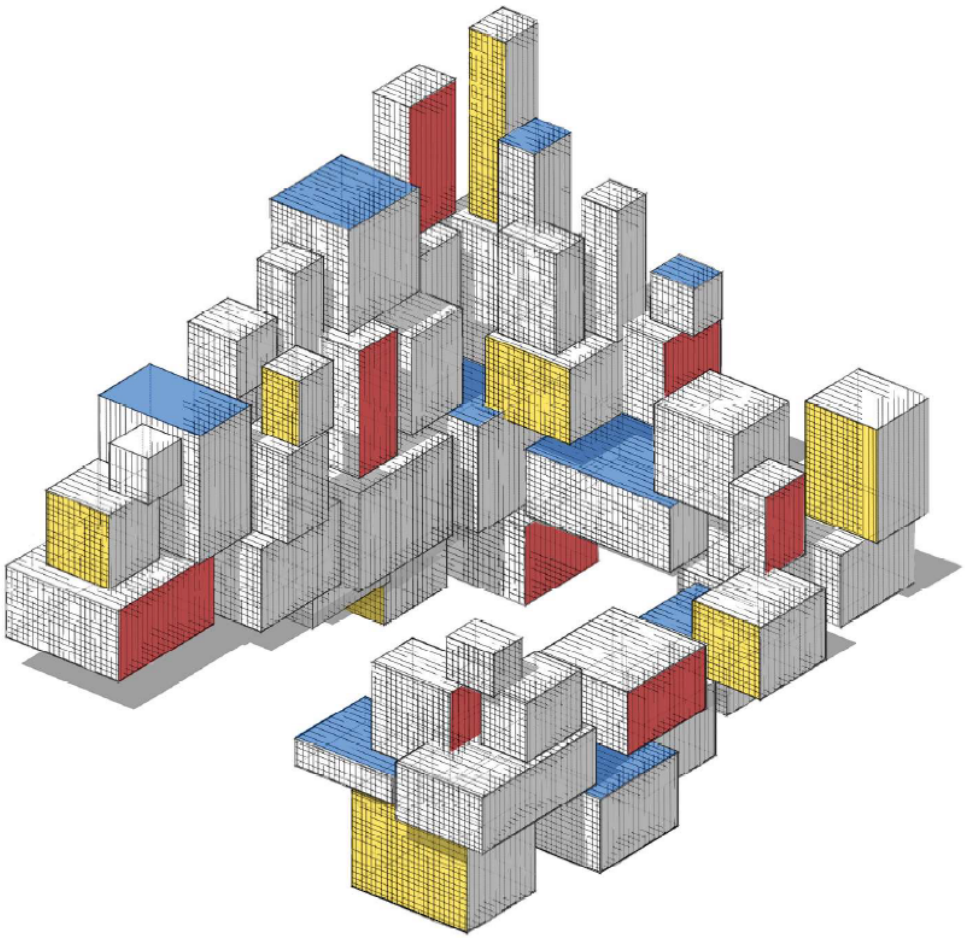
Generalized Linear Models



Purpose

In this lecture we discuss:

- Variable selection for linear models
- Generalized linear models

Variable Selection

Among a large number of possible explanatory variables in a linear model, we wish to select those which best explain the observed responses.

By eliminating redundant explanatory variables, we reduce the statistical error without increasing the approximation error, and thus reduce the (expected) generalization risk of the learner.

We present two methods for variable selection. They are illustrated on a few variables from the birth weight data set **`birthwt`**.

The data set contains information on the birth weights (masses) of babies, as well as various characteristics of the mother.

Birthweight Data

The data can be obtained from:

```
bwt = sm.datasets.get_rdataset("birthwt","MASS").data
```

```
age:   mother's age in years
lwt:   mother's weight in lbs
race:  mother's race (1 = white, 2 = black, 3 = other)
smoke: smoking status during pregnancy (0 = no, 1 = yes)
ptl:   no. of previous premature labors
ht:    history of hypertension (0 = no, 1 = yes)
ui:    presence of uterine irritability (0 = no, 1 = yes)
ftv:   no. of physician visits during first trimester
bwt:   birth weight in grams
```

Let `ftv1` and `ptl1` be the indicators of `ftv` $\geqslant 1$ and `ptl` $\geqslant 1$.

