{X}_1}$ will tell me **the dynamics of the strongest!** — this is the **first principal component**.
- $\sigma^2_{\boldsymbol{X}_2}$ captures less system dynamics, and forms the **second principal component**.
- And so on...

The diagonalization (1)

1. Recall that X is our data matrix.
2. Compute a non-normalized covariance via XX^T.
3. We saw that XX^T is symmetric, that is, it has **real eigenvalues** and all eigenvectors are **orthogonal** to each other.
4. For such matrices, we can always performed the **eigenvalue decomposition**.

Eigenvalue Decomposition

$$XX^T = S\Lambda S^{-1},$$

where Λ is a diagonal matrix, and S is a matrix of eigenvectors of XX^T. (S's columns are normalized right eigenvectors of XX^T.)

5. Since the eigenvectors are orthonormal, $S^{-1} = S^T$!
6. Λ is a diagonal matrix with eigenvalues of XX^T!

The diagonalization (2)

- Recall that when we started to work with our data X, we actually worked in a sort of arbitrary coordinate system.
- We would like to figure out, what is the bases, that we should use, in order to have a diagonal covariance matrix instead of our original one (which has a bunch of correlated data measures).
- So, let us create a new set of measurements Y, that is related to the old set of measurements X as follows:

$$Y = S^T X.$$

(Note that this is just a linear transformation!) And, we would like to work in this new bases from now on.

The diagonalization (3)

Let us calculate the covariance of Y now.

$$\begin{aligned} C_Y &= \frac{1}{n-1} Y Y^T = \frac{1}{n-1} (S^T X)(S^T X)^T = \\ &= \frac{1}{n-1} S^T \underbrace{X X^T}_{S \Lambda S^T} S = \frac{1}{n-1} S^T S \Lambda S^T S = \frac{1}{n-1} \Lambda. \end{aligned}$$

What is Λ? — a diagonal matrix!
To conclude:

- If we work at the S^T bases — the covariance matrix C_Y of $Y = S^T X$ is diagonal ⇒ NO REDUNDANCY!
- Effectively, we figured out, what is the right way to look at our problem.

How many principal components should I use?

$$C_Y = \begin{pmatrix} \sigma^2_{\mathbf{Y}_1} & 0 & \dots & 0 \\ 0 & \sigma^2_{\mathbf{Y}_2} & \dots & 0 \\ \vdots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \sigma^2_{\mathbf{Y}_m} \end{pmatrix}.$$

Suppose that I order the variances, such that

$$\sigma^2_{\mathbf{Y}_1} \geq \sigma^2_{\mathbf{Y}_2} \geq \cdots \geq \sigma^2_{\mathbf{Y}_m}.$$

The **Percentage of Variance Explained** (PVE) by the principal component i, is defined by:

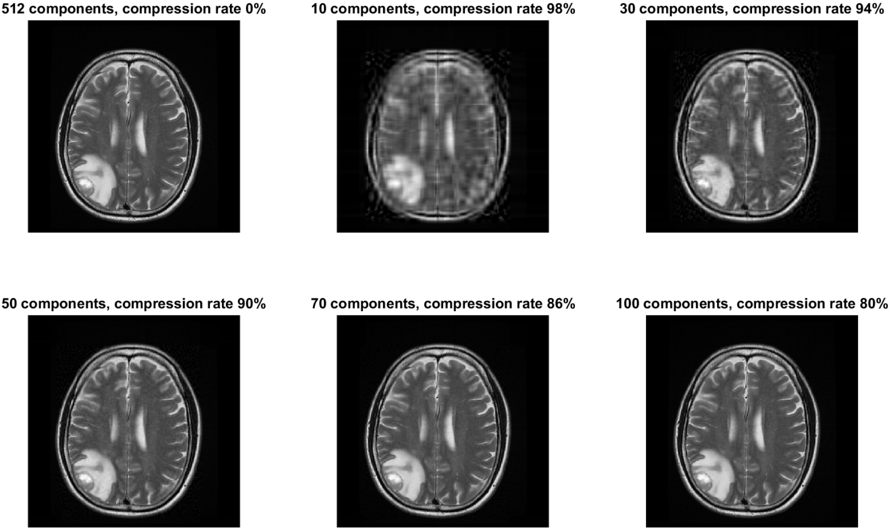
$$\frac{\sigma^2_{\mathbf{Y}_i}}{\sum_{j=1}^{m} \sigma^2_{\mathbf{Y}_j}}.$$

So, the first $k \leq m$ principal components explain $\frac{\sum_{i=1}^{k} \sigma^2_{\mathbf{Y}_i}}{\sum_{j=1}^{m} \sigma^2_{\mathbf{Y}_j}}$ of the system variance.

