Topic 3: Logistic Regression

The *Logistic Regression (LR)* is a parametric method for **binary** classification [Coelho and Richert, 2015, Chapter 5] of the data which are linearly separable. It is used in credit scoring and behavioural scoring for financial institutes to decide on the granting of credit to applicants [Thomas, 2000].

Logistic regression is a huge topic which a lot of statistical inference theory which can be found in Hosmer et al. [2013] and Agresti [2002]. Its generalisation to multinomial LR and ElasticNet are still founded in mathematical statistics but its generalisation to artificial neural networks is inspired by biophysics rather than mathematical theory.

3.1 Logistic Regression

The logistic regression [Cox, 1958] avoids the out-of-range problem in linear regression (1.3) by modelling $\mathbb{P}(Y = 1|X)$ using the `https://en.wikipedia.org/wiki/Logistic_function` [Berkson, 1944]:

$$S : (-\infty, \infty) \to (0, 1), \quad S(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}} \tag{3.1}$$

leading to a "multiple" or "multivariate" logistic regression:

$$\begin{aligned} \mathbb{P}(Y = 1|X_1 = x_1, \cdots, X_p = x_p) &= \frac{1}{1 + \exp(-(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p))} \\ &= S(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p). \end{aligned} \tag{3.2}$$

It can be written in vector form:

$$\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = S(\beta^T \tilde{\boldsymbol{x}})$$

where $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_p)$ and $\widetilde{\boldsymbol{x}}_j = (1, \boldsymbol{x}_j)$.

Given an input $\boldsymbol{x}$, the LR model provides a prediction as follows based on the conditional probability (assuming the cut-off is 0.5):

$$h(\boldsymbol{x}) = \begin{cases} 0, & \mathbb{P}(Y=1|X=\boldsymbol{x}) < 0.5 \\ 1, & \mathbb{P}(Y=1|X=\boldsymbol{x}) \geq 0.5 \end{cases}$$

or based the log-odds (or logit or 'link'?):

$$h(\boldsymbol{x}) = \begin{cases} 0, & \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p < 0 \\ 1, & \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \geq 0 \end{cases}$$

3.1.1 Estimating the Regression Coefficients

The parameters/coefficients β_j in the logistic model (3.2) are **unknown**, and have to be estimated from the training data $(\boldsymbol{x}_i, y_i)$, $i = 1, \cdots, n$ using **maximum likelihood estimation** (MLE), i.e. we estimate the parameters β_j so that the **likelihood function** of $\beta_0, \cdots, \beta_p$ is maximised:

$$L(\beta_0, \cdots, \beta_p;\ y_1, \cdots, y_n | \boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) = \prod_{i=1}^{n} \mathbb{P}(Y = y_i | \boldsymbol{X} = \boldsymbol{x}_i) \tag{3.3}$$

where Y is binary and follows a **Bernoulli distribution**, the feature data is p-dimensional $\boldsymbol{x}_i = (x_1^{(i)}, \cdots, x_p^{(i)})$.

According to `https://en.wikipedia.org/wiki/Bernoulli_distribution`, if $X \sim Bernoulli(p)$, then the probability mass function of observing $x \in \{0, 1\}$ is given by $\mathbb{P}(x) = p^x (1-p)^{1-x}$. For $Y \sim Bernoulli(\mathbb{P}(Y = 1|\boldsymbol{X}))$, we have

$$\mathbb{P}(Y = y_i | \boldsymbol{X} = \boldsymbol{x}_i) = \left(\frac{e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}}{1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}} \right)^{y_i} \left(1 - \frac{e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}}{1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}} \right)^{1-y_i} = e^{y_i \widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}} \cdot (1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}})^{-y_i} \cdot (1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}})^{-(1-y_i)}$$

where $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_p)$ and $\widetilde{\boldsymbol{x}}_i = (1, \boldsymbol{x}_i)$.

Substituting it into (3.3), we have

$$L := L(\beta_0, \cdots, \beta_p;\ y_1, \cdots, y_n | \boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) = \prod_{i=1}^{n} (e^{y_i \widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}})(1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}})^{-1}.$$

Taking natural log leads to

$$\ln L = \sum_{i=1}^{n} y_i \widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}). \tag{3.4}$$

If there is a maximum $\hat{\boldsymbol{\beta}}$ such that $\ln L$ is maximum, then Calculus theory tells us that:

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, L = \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \ln L \\ \Rightarrow \frac{\partial}{\partial \boldsymbol{\beta}} (\ln L)\Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} &= \frac{\partial}{\partial \boldsymbol{\beta}} \left(\sum_{i=1}^{n} y_i \widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}) \right) \Bigg|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} = \mathbf{0}. \end{aligned}$$

leading to

$$\sum_{i=1}^{n} y_i x_k^{(i)} - \sum_{i=1}^{n} \frac{x_k^{(i)} e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}}{1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}} = 0, \quad k = 0, 1, \cdots, p$$

where $x_0^{(i)}$ is defined to be 1.

This is a system of nonlinear equations which can be solved using numerically such as applying `https://en.wikipedia.org/wiki/Newton's_method` to estimate $\hat{\boldsymbol{\beta}}$.

A special case ($p = 1$) of the above mathematical derivation was asked in the past final exam.