Forward Selection

The forward selection method is an iterative method for variable selection.

In the first iteration we consider which feature `f1` is the most significant in terms of its P-value in the models `bwt~f1`, with `f1` $\in$ {`lwt`, `age`, ...}.

This feature is then selected into the model. In the second iteration, the feature `f2` that has the smallest P-value in the models `bwt~f1 + f2` is selected, where `f2` $\neq$ `f1`, and so on.

Usually only features are selected that have a P-value of at most 0.05.

The following Python implentation yields the following selection:
`ui, ht, lwt, smoke`

forwardselection.py

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

bwt = sm.datasets.get_rdataset("birthwt","MASS").data
ftv1 = (bwt['ftv']>=1).astype(int)
ptl1 = (bwt['ptl']>=1).astype(int)

remaining_features = {'lwt', 'age', 'C(ui)', 'smoke','C(ht)', 'ftv1', 'ptl1'}
selected_features = []
while remaining_features:
  PF = []  #list of (P value, feature)
  for f in remaining_features:
    temp = selected_features + [f]  #temporary list of features
    formula = 'bwt~' + '+'.join(temp)
    fit = ols(formula,data=bwt).fit()
    pval= fit.pvalues[-1]
    if pval < 0.05:
       PF.append((pval,f))
  if PF:  #if not empty
     PF.sort(reverse=True)
     (best_pval, best_f) = PF.pop()
     remaining_features.remove(best_f)
     print('feature {} with P-value = {:.2E}'.
            format(best_f, best_pval))
     selected_features.append(best_f)
  else:
     break
```

```
feature C(ui) with P-value = 7.52E-05
feature C(ht) with P-value = 1.08E-02
feature lwt with P-value = 6.01E-03
feature smoke with P-value = 7.27E-03
```

Do the mother's weight, her age, her race, and whether she smokes explain the baby's birthweight?

```
formula = 'bwt~lwt+age+C(race)+ smoke'
bwt_model = ols(formula, data=bwt).fit()
print(bwt_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable: bwt                R-squared: 0.148
Model: OLS                        Adj. R-squared: 0.125
Method: Least Squares             F-statistic: 6.373
No. Observations: 189             Prob (F-statistic):  1.76e-05
Df Residuals: 183                 Log-Likelihood: -1498.4
Df Model: 5                       AIC: 3009.
                                  BIC: 3028.
==============================================================================
                 coef    std err      t      P>|t|  [0.025     0.975]
------------------------------------------------------------------------------
Intercept     2839.4334   321.435   8.834   0.000  2205.239   3473.628
C(race)[T.2]  -510.5015   157.077   -3.250  0.001  -820.416   -200.587
C(race)[T.3]  -398.6439   119.579   -3.334  0.001  -634.575   -162.713
smoke         -401.7205   109.241   -3.677  0.000  -617.254   -186.187
lwt              3.9999     1.738    2.301  0.022     0.571      7.429
age             -1.9478     9.820   -0.198  0.843   -21.323     17.427
==============================================================================
Omnibus: 3.916                Durbin-Watson: 0.458
Prob(Omnibus): 0.141          Jarque-Bera (JB): 3.718
Skew: -0.343                  Prob(JB): 0.156
Kurtosis: 3.038               Cond. No. 899.
```

The individual Student tests indicate that:

- the mother's weight is linearly associated with child weight, after adjusting for age, race, and smoking status (P-value = 0.022).
- the age of the mother is not significantly linearly associated with child weight at birth, when mother weight, race, and smoking status are already taken into account (P-value = 0.843);
- weight at birth is significantly lower for a child born to a mother who smokes, compared to children born to non-smoking mothers of the same age, race, and weight, with a P-value of 0.00031 (to see this, inspect `bwt_model.pvalues`).
- regarding the interpretation of the variable `race`, we note that the first level corresponds to white mothers. The estimate of −510.501 g for `C(race)[T.2]` represents the difference in the child's birth weight between black mothers and white mothers (reference group), and this result is significantly different from zero (P-value = 0.001) in a model adjusted for the mother's weight, age, and smoking status.

Interaction

We can also include interaction terms in the model. Let us see whether there is any interaction effect between smoke and age via the model

$$\texttt{Bwt} = \beta_0 + \beta_1 \texttt{age} + \beta_2 \texttt{smoke} + \beta_3 \texttt{age} \times \texttt{smoke} + \varepsilon.$$

In Python this can be done as follows:

```
formula = 'bwt~age*smoke'
bwt_model = ols(formula, data=bwt).fit()
print(bwt_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable: bwt         R-squared: 0.069
Model: OLS                 Adj. R-squared: 0.054
Method: Least Squares      F-statistic: 4.577
No. Observations: 189      Prob (F-statistic):  0.00407
Df Residuals: 183          Log-Likelihood: -1506.8
Df Model: 5                AIC: 3009.
                           BIC: 3028.
==============================================================================
                 coef    std err     t      P>|t|    [0.025   0.975]
------------------------------------------------------------------------------
Intercept      2406.1  292.190    8.235   0.000  1829.6   2982.5
smoke           798.2  484.342    1.648   0.101  -157.4   1753.7
age              27.7   12.149    2.283   0.024     3.8     51.7
age:smoke       -46.6   20.447   -2.278   0.024   -86.9     -6.2
```

Interaction

We conclude that the effect of the mother's age on the child's weight depends on the smoking status of the mother (P-value = 0.024).

For non-smoking mothers (`smoke = 0`), the mean child weight at birth increases on average by 27.7 grams for each year of the mother's age. This is statistically significant, as can be seen from the 95% confidence intervals for the parameters (which does not contain zero):

```
bwt_model.conf_int()
```

	0	1
Intercept	1829.605754	2982.510194
age	3.762780	51.699977
smoke	-157.368023	1753.717779
age:smoke	-86.911405	-6.232425

For smoking mothers, there seems to be a decrease in birthweight, $\widehat{\beta}_1 + \widehat{\beta}_3 = 27.7 - 46.6 = -18.9$, but (exercise!) this is not statistically significant.

Generalized Linear Model

The normal linear model deals with continuous response variables — such as height and crop yield — 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

$$\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = h(\boldsymbol{x}^\top \boldsymbol{\beta}) \tag{1}$$

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 Model

As for the linear model, (1) is a model for a single pair $(\boldsymbol{x}, Y)$.

Using the model simplification where the response variables are assumed to be constant, the corresponding model for a whole training set $\mathcal{T} = \{(\boldsymbol{x}_i, Y_i)\}$ is that the $\{\boldsymbol{x}_i\}$ are fixed and that the $\{Y_i\}$ are independent; each Y_i satisfying (1) with $\boldsymbol{x} = \boldsymbol{x}_i$.

Writing $\boldsymbol{Y} = [Y_1, \ldots, Y_n]^\top$ and defining $\boldsymbol{h}$ as the multivalued function with components h, we have

$$\mathbb{E}_{\mathbf{X}} \boldsymbol{Y} = \boldsymbol{h}(\mathbf{X}\boldsymbol{\beta}),$$

where $\mathbf{X}$ is the (model) matrix with rows $\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top$.

Generalized Linear Model

A common assumption is that $Y_1, \ldots, Y_n$ come from the same family of distributions, e.g., normal, Bernoulli, or Poisson.

The central focus is the parameter vector $\boldsymbol{\beta}$, which summarizes how the matrix of explanatory variables $\mathbf{X}$ affects the response vector $\boldsymbol{Y}$.

The class of generalized linear models can encompass a wide variety of models.

Obviously the normal linear model is a generalized linear model, with $\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = \boldsymbol{x}^\top \boldsymbol{\beta}$, so that h is the identity function.

In this case, $Y \sim \mathcal{N}(\boldsymbol{x}^\top \boldsymbol{\beta}, \sigma^2)$, $i = 1, \ldots, n$, where σ^2 is a dispersion parameter.