Image compression example — (pcaexample.py)

```
from PIL import Image
import numpy as np
from sklearn.decomposition import PCA
img = Image.open("mribrain.jpg").convert(’L’)
# load 2D array
im_arr = np.frombuffer(img.tobytes(), dtype=np.uint8)
im_arr = im_arr.reshape((img.size[1], img.size[0]))
# PCA
ncomp = 20 # change the number of components
pca = PCA(n_components=ncomp)
X_pca = pca.fit_transform(im_arr)

X_back = pca.inverse_transform(X_pca)
X_back[X_back<1] = 0
X_back = np.floor(X_back)
X_back = X_back.astype(np.uint8)

img_to_show = Image.fromarray(X_back,’L’)
print("# components = ",ncomp, " variance explained = ",
np.sum(pca.explained_variance_ratio_), " compression rate = ", 1 - (ncomp/512))
img_to_show
```

Image compression example — (pcaexample.py)



512 components, compression rate 0%

10 components, compression rate 98%

30 components, compression rate 94%

50 components, compression rate 90%

70 components, compression rate 86%

100 components, compression rate 80%

PCA Assumptions

1. A data **linearity** is assumed. (Kernel PCA.)

$$X_A = \begin{pmatrix} \boldsymbol{X}_1 \\ (3 \cdot \boldsymbol{X}_1 + 8) \end{pmatrix} \qquad X_B = \begin{pmatrix} \boldsymbol{X}_1 \\ (\boldsymbol{X}_1)^2 \end{pmatrix}$$

2. We assume that bigger variances have more important dynamics.
3. The principal components are assumed to be orthogonal.
4. We assume that the data-points come from the Gaussian distribution.

PCA — conclusions

- $+$
  1. A simple method — no parameters to tweak and no coefficients to adjust.
  2. A dramatic reduction in a data size.
  3. Easy to compute.
  4. Very powerful for many practical applications.
- $-$
  1. How do we incorporate a prior knowledge?.
  2. Too expensive for many applications — $O\left(n^3\right)$ complexity.
  3. Problems with outliers.
  4. Assumes linearity.

PCA regression example (stockpcr.py)

- The principal components regression (PCR) approach involves constructing principal components regression the first k principal components, $Z_1, \ldots, Z_k$ $k \ll p$, and then using these components as the predictors in a linear regression model that is fit using least squares.
- The key idea is that often a small number of principal components suffice to explain most of the variability in the data, as well as the relationship with the response

PCA regression example (stockpcr.py)

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn.metrics import zero_one_loss
from sklearn.metrics import confusion_matrix
df = pd.read_csv("stock.csv",index_col=0)
df.Direction = df.Direction.astype(’category’)
df["DirectionNum"] = df.Year
df.loc[df[’Direction’] =="Up", ’DirectionNum’] = 1
df.loc[df[’Direction’] =="Down", ’DirectionNum’] = 0
# construct X,Y
Y = df.DirectionNum
X = df.drop(["Direction","DirectionNum","Today"],axis=1)
# train test set
Xtr = X[X.Year<2005]
Ytr = Y[X.Year<2005]
Xtest = X[X.Year == 2005]
Ytest = Y[X.Year == 2005]
```

PCA regression example (stockpcr.py)

```
# PCA
ncomp = 4 # change the number of components
pca = PCA(n_components=ncomp)
X_pca = pca.fit_transform(Xtr)