Example 3.1.1 (Final Exam May 2019, Q5(a))**.** The parameters β in logistic regression can be estimated through maximum likelihood estimation. The estimation can be done by solving the equations from maximized log-likelihood functions, i.e. $\frac{\partial l(\beta)}{\partial \beta_j} = 0$, $j = 0, 1$ through numerical methods. For logistic regression with one predictor, show that the differentiated log-likelihood functions for β_0 and β_1 respectively are

$$\frac{\partial l(\beta)}{\partial \beta_0} = \sum_{i=1;y_i=1}^{n} 1 - \sum_{i=1}^{n} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$$
$$\frac{\partial l(\beta)}{\partial \beta_1} = \sum_{i=1;y_i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}.$$

Your proof shall start from stating probability of Class 1, $\mathbb{P}(y_i = 1)$ and probability of Class 0, $\mathbb{P}(y_i = 0)$. (10 marks)

Solution: The probabilities for $\mathbb{P}(Y = 1)$ and $\mathbb{P}(Y = 0)$ are modelled by

$$\mathbb{P}(Y = 1) = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$$
$$\mathbb{P}(Y = 0) = 1 - \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} = \frac{1}{1+e^{\beta_0+\beta_1 x_i}}$$

Likelihood function, $\mathbb{L}(\beta|\boldsymbol{Y})$

$$\mathbb{L}(\beta|\boldsymbol{Y}) = \prod_{i=1;y_i=1}^{n} \mathbb{P}(y_i = 1) \prod_{i=1;y_i=0}^{n} \mathbb{P}(y_i = 0) = \prod_{i=1;y_i=1}^{n} e^{\beta_0+\beta_1 x_i} \prod_{i=1;y_i=0}^{n} \frac{1}{1+e^{\beta_0+\beta_1 x_i}}$$

Log-likelihood function, $l(\beta)$,

$$l\beta) = \ln \mathbb{L}(\beta|\boldsymbol{Y}) = \sum_{i=1;y_i=1}^{n} (\beta_0 + \beta_1 x_i) - \sum_{i}^{n} \ln(1+e^{\beta_0+\beta_1 x_i})$$

Differentiate $l(\beta)$ with respect to β_0 and β_1 leads to

$$\frac{\partial l(\beta)}{\partial \beta_0} = \sum_{i=1;y_i=1}^{n} 1 - \sum_{i=1}^{n} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$$
$$\frac{\partial l(\beta)}{\partial \beta_1} = \sum_{i=1;y_i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}.$$

Remark: The above calculations will only work if the inputs are not perfectly correlated (linearly dependent). If there is any perfect correlation due to 'duplicate entries', the duplicate entries can be dropped (until no more duplicate entries).

3.1.2 Hypothesis Testing and Inference

For most predictive models, it is impossible to develop the hypothesis testing theory for them. However, statisticians have developed the hypothesis testing theory for LR to regarding the confidence of the model and the parameters β_j.

To test if a particular coefficient $\beta_j = 0$, we set up the test

$$H_0 : \beta_j = 0 \quad \text{vs} \quad H_1 : \beta_j \neq 0.$$

and form the *Z-statistic* which is closed to normal when "n" is large [Hastie et al., 2008]:

$$\frac{\hat{\beta}_j - 0}{[SE(\hat{\beta})]_{jj}} \sim Normal(0, 1).$$

The *standard error* $SE(\hat{\beta})$ is the inverse of the estimated $(p+1)\times(p+1)$ *Fisher information matrix* [Ohlsson and Johansson, 2010]:

$$SE(\hat{\beta}) = \left[\frac{\partial^2}{\partial \boldsymbol{\beta}^2} \left(\sum_{i=1}^{n} y_i \widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\widetilde{\boldsymbol{x}}_i^T \boldsymbol{\beta}}) \right) \right]^{-1}$$

By considering a significance level of $\alpha = 5\%$.

Z-**statistic**	large	small
p-**value**	**small**	**large**
null hypothesis H_0	should be rejected	should not be rejected
interpretation	X is a factor influencing Y	X and Y is most likely not related.

A $(1 - \frac{\alpha}{2}) \times 100\%$ confidence interval for β_j, $j = 1, \cdots, p$, can be calculated using the Z-statistic:

$$\hat{\beta}_j \pm Z_{1-\alpha/2} SE(\hat{\beta}_j).$$

A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. In this case, $\alpha = 0.05$ so $Z_{1-\alpha/2} =$ `qnorm(1-0.05/2)` $= 1.959964 \approx 1.96$, therefore, the 95% confidence interval for β_j takes the form

$$[\hat{\beta}_j - 1.96 \cdot SE(\hat{\beta}_j),\ \hat{\beta}_j + 1.96 \cdot SE(\hat{\beta}_j)]. \tag{3.5}$$

The implementation of the above statistical estimates are encoded in R as the algorithm for generalised linear model (GLM).

Logistic Regression in R's GLM

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

# iteratively reweighted least squares
glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)
```

To set GLM to LR mode, we need to set `family=binomial`. The `start` can be used if we know the range of the parameters β_j.

Deviance is a measure of the badness of fit for GLM — higher numbers indicate worse fit. R reports two forms of deviance — the *null deviance*

$$= 2(LL(Saturated\_Model) - LL(Null\_Model)) \text{ on df} = df\_Sat - df\_Null$$

and the *residual deviance*

$$= 2(LL(Saturated\_Model) - LL(Proposed\_Model)) \text{ on df} = df\_Sat - df\_Proposed$$

where LL is the log-likelihood in (3.4).

- The Saturated Model is a model that assumes each data point has its own parameters (which means we have n parameters to estimate.)
- The Null Model assumes one parameter for all of the data points, which means we only estimate 1 parameter.
- The Proposed Model assumes we can explain our data points with p parameters + an intercept term, so we have $p + 1$ parameters.

The null deviance shows how well the response variable is predicted by a model that includes only the intercept β_0 while the residual deviance shows how well the response variable is predicted by our estimated LR model.

The ***Akaike Information Criterion (AIC)*** provides a method for assessing the quality of our model through comparison of related models. It is based on the deviance and intent to prevent us from including irrelevant predictors. The AIC number itself is not meaningful. If we have more than one similar candidate models (where all of the variables of the simpler model occur in the more complex models), then *we should select the model that has the smallest AIC.*

Example 3.1.2 (LR on Default Data with a Numeric Feature)**.** Consider the **Default** data set from R's ISLR package, where the response variable Default falls into one of two categories, Yes or No. Consider just using the predictor `balance` to estimate the probability of default using *logistic regression.* The R script to train the logistic regression model will give the estimates of the parameters and the hypothesis testing of the model.

Solution:

```
library(ISLR)
data(Default)
Prob.Default = as.numeric(Default$default=="Yes")
#pdf("t4-cly-011.pdf")
plot(Default$balance,Prob.Default,xlab="Balance",pch='+',
  xlim=c(0,2750),ylim=c(-0.2,1.2))
model = glm(default ~ balance, data=Default, family=binomial)
print(summary(model)) #print(coef(model))
newdata = data.frame(balance=seq(-1,2750,1))
newdata$Prob.of.Default = predict(model,newdata,type="response")
lines(Prob.of.Default ~ balance, newdata, col="green4", lwd=3)
```

Here's the statistical analysis (involving the Wald test Z-statistic) produced by R:

```
Call:
glm(formula = default ~ balance, family = binomial, data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2697  -0.1465  -0.0589  -0.0221   3.7589

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1596.5  on 9998  degrees of freedom
AIC: 1600.5
```

Example 3.1.3 (LR Predictive Model with a Numeric Feature)**.** Consider the **Default** data set with the statistical analysis given in Example 3.1.2, write down the mathematical formulation of the predictive model.

Example 3.1.4. Consider the logistic model for the **Default** data set in Example 3.1.3, predict the default probability for an individual with a balance of (a) \$1000, (b) \$2000.
(Answer: 0.00576, 0.586)

Example 3.1.5. Consider the **Default** data set with the results given in Example 3.1.2, compute the 95% confidence interval for β_0 and β_1.

Solution: Based on (3.5), there is an approximately 95% chance that the interval

$$[-10.6513 - 1.96 \cdot 0.3612,\ -10.6513 + 1.96 \cdot 0.3612] = [-11.3593, -9.9433]$$

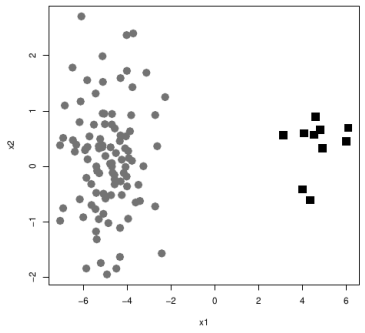
will contain the true value of β_0; and there is an approximately 95% chance that the interval

$$[0.0055 - 1.96 \cdot 0.0002, 0.0055 + 1.96 \cdot 0.0002] = [0.0051, 0.0059]$$

will contain the true value of β_1.

Example 3.1.6 (Completely Separable 2D Data)**.** Consider a simulated data below:

```
set.seed(2023)
cls1_x1 = rnorm(10, mean=5)
cls1_x2 = rnorm(10, mean=0)
cls1_y  = rep(0, length(cls1_x1))

cls2_x1 = rnorm(100, mean=-5)
cls2_x2 = rnorm(100, mean=0)
cls2_y  = rep(1, length(cls2_x1))

d.f = data.frame(x1=c(cls1_x1, cls2_x1),
                 x2=c(cls1_x2, cls2_x2),
                  y=c(cls1_y, cls2_y))
```



Fitting the data with logistic regression model leads to the following summary:

```
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred

Call:
glm(formula = y ~ ., family = binomial, data = d.f)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.231e-05   2.110e-08   2.110e-08   2.110e-08   2.774e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.085  23013.497   0.000    1.000
x1             -8.100   8369.842  -0.001    0.999
x2              0.199  19039.246   0.000    1.000

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.7020e+01  on 109  degrees of freedom
Residual deviance: 1.7484e-09  on 107  degrees of freedom
AIC: 6
```

- The p-value indicates that the features does not explain the output — This is due to the (3.4) being unbounded when the data is separable by a hyperplane.
- The small residual deviance indicates that the model fits OK. □

Example 3.1.7 (May 2022 Semester Final Exam, Q5(a))**.** The data from a study of low birth weight infants in a neonatal intensive care unit is used to examine the development of bronchopulmonary dysplasia (BPD), a chronic lung disease, in a sample of 223 infants weighing less than 1750 grams. The response variable is binary, denoting whether an infant develops BPD by day 28 of life (where BPD is defined by both oxygen requirement and compatible chest radiograph). Consider the trained logistic regression model is summarised below.

```
Call:
glm(formula = bpd ~ ., family = binomial, data=d.f.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0171  -0.6617  -0.3138   0.6840   2.2970

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 12.983392   3.797733   3.419 0.000629 ***
brthwght    -0.003521   0.001091  -3.227 0.001249 **
gestage     -0.314217   0.146186  -2.149 0.031601 *
toxemia     -2.268523   0.909026  -2.496 0.012576 *
---
Signif.:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 172.77  on 132  degrees of freedom
Residual deviance: 112.47  on 129  degrees of freedom
AIC: 120.47
```

```
Number of Fisher Scoring iterations: 5
```

In the logistic regression model, the output `bpd` is either 0 (representing no) or 1 (representing yes). The predictors are listed below:

- `brthwght`: birth weight in number of grams;
- `gestage`: gestational age in number of weeks;
- `toxemia`: a condition in pregnancy characterised by abrupt hypertension, albuminuria and edema. This is a binary variable with 0 = no, 1 = yes.

(i) Write down the mathematical expression of the logistic regression model for the given data with the coefficient values. (4 marks)

Solution: Let Y denote the `bpd`, X_1 denote `brthwght`, X_2 denote `gestage` and X_3 denote `toxemia`. The logistic model is

$$P(Y = 1|X_1 = x_1, X_2 = x_2, X_3 = x_3) = \frac{1}{1 + \exp(-\beta^T \boldsymbol{x})} \quad \text{[2 marks]}$$

where

$$\beta^T \boldsymbol{x} = 12.983392 - 0.003521x_1 - 0.314217x_2 - 2.268523x_3 \quad \text{[2 marks]}$$

(ii) For an infant with a birth weight of 1020 grams, with a gestational age is 29 weeks and the pregnancy does not have toxemia, by calculating the **conditional probability** of the logistic regression model, determine if the infant has a BPD problem. (7 marks)

(Answer: $P(Y = 1|X_1 = 1020, X_2 = 29, X_3 = 0) = 0.5694675 \geq 0.5$, has BPD problem)

(iii) According to the three-predictor model, `brthwght` has the lowest p-value. If only the predictor `brthwght` is used to fit the logistic regression model, the following analysis is obtained.