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

Estimation of the parameter vector $\boldsymbol{\beta}$ from the observed data is not as straightforward as for the ordinary linear model, but can be accomplished via the minimization of a suitable training loss, as explained below.

Logistic Regression

As the $\{Y_i\}$ are independent, the pdf of $\boldsymbol{Y} = [Y_1, \ldots, Y_n]^\top$ is

$$g(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X}) = \prod_{i=1}^{n} [h(\boldsymbol{x}_i^\top \boldsymbol{\beta})]^{y_i} [1 - h(\boldsymbol{x}_i^\top \boldsymbol{\beta})]^{1-y_i}.$$

Maximizing the log-likelihood $\ln g(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X})$ with respect to $\boldsymbol{\beta}$ gives the maximum likelihood estimator of $\boldsymbol{\beta}$. In a supervised learning framework, this is equivalent to minimizing:

$$\begin{aligned} -\frac{1}{n} \ln g(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X}) &= -\frac{1}{n} \sum_{i=1}^{n} \ln g(y_i \mid \boldsymbol{\beta}, \boldsymbol{x}_i) \\ &= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \ln h(\boldsymbol{x}_i^\top \boldsymbol{\beta}) + (1 - y_i) \ln(1 - h(\boldsymbol{x}_i^\top \boldsymbol{\beta})) \right]. \end{aligned} \tag{2}$$

Cross-Entropy Training Loss

We can interpret (2) as the cross-entropy training loss associated with comparing a true conditional pdf $f(y \mid \boldsymbol{x})$ with an approximation pdf $g(y \mid \boldsymbol{\beta}, \boldsymbol{x})$ via the loss function

$$\text{Loss}(f(y \mid \boldsymbol{x}), g(y \mid \boldsymbol{\beta}, \boldsymbol{x})) := -\ln g(y \mid \boldsymbol{\beta}, \boldsymbol{x}) = -y \ln h(\boldsymbol{x}^\top \boldsymbol{\beta}) - (1-y) \ln(1 - h(\boldsymbol{x}^\top \boldsymbol{\beta})).$$

The cross-entropy training loss (2) can be rewritten as

$$r_\tau(\boldsymbol{\beta}) := \frac{1}{n} \sum_{i=1}^{n} \left[(1 - y_i) \boldsymbol{x}_i^\top \boldsymbol{\beta} + \ln\left(1 + \mathrm{e}^{-\boldsymbol{x}_i^\top \boldsymbol{\beta}}\right) \right].$$

Minimizing $r_\tau(\boldsymbol{\beta})$ constitutes a convex optimization problem with a positive semidefinite Hessian matrix $\mathbf{H}(\boldsymbol{\beta})$. Consequently, we can find an optimal $\boldsymbol{\beta}$ efficiently; e.g., via Newton's method: iteratively compute

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \mathbf{H}^{-1}(\boldsymbol{\beta}_{t-1}) \nabla r_\tau(\boldsymbol{\beta}_{t-1}), \tag{3}$$

until the sequence $\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots$ has converged.

Gradient and Hessian

We leave it as exercise to show that the gradient $\nabla r_\tau(\boldsymbol{\beta})$ and Hessian $\mathbf{H}(\boldsymbol{\beta})$ of $r_\tau(\boldsymbol{\beta})$ are given by

$$\nabla r_\tau(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} (\mu_i - y_i)\, \boldsymbol{x}_i \tag{4}$$

and

$$\mathbf{H}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \mu_i (1 - \mu_i)\, \boldsymbol{x}_i\, \boldsymbol{x}_i^\top, \tag{5}$$

respectively, where $\mu_i := h(\boldsymbol{x}_i^\top \boldsymbol{\beta})$.

Example

The figure shows the outcomes of 100 independent Bernoulli random variables, where each success probability, $(1+\exp(-(\beta_0+\beta_1 x)))^{-1}$, depends on x and $\beta_0 = -3,\ \beta_1 = 10$. The minimum training loss curve (red line) is obtained via the Newton scheme (3), giving estimates $\widehat{\beta_0} = -2.66$ and $\widehat{\beta_1} = 10.08$.

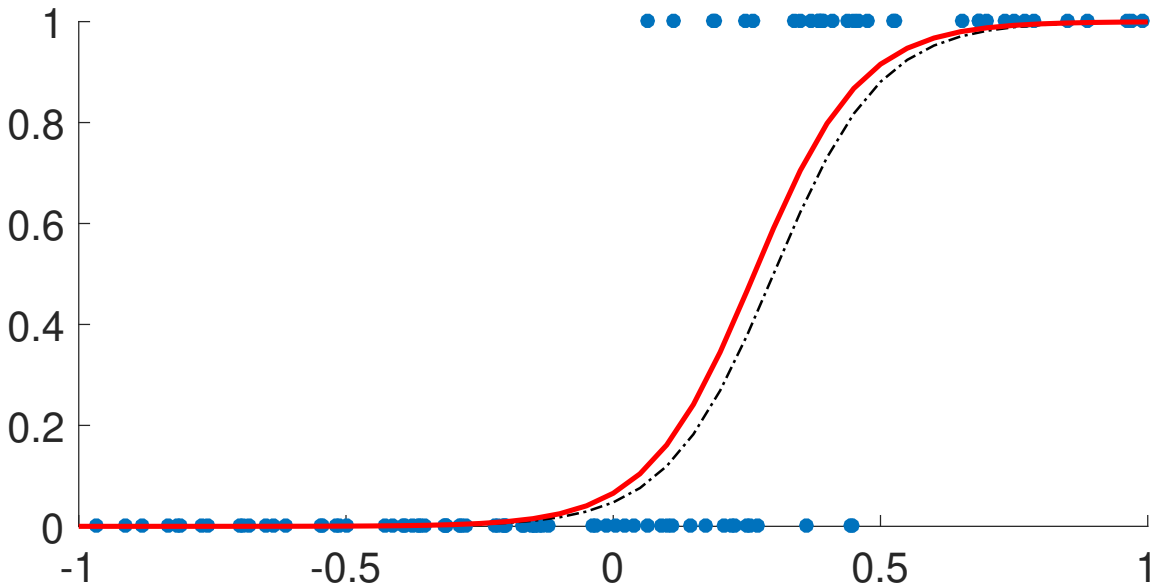


Figure: Logistic regression data (blue dots), fitted curve (red), and true curve (black dashed).

logreg1d.py