# fit the model
model = LogisticRegression(solver='lbfgs')
model.fit(X_pca,Ytr)

# predict
x_test_pca = pca.transform(Xtest)
y_pred = model.predict(x_test_pca)

print("misclassification % = ",zero_one_loss(Ytest,y_pred))
print("Confusion matrix: \n", confusion_matrix(Ytest,y_pred))
```

Generalized linear models (1)

- The normal linear model that we discussed previously, deals with continuous response variables and continuous or discrete explanatory variables.
- Generalized linear models allow for arbitrary response distributions, including *discrete* ones.

Definition (Generalized Linear Model)

In a *generalized linear model* (GLM) the expected response for a given feature vector $\boldsymbol{x} = [x_1, \ldots, x_p]^\top$ is of the form

$$\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = h(\boldsymbol{x}^\top \boldsymbol{\beta}) \tag{2}$$

for some function h, which is called the *activation function*. The distribution of Y (for a given $\boldsymbol{x}$) may depend on additional *dispersion* parameters that model the randomness in the data that is not explained by $\boldsymbol{x}$.

Generalized linear models (2)

- The *inverse* of function h is called the *link function*. As for the linear model

  $$\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = h(\boldsymbol{x}^\top \boldsymbol{\beta}),$$

  is a model for a single pair $(\boldsymbol{x}, Y)$.
- A common assumption is that $Y_1, \ldots, Y_n$ come from the same family of distributions, e.g., normal, Bernoulli, or Poisson.

Logistic regression

- In a *logistic regression* or *logit model*, we assume that the response variables $Y_1, \ldots, Y_n$ are independent and distributed according to

$$Y_i \sim \mathrm{Ber}(h(\boldsymbol{x}_i^\top \boldsymbol{\beta})),$$

where h here is defined as the cdf of the *logistic distribution*

$$h(x) = \frac{1}{1 + \mathrm{e}^{-x}}.$$

- Large values of $\boldsymbol{x}_i^\top \boldsymbol{\beta}$ thus lead to a high probability that $Y_i = 1$, and small (negative) values of $\boldsymbol{x}_i^\top \boldsymbol{\beta}$ cause Y_i to be 0 with high probability.

Poisson regression

- Poisson regression assumes the response variable Y has a Poisson distribution.
- The Poisson regression is generally used for counting data.
- For example, suppose that you would like to count (predict) a future number of accidents in a mine.
- In *Poisson regression*, we assume that the response variables $Y_1, \ldots, Y_n$ are independent and distributed according to

$$Y_i \sim \text{Poiss}(\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}).$$

Logistic regression example LogitExample.py (1)

```
from sklearn import datasets
import numpy as np
data = datasets.load_wine ()
X = data.data[:, [9 ,10]]
y = np.array(data. target ==1 , dtype =np.uint)
X = np.append(np.ones(len(X)). reshape ( -1 ,1) ,X,axis =1)


# ######################################################

import statsmodels.api as sm
model = sm.Logit (y, X)
result = model.fit()
print(result.summary())
```

Logistic regression example LogitExample.py (2)