```
Call:
glm(formula = bpd ~ brthwght, family = binomial,
      data = d.f.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7404  -0.7517  -0.4019   0.8288   2.5192

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.6805219  0.9477546   4.939 7.87e-07 ***
```

```
brthwght     -0.0046773  0.0008549  -5.471 4.47e-08 ***
---
Signif.:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 172.77  on 132  degrees of freedom
Residual deviance: 128.67  on 131  degrees of freedom
AIC: 132.67

Number of Fisher Scoring iterations: 5
```

Investigate the infant with a birth weight of 1020 grams from part (ii) again by calculating the conditional probability to determine if the infant has a BPD problem. (4 marks)

(Answer: $P(Y = 1|X_1 = 1020) = 0.4774343$, no)

(iv) Comment on the results from part (ii) and part (iii) and use CRISP-DM framework to explain which model we should choose. Provide proper justification. (2 marks)

Solution: The conditional probabilities are different but around 0.5, so the two models provide different prediction. According CRISP-DM framework, the crucial decision on selecting a predictive model is based on validation. A good model should pass the cross-validation test. ..... [2 marks]

3.1.3 Qualitative Predictors

When a predictor (or factor) is **qualitative**, we need to introduce **dummy variable(s)** to turn it to numeric features. When it has C levels, $C - 1$ dummy variables will be created — this is called **(nearly) one-hot encoding**. Note that some books refer to the encoding of all levels as one-hot encoding. Here, we will remove the reference level because it is not essential in the computation.

Example 3.1.8 (Binary feature)**.** The feature "gender" has two levels 0 (male) and 1 (female), a dummy variable `gender1` is created with 0 as reference:

$$\text{gender1} = \begin{cases} 1, & \text{if gender} = 1 \\ 0, & \text{if gender} = 0 \end{cases}$$

Example 3.1.9 (Categorical feature with 4 levels)**.** The feature "blood type" has four levels A, AB, B, O and 3 dummy variables will be created with A as reference (R usually uses alphabetical ordering):

$$\text{bloodtypeAB} = \begin{cases} 1, & \text{if bloodtype} = \text{AB} \\ 0, & \text{if bloodtype} \neq \text{AB} \end{cases}, \quad \text{bloodtypeB} = \begin{cases} 1, & \text{if bloodtype} = \text{B} \\ 0, & \text{if bloodtype} \neq \text{B} \end{cases},$$

$$\text{bloodtypeO} = \begin{cases} 1, & \text{if bloodtype} = \text{O} \\ 0, & \text{if bloodtype} \neq \text{O} \end{cases}$$

Example 3.1.10 (One-Hot Encoding in R)**.** Consider the data X:

gender	bloodtype
0	A
1	B
1	AB
0	B
0	O

It can be encoded in R and the one-hot encoding can be computed using the `model.matrix` command.

```
d.f = data.frame(gender=c("0", "1", "1", "0", "0"),
                 bloodtype=c("A", "B", "AB", "B", "O"))
model.matrix(~ gender + bloodtype, d.f)
```

The one-hot encoded matrix of the data X is

```
  (Intercept) gender1 bloodtypeAB bloodtypeB bloodtypeO
1           1       0           0          0          0
2           1       1           0          1          0
3           1       1           1          0          0
4           1       0           0          1          0
5           1       0           0          0          1
```

Example 3.1.11 (One-Hot Encoding in Python)**.** In Python's patsy and statmodels modules, one-hot encoding is regarded as a kind of "Treatment Coding" or "Dummy Coding" which compares each level of the categorical feature to a base reference level (which is the value of the intersect).

```
import pandas as pd
df = pd.DataFrame({"gender" : ["0", "1", "1", "0", "0"],
                   "bloodtype" : ["A", "B", "AB", "B", "O"]})

### One-hot encoding in Python
from patsy.contrasts import Treatment
levels_gender = sorted(list(set(df.gender)))
levels_bloodt = sorted(list(set(df.bloodtype)))
contrast_gender = Treatment().code_without_intercept(levels_gender)
contrast_bloodt = Treatment().code_without_intercept(levels_bloodt)

### Model Matrix in Python
### Ref: https://stackoverflow.com/questions/48852930/replace-a-string-numpy-array-w
df_int = pd.DataFrame()
df_int['gender'], levels_gender = pd.factorize(df.gender)
df_int['bloodtype'], levels_bloodt = pd.factorize(df.bloodtype)

model_matrix = pd.DataFrame(contrast_gender.matrix[df_int.gender],
    columns=contrast_gender.column_suffixes).join(
  pd.DataFrame(contrast_bloodt.matrix[df_int.bloodtype],
    columns=contrast_bloodt.column_suffixes) )
print("Model matrix for gender and bloodtype in Python:\n", model_matrix)
```

Note: Apart from the on-hot encoding, Python's patsy offers simple coding `Simple()`, sum (deviation) coding `Sum()`, backward difference coding `Diff()`, Helmert coding `Helmert()` and orthogonal polynomial coding `Poly()`.

When a qualitative predictor X_j has C levels, logistic regression (3.2) can be generalised to work with X_j by working with the $(C-1)$ **dummy variables** $X_j.\text{level2}, \cdots, X_j.\text{level}C$

$$\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \frac{1}{1+\exp(-(\beta_0+\cdots+\beta_j^{(2)}x_j.\text{level2}+\cdots+\beta_j^{(C)}x_j.\text{level}C+\cdots))}$$

where

$$x_j.\text{level}\ell = \begin{cases} 1, & x_j = \text{level } \ell, \\ 0, & \text{otherwise}, \end{cases} \quad \ell = 2, \cdots, C.$$

3.1.4 Statistical Meaning of β_j

Rearranging (3.2) gives the **odds** on the right-hand side of

$$\frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=0|\boldsymbol{X}=\boldsymbol{x})} = \frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{1-\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})} = \exp(\tilde{\boldsymbol{x}}^T\boldsymbol{\beta}). \tag{3.6}$$

Taking the logarithm of both sides of (3.6), we obtain the **log-odds** or **logit**, which is linear with respect to the input features:

$$\ln \frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{1-\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})} = \beta_0+\beta_1X_1+\cdots+\beta_pX_p. \tag{3.7}$$

In this case, the logit of Y can be inferred linearly from inputs $\boldsymbol{X}$, i.e. a unit increase in x_j changes the logit by β_j.

If we only change a feature X_j from a to b and keep other features fixed, we can characterise β_j by the **odds ratio**

$$\text{OR} = \frac{\frac{\mathbb{P}(Y=1|X_j=b)}{\mathbb{P}(Y=0|X_j=b)}}{\frac{\mathbb{P}(Y=1|X_j=a)}{\mathbb{P}(Y=0|X_j=a)}} = \frac{\exp(\cdots+\beta_j\cdot b+\cdots)}{\exp(\cdots+\beta_j\cdot a+\cdots)} = \exp(\beta_j(b-a)).$$

The coefficient $\beta_j \neq 0$ of (3.2) can influence the probability of $Y=1$ in the following way.

β_j	OR	Relative probability of $\mathbb{P}(Y=1\|x_j=x)$	Probability to be classified into $Y=1$
>0	>1	Higher	$X_j=b$ has higher chance than $X_j=a$
<0	<1	Lower	$X_j=a$ has higher chance than $X_j=b$

Note that $b=1$ and $a=0$ for the dummy variables of categorical data.

Example 3.1.12 (LR on Default Data with a Binary Feature)**.** Suppose that the **Default** data set is now depending on another qualitative predictor, `student`. The R script to fit the logistic model is listed below.

```
library(ISLR)
data(Default)
glm.model = glm(default ~ student, data=Default, family=binomial)
print(summary(glm.model))
```

and the output is shown below.

```
Call:
glm(formula = default ~ student, family = binomial, data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.2970  -0.2970  -0.2434  -0.2434   2.6585
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 2908.7  on 9998  degrees of freedom
AIC: 2912.7

Number of Fisher Scoring iterations: 6
```

(a) Compare the probability of default for a student with a non-student. Explain. (Hint: Use the sign of the model coefficient for student)

(b) Predict the probability of default for (i) student (ii) non-student. (Answer: 0.043139, 0.029195)

Example 3.1.13 (LR on Default Data with a Mix of Numeric and Categorical Features)**.** Suppose that the **Default** dataset is now depending on three predictors, `balance`, `income` and `student`. The result from logistic regression is shown below.

```
Call:
glm(formula = default ~ balance + income + student, family = binomial,
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4691  -0.1418  -0.0557  -0.0203   3.7383

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

Discuss the results involving the significance of each variable based on the coefficients and then the odds.

Solution: Based on the p-value (with 5% significance level), we find that `balance` and `student` are **probably significant** while `income` is **probability insignificant**. A reduced model could be fitted and then compare with the full model to see if a reduced model is better.

The odds of the default

- increases with the balance because $\beta_1 = 5.737 \times 10^{-3} > 0$;
- increases with the income because $\beta_2 = 3.033 \times 10^{-6} > 0$. However, the p-value > 0.05 does not rule out $\beta_2 = 0$ based on the hypothesis testing;
- is lower for students compare to non-students.

Example 3.1.14 (Final Exam Jan 2019, Q4(b))**.** You are a data analytics consultant to a property investing firm. You task is to help the investment firm make money through property price arbitrage — identify properties that are selling at a lower premium. There is a property listing site that aggregates ready-to-buy properties and quoted prices across the country. You have accumulated data for the properties sold this month along with the features of these properties. Descriptions of the data are shown below:

Dist_train	Distance to nearest train station from the property
Dist_market	Distance to nearest grocery market from the property
Dist_school	Distance to nearest school from the property
Carpet	Carpet area of the property in square feet
Builtup	Built-up area of the property in square feet
Parking	Type of car parking available with the property (Covered; No; Open)
City	Categorization of the city based on population size (A; B; C)
Rainfall	Annual rainfall in the area where property is located
House_price	Is a property sold underpriced? (yes=1; no=0)

A logistic regression model is fitted and results are shown in Table Q4(b).

	Coefficient	Std. error	Z-statistic	p-value
Intercept	0.0054	4.40E+03	12.254	< 2e-16
Dist_train	51.7124	3.1921	1.624	0.1048
Dist_market	16.3212	2.5077	0.651	0.5153
Dist_school	32.7063	3.5612	0.919	0.3586
Carpet	0.5900	4.19E+02	-1.409	0.1592
Builtup	0.5660	3.49E+02	1.621	0.1055
Parking_No	-0.0660	1.65E+03	-4.008	6.87E-05
Parking_Open	0.5165	1.48E+03	-3.816	0.0002
City_B	-0.0408	1.33E+03	-3.068	0.0022
City_C	-0.0176	1.15E+03	-15.271	< 2e-16
Rainfall	0.0279	1.29E+03	-21.681	< 2e-16

Table Q4(b)

With 5% significance,

(i) Comment on the significance of each variable. (2 marks)

Solution: Only Parking, City and Rainfall are significant since the p-value is less than significance of 0.05. [2 marks]

Remark: This is a simplified question. In reality, we must check if the regression model fits the data, otherwise, the significance variables are meaningless.

(ii) In a real-world problem, we will refit the data to a reduced model. However, in this exam environment, you may just pick the coefficients from the full model and then write down the reduced/selected-feature logistic model based on the full model. (2 marks)

Solution: Never-ever write the feature-selected model this way in the real-world calculation and my exam. This answer is used by the previous lecturer because R is not allowed in final exam: The "reduced" logistic regression model based on the p-values are approximately:

$$\mathbb{P}(\text{House\_Price} = 1|\boldsymbol{X}) = \frac{1}{1+\exp(-(0.0054 + \boldsymbol{w}^T\boldsymbol{X})}$$

where
$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{X} = &-0.0660(\text{Parking\_No}) + 0.5165(\text{Parking\_Open})\\ &- 0.0408(\text{City\_B}) - 0.0176(\text{City\_C}) + 0.0279(\text{Rainfall}).\end{aligned}$$

(iii) Calculate the odds and compare the probability of underprice for houses with different types of parking based on the reduced model. (5 marks)

Solution: First, note that $Y = 1$ refers to the property sold is underpriced.

There are two **dummy variables** related to the input parking — Parking_No, Parking_Open. They are using Covered as comparison.

Odds ratio	Odds vs Odds	Prob vs Prob
$\frac{odds(Parking=No)}{odds(Parking=Covered)} = e^{-0.0660} = 0.9361 < 1$	Odds of $Y = 1$ is higher for Parking=Covered	Probability of underprice for the house with covered parking will be higher than no parking.
$\frac{odds(Parking=Open)}{odds(Parking=Covered)} = e^{0.5165} = 1.6762 > 1$	Odds of $Y = 1$ is higher for Parking=Open	Probability of underprice for the house with open parking will be higher than covered parking.

Combining the results from the table, we have

$$odds(Parking = No) < odds(Parking = Covered) < odds(Parking = Open).$$

The ordering of probability of the property sold is underprice ($Y = 1$) based on parking is

$$No < Covered < Open$$

(iv) Calculate the odds and compare the probability of underprice for houses with different types of city. (5 marks)

(v) State a possible issue that might be found in the data. Suggest a more suitable solution to solve the stated problem. (2 marks)

Solution: Problem: High correlation between builtup area and carpet area.

Solution: Use PCA instead of original variables.

(vi) State the purposes of principal components. Discuss how principal component achieve the stated purposes. (4 marks)

Solution: Reduce dimension. Consider only the first few principal components which contributes most of the variation.

Avoid correlation. Each principle component is designed to be perpendicular. Hence, there will be no correlation.

Example 3.1.15 (Final Exam May 2019, Q2)**.** (a) The human resource department would like to determine potential employees for promotion. You have collected some data from previous employee promoting records as described below:

exp	Number of years of experience working in the company
sal_mth	Average monthly salary in last 12 months
sal_yr	Yearly salary in last 12 months
pjt	Is there any project involved? [Yes; No]
dpmt	Department [A; B; C; D]
emp_id	Employee ID
promote	Is the employee getting promoted? [Yes=1; No=0]

A logistic regression has been constructed to predict the promotion of an employee. Table Q2(a) shows parts of the results of the logistic regression.

	Coefficient	P-value
Intercept	0.0035	<2e-16
exp_yr	0.7124	<2e-16
sal_mth	-0.0212	0.0057
sal_yr	-0.0363	0.0086
pjt_Yes	0.0330	0.2479
dpmt_B	1.0447	0.0002
dpmt_C	-1.5318	6.87e-05
dpmt_D	2.1539	0.0017
emp_id	-0.0279	0.5245

Table Q2(a)

(i) Write the logistic regression model that compute the probability that an employee get promoted, $\mathbb{P}(Y = 1)$. (3 marks)

(ii) Calculate the odds and compare the probability of promotion for employee with 7 years of working experience and an employee with 2 years of working experience. (3 marks)

(Answer: $odds(Y = 1|exp\_yr = 7) = 35.2336\, odds(Y = 1|exp\_yr = 2)$, 7-year experience has a higher probability than the 2-year experience)

(iii) Calculate the odds and compare the probability of promotion for employee in different departments. Arrange the probability of promotion of department from lowest to highest. (8 marks)

(c) State two possible issues found in the data. Suggest a suitable solution for each of the issue stated. (4 marks)

Solution: Issue 1: emp_id should not be a variable as this shall not affect the result.
Solution 1: Remove emp_id and label as index variable.
Issue 2: sal_mth and sal_yr are highly correlated.
Solution 2: Remove one of the variables / Perform PCA.

3.1.5 Model Evaluation and Classifier Boundary

Example 3.1.16 (Final Exam Jan 2021, Q2(b))**.** The **testing dataset** of a social network advertisement is given in Table 2.2. The variables "Gender", "Age" and "EstimatedSalary" are the predictors and the variable "Purchased" is the response. The "Gender" is a binary categorical data with levels "Male" and "Female", the "Age" and the "EstimatedSalary" are quantitative data. The "Purchased" is a binary response with values 0 (representing "no purchase", assuming **0 is the positive class**) and 1 (representing "purchase").

Table 2.2: The testing data of a social network advertisement.

Gender	Age	EstimatedSalary	Purchased
Male	29	80000	0
Male	45	26000	1
Female	48	29000	1
Male	45	22000	1
Female	47	49000	1
Male	48	41000	1
Male	46	23000	1
Male	47	20000	1
Male	49	28000	1
Female	47	30000	1

Suppose a logistic regression model is trained and the coefficients are stated in Figure 2.1.

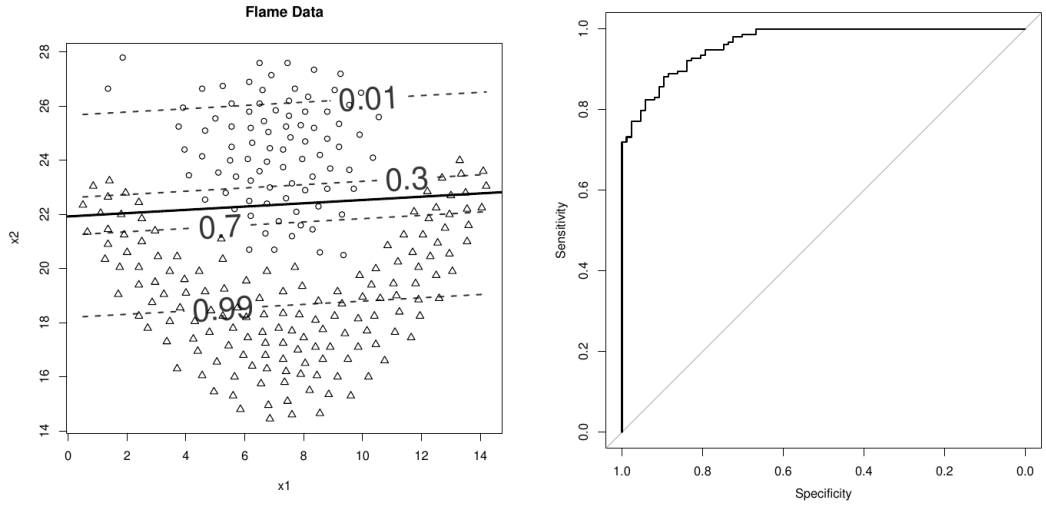
Figure 2.1: The coefficients of the logistic regression based on an insurance claim data.

```
glm(formula=Purchased~., family=binomial, data=data.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9882  -0.5640  -0.1372   0.5532   2.1820

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.188e+01  2.497e+00  -4.757 1.96e-06 ***
GenderMale       4.221e-01  5.927e-01   0.712 0.476319
Age              2.178e-01  4.751e-02   4.584 4.56e-06 ***
EstimatedSalary  3.868e-05  1.001e-05   3.863 0.000112 ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 135.37  on 99  degrees of freedom
Residual deviance:  74.91  on 96  degrees of freedom
```

Write down the **mathematical formula** of the logistic regression model and then use it to **predict** the variable “Purchase” of the insurance data in Table 2.2 as well as **evaluating** the performance of the model by calculating the confusion matrix, accuracy, sensitivity, specificity, PPV, NPV of the logistic model (assuming 0 is the positive class). [**Note**: The default cut-off is 0.5] (5 marks)
(Hint: Use Excel)

Example 3.1.17 (Decision Boundaries and ROC Curve for (Milely Nonlinear) Flame Data)**.** For the "flame" data, the "boundary" of the classifier is shown in the left figure below as the solid line:



The dashed lines correspond to different "cut-off" 0.01, 0.3, 0.7 and 0.99.

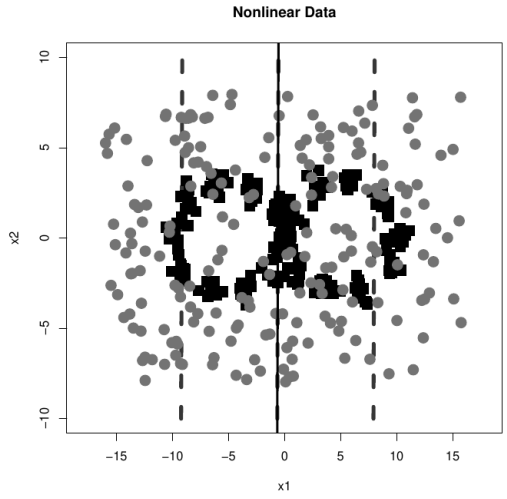
The ROC curve can be understood as the result of varying the "cut-off" and calculating the "sensitivity" (TPR) and "specificity" mentioned in Topic 1. If we calculate out, we have

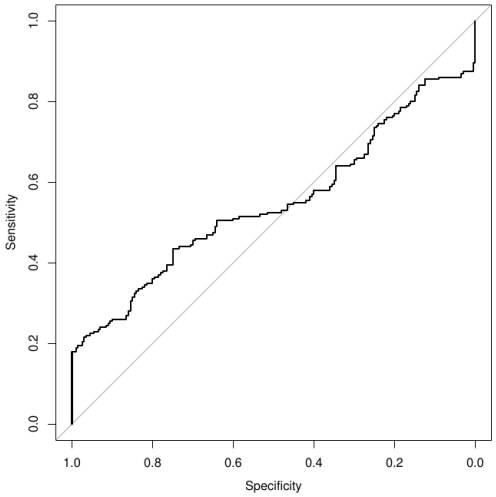
	0.01		0.3		0.7		0.99	
Predicted	1	2	1	2	1	2	1	2
1	19	0	64	6	79	23	87	80
2	68	153	23	147	8	130	0	73
	TPR = 0.2184	FPR = 0	0.7356	0.0392	0.9080	0.1503	1	0.5229

Example 3.1.18 (Decision Boundaries and ROC Curve for LR with Strongly Nonlinear Data)**.** Consider a simulated data generated using the following R script:

```
t = seq(-pi, pi-0.5, length=10)
deterministic = data.frame(x1=c(5*(1+cos(t)),-5*(1+cos(t))),
                           x2=c(   3*sin(t),     3*sin(t)))
x1min = -16; x1max =  16; x2min = -8; x2max =  8
sdev  = 0.5
n = nrow(deterministic)
m = 10
set.seed(2023)
d.f    = data.frame(x1 =
           c(rep(deterministic$x1, each=m) + rnorm(n*m, sd=sdev),
             runif(n*m, min=x1min, max=x1max))
           )
d.f$x2 = c(rep(deterministic$x2, each=m) + rnorm(n*m, sd=sdev),
             runif(n*m, min=x2min, max=x2max))
d.f$y  = c(rep(0,n*m), rep(1,n*m))
```

The decision boudaries is going to be impossible to obtain because the probabilities ranges from around 0.4 to around 0.6 and the ROC curve definitely says that logistic regression model does not fit the model.





Example 3.1.19 (Model Validation using LOOCV)**.** Consider the weather data used in the book Witten et al. [2011] (in Weka arff format from `http://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/weather.arff`). Write an R script to test it using LOOCV.

Solution: A simple script is given below.

```
library(foreign)
d.f = read.arff("weather.arff")
errors = NULL
for(i in 1:nrow(d.f)) {
    d.f.test = d.f[ i,]
    d.f.tran = d.f[-i,]   # Leave-one-out
    logreg.model = glm(play~., family=binomial(link='logit'),
        data=d.f.tran, control=list(maxit=50))
    play.p = predict(logreg.model, newdata=d.f.test, type='response')
    play.p = ifelse(play.p > 0.5,"yes","no")
    errors[i] = (play.p!=d.f.test$play)
}
cat("error rate =", 100*sum(errors)/length(errors), "%\n")
```

Not only that the error rate is 35.71% (high) but the coefficients in the logistic models are all having p-value much larger than 5% which indicates that logistic model is not suitable for modelling the weather data.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   40.4086  4370.5739   0.009    0.993
outlookrainy -20.7215  4370.5223  -0.005    0.996
outlooksunny -21.4922  4370.5222  -0.005    0.996
temperature   -0.0739     0.1957  -0.378    0.706
humidity      -0.1517     0.1229  -1.235    0.217
windyTRUE     -3.6220     2.9590  -1.224    0.221

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 16.0483  on 12  degrees of freedom
Residual deviance:  8.4622  on  7  degrees of freedom
AIC: 20.462
```

3.2 Generalised Linear Model

Logistic regression model (3.2) have three classes of generalisations:

1. **Generalised linear model (GLM)** [McCullagh and Nelder, 1989, Kutner et al., 2005], i.e. Y is related to $\beta_0 + \beta_1 X_1 + ... + \beta_p X_p$. For logistic regression, Y follows binomial model; for linear regression, Y follows Gaussian distribution. In general, Y can follow a known parametric statistical model. The R's implementation is shown on Page 68.
2. **Generalised additive models (GAM)** [Hastie et al., 2008, Section 9.1] $\mathbb{E}[Y|X_1, \cdots, X_p] = \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$.
3. **Multinomial logistic regression (MLR)** and **artificial neural network (ANN)**

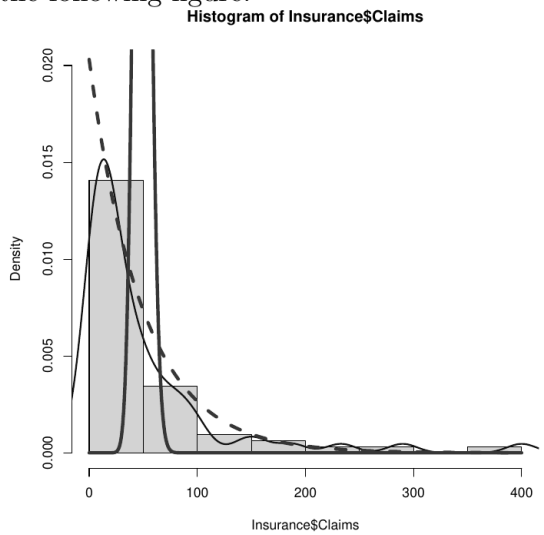
We will use an insurance data considered in the book "Modern Applied Statistics with S-PLUS" (MASS) to illustrate GLM with Poisson distribution.

Example 3.2.1. Consider the `Insurance` data consisting of the numbers of policyholders of an insurance company who were exposed to risk, and the numbers of car insurance claims made by those policyholders in the third quarter of 1973 from the MASS textbook and use (i) Poisson model, (ii) exponential model and (iii) GLM model to fit the data.

Solution: The `Insurance` data consists 4 inputs and 1 output:

- District: it is a categorical data representing the district of residence of policyholder (1 to 4) which represents 4 is major cities;
- Group: it is an ordered data representing the group of car with levels $<$ 1 litre, 1–1.5 litre, 1.5–2 litre, $>$ 2 litre;
- Age: it is an ordered data representing the age of the insured in 4 groups labelled $<$ 25, 25–29, 30–35, $>$ 35;
- Holders: it is an integral data representing the numbers of policyholders;
- Claims: it is an integral data representing the numbers of claims.

The fittings using (i) Poisson model and (ii) exponential model on the Claims are shown in the following figure.



The fitting of the Claims with the rest as inputs using GLM with Poisson leads to the following estimate similar to logistic regression:

```
Call:
glm(formula = Claims ~ District + Group + Age + offset(log(Holders)),
    family = poisson, data = Insurance)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.46558  -0.50802  -0.03198   0.55555   1.94026

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.810508   0.032972 -54.910  < 2e-16 ***
District2    0.025868   0.043016   0.601 0.547597
District3    0.038524   0.050512   0.763 0.445657
District4    0.234205   0.061673   3.798 0.000146 ***
Group.L      0.429708   0.049459   8.688  < 2e-16 ***
Group.Q      0.004632   0.041988   0.110 0.912150
Group.C     -0.029294   0.033069  -0.886 0.375696
Age.L       -0.394432   0.049404  -7.984 1.42e-15 ***
Age.Q       -0.000355   0.048918  -0.007 0.994210
Age.C       -0.016737   0.048478  -0.345 0.729910
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 236.26  on 63  degrees of freedom