```
import numpy as np
import matplotlib.pyplot as plt
from  numpy.linalg import lstsq

n = 100                                          # sample size
x = (2*np.random.rand(n)-1).reshape(n,1)  # explanatory variables
beta = np.array([-3, 10])
Xmat = np.hstack((np.ones((n,1)), x))
p = 1/(1 + np.exp(-Xmat @ beta))
y = np.random.binomial(1,p,n)                    # response variables

# initial guess
betat = lstsq((Xmat.T @ Xmat),Xmat.T @ y, rcond=None)[0]

grad = np.array([2,1])                                  # gradient
```

```
while (np.sum(np.abs(grad)) > 1e-5) :      # stopping criteria
    mu = 1/(1+np.exp(-Xmat @ betat))
    # gradient
    delta = (mu - y).reshape(n,1)
    grad = np.sum(np.multiply( np.hstack((delta,delta)),Xmat), axis=0).
        T
    # Hessian
    H = Xmat.T @ np.diag(np.multiply(mu,(1-mu))) @ Xmat
    betat = betat - lstsq(H,grad,rcond=None)[0]
    print(betat)

plt.plot(x,y, '.') # plot data

xx = np.linspace(-1,1,40).reshape(40,1)
XXmat = np.hstack( (np.ones((len(xx),1)), xx))
yy = 1/(1 + np.exp(-XXmat @ beta))
plt.plot(xx,yy,'r-')                       #true logistic curve
yy = 1/(1 + np.exp(-XXmat @ betat));
plt.plot(xx,yy,'k--')
```