```
    Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  178
Model:                          Logit   Df Residuals:                      175
Method:                           MLE   Df Model:                            2
Date:                Wed, 27 Jan 2021   Pseudo R-squ.:                     inf
Time:                        21:51:59   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.8275      2.062      3.312      0.001       2.787      10.868
x1            -2.1460      0.337     -6.368      0.000      -2.807      -1.485
x2             1.9410      1.549      1.253      0.210      -1.096       4.978
==============================================================================
```

Empirical CDF (1)

Definition (Empirical CDF – the ECDF)

Let $X_1, \ldots, X_n$ be an iid sample from a CDF $F(x)$. Then, by assigning a mass of $1/n$ to each sample point, we define the Empirical CDF $\widehat{F}_n(x)$ to be

$$\widehat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \leq x\}}.$$

Using the ECDF, we can calculate the probability of say $\mathbb{P}(a \leq X \leq b)$ for $a < b$. Of course, this probability can be directly estimated from the data using the equation $\frac{1}{n} \sum_{i=1}^{n} 1_{\{a \leq X_i \leq b\}}$, however, it also holds that

$$\frac{1}{n} \sum_{i=1}^{n} 1_{\{a \leq X_i \leq b\}} = \frac{1}{n} \sum_{i=1}^{n} 1_{X_i \leq b} - \frac{1}{n} \sum_{i=1}^{n} 1_{X_i \leq a} = \widehat{F}_n(b) - \widehat{F}_n(a).$$

Empirical CDF (2)

Theorem

Let $X_1, \ldots, X_n$ be an iid sample from a CDF $F(x)$. Then, ECDF $\widehat{F}_n(x)$ is defined via

$$\widehat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \le x\}}.$$

Then, for any fixed x, it holds that

1. $\mathbb{E}[\widehat{F}_n(x)] = F(x)$
2. $\mathrm{Var}(\widehat{F}_n(x)) = \frac{F(x)(1-F(x))}{n}$, *and*
3. $\widehat{F}_n(x) \xrightarrow{P} F(x)$.

Statistical functionals (1)

Definition (Statistical functional and plugin estimator)

A statistical functional $T(F)$ is any function of a CDF F. The plugin estimator $\widehat{\theta}_n$ of $\theta = T(F)$ is defined via:

$$\widehat{\theta}_n = T(\widehat{F}_n).$$

Example: correlation

Let $\boldsymbol{W} = (X, Y)$ be a pair of random variables and let

$$T(F(x,y)) = \rho = \frac{\mathbb{E}\left[(X-\mu_x)(Y-\mu_y)\right]}{\sigma_x \sigma_y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2}\sqrt{\mathbb{E}[Y^2] - \mathbb{E}[Y]^2}}$$

be the correlation between X and Y. In order to obtain $\widehat{\rho} = T(\widehat{F}_n)$, we will need the following:

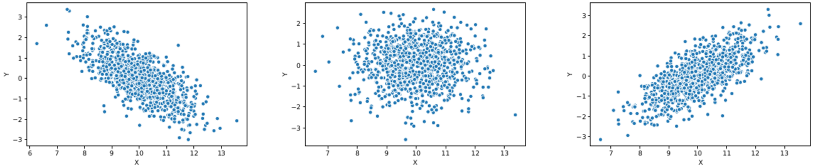
Statistical functionals (2)

1. $\mathbb{E}[X] = \theta^{(1_x)} = \int x dF(x,y) \Rightarrow \theta_n^{(1_x)} = \frac{1}{n}\sum_{i=1}^{n} x_i;$
2. $\mathbb{E}[Y] = \theta^{(1_y)} = \int x dF(x,y) \Rightarrow \theta_n^{(1_y)} = \frac{1}{n}\sum_{i=1}^{n} y_i;$
3. $\mathbb{E}[X^2] = \theta^{(2_x)} = \int x^2 dF(x,y) \Rightarrow \theta_n^{(1_x)} = \frac{1}{n}\sum_{i=1}^{n} x_i^2;$
4. $\mathbb{E}[Y^2] = \theta^{(2_y)} = \int y dF(x,y) \Rightarrow \theta_n^{(2_y)} = \frac{1}{n}\sum_{i=1}^{n} y_i^2;$
5. $\mathbb{E}[XY] = \theta^{(xy)} = \int xy dF(x,y) \Rightarrow \theta_n^{(xy)} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i;$
   and thus,

$$\widehat{\rho} = \frac{\theta_n^{(xy)} - \theta_n^{(1_x)}\theta_n^{(1_y)}}{\sqrt{\theta_n^{(2_x)} - \left(\theta_n^{(1_x)}\right)^2}\sqrt{\theta_n^{(2_y)} - \left(\theta_n^{(1_y)}\right)^2}}$$

Statistical functionals (3)

3 datasets generated from bivariate normal distribution with mean $\mu = (10, 0)^\top$ and correlations of -0.7, 0, and 0.7. The sample correlation based on 1000 points is -0.712, -0.004, and 0.709.



(a) $\rho = -0.7$ (b) $\rho = 0$ (c) $\rho = 0.7$

Figure 4: 3 datasets generated from bivariate normal distribution with mean $\mu = (10, 0)^\top$ and correlations of -0.7, 0, and 0.7. The sample correlation based on 1000 points is -0.712, -0.004, and 0.709.

The bootstrap method

Naturally, obtaining a plugin estimator $\widehat{\theta}_n$ is not sufficient; namely, we need also to supply the corresponding confidence interval. For some estimators, we can do this analytically.
In general, let $T_n = s(X_1, \ldots, X_n)$ be a statistic and suppose that

$$\widehat{\theta}_n = T(\widehat{F}_n) = T_n = \frac{1}{n}\sum_{i=1}^{n} X_i = \overline{X}_n.$$

Then, we can obtain the variance of *this specific* T_n with respect to F, is

$$\mathrm{Var}_F\left(T_n\right) = \mathrm{Var}_F\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2} n \mathrm{Var}_F(X) = \frac{\sigma^2}{n},$$

where σ^2 can be estimated (replaced) by $\mathrm{Var}_{\widehat{F}}(T_n) = \widehat{\sigma}^2$, which is given by

$$\mathrm{Var}_{\widehat{F}}(T_n) = \widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2, \text{ where } \overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{3}$$

From CLT, we know that T_n is approximately distributed $\mathsf{N}\left(\mathbb{E}[X], \sigma^2/n\right)$, and therefore, the $1-\alpha$ CI for T_n is $\overline{X}_n \pm z_{1-\alpha/2}\widehat{\sigma}/\sqrt{n}$.

The bootstrap method

The issue is that $\mathrm{Var}_{\widehat{F}}(T_n)$ can be unavailable for different statistic function T_n (think about a median or say $T_n = \max\{X_1, \ldots X_n\} \times \min\{X_1, \ldots X_n\}$). The bootstrap method aims to find confidence intervals for any statistic T_n using a computational simulation method. We next discuss the bootstrap idea in more details. The bootstrap method consists of two steps.

1. The estimation of $\mathrm{Var}_F(T_n)$ with $\mathrm{Var}_{\widehat{F}}(T_n)$.
2. The approximation of $\mathrm{Var}_{\widehat{F}}(T_n)$ using Monte Carlo simulation.
3. Develop confidence interval for T_n.

The $\overline{X}_n$ estimator: For the estimator $T_n = \frac{1}{n}\sum_{i=1}^{n} X_i = \overline{X}_n$ in (3), step 1 was sufficient (namely, we do not need Step 2, since in this case, $\mathrm{Var}_{\widehat{F}}(T_n)$ can be computed analytically via (3), and the CI in Step 3 is readily available via the central limit theorem).

The Nonparametric Bootstrap

1. Given a dataset $X_1, \ldots, X_n$, suppose that this sample is an iid sample from the CDF $F(x)$.
2. Then, a bootstrapped sample $X_1^*, \ldots, X_n^*$ is a sample from the ECDF $\widehat{F}_n(x)$. In order to obtain a bootstrapped sample, recall that the ECDF $\widehat{F}_n(x)$ assigns a mass of $1/n$ to each X_i for $1 \leq i \leq n$, so, the bootstrapped sample $X_1^*, \ldots, X_n^*$ is just a sample with replacement from $X_1, \ldots, X_n$.
3. Please note that $X_1^*, \ldots, X_n^*$ can contain several X_i values for some $1 \leq i \leq n$.

Example: Suppose that the original data $\boldsymbol{X}$ is $\{1, 2, 5\}$. Then, some possible bootstrapped samples are

$$\boldsymbol{X}_1^* = (1, 2, 5),\ \boldsymbol{X}_2^* = (5, 5, 1),\ \boldsymbol{X}_3^* = (1, 1, 1),$$

etc..

The Nonparametric Bootstrap (2)

Note that each bootstrapped sample $X_1^{*(i)}, \ldots, X_n^{*(i)}$ is generated from $\widehat{F}_n$. Moreover, we can now calculate

$$T_n^{*(i)} = s\left(X_1^{*(i)}, \ldots, X_n^{*(i)}\right).$$

Now, as soon as we obtained the bootstrapped statistic values for $T_n^{*(1)}, \ldots, T_n^{*(B)}$, we can actually solve the second problem, namely, to obtain an estimator for $\mathrm{Var}_{\widehat{F}}(T_n)$. The latter is achieved via

$$\mathrm{Var}_{\widehat{F}}^{(boot)}(T_n) = \frac{1}{B}\sum_{i=1}^{B}\left(T_n^{*(i)} - \overline{T}_n^*\right)^2, \quad \text{where } \overline{T}_n^* = \frac{1}{B}\sum_{i=1}^{B} T_n^{*(i)}.$$

Why the bootstrap is so important?

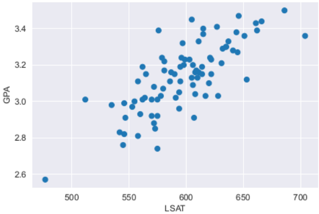
The bootstrap method can give us CIs!!!

$$\mathrm{CI}_{\mathrm{Normal}} = T_n \pm z_{1-\alpha/2} \times \mathrm{se}^{(boot)}, \text{ where } \mathrm{se}^{(boot)}$$
$$= \sqrt{\mathrm{Var}_{\widehat{F}}^{(boot)}\left(\widehat{T_n}\right)} = \sqrt{\frac{\sum_{i=1}^{B}\left(T_n^{*(i)} - \overline{T}_n^{*}\right)^2}{B}}.$$

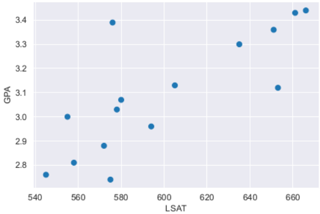
1. Suppose that we have a regression problem with coeficient β.
2. $\hat{\beta}$ can be obtained via the model training (but please note that $\hat{\beta}$ is a statistical functional) why?
3. This means that the corresponding CI for $\hat{\beta}$ can be obtained via bootstrap.

Example (NonParametricLSAT.py)

1. The data contains the score for the class on a national law test, and GPA, the average undergraduate grade-point average for the class.
2. A sample of 15 observations were taken from 82 law schools participating in the study.
3. We are interested to estimate the correlation between LSAT and GPA. Since we have an access to the full data (82 pairs), we can find the true correlation, which is equal to $\theta = 0.75999786$.
4. A typical normal CI is (0.553, 0.9995).



(a) Full 82 points data



(b) A sample of 15 points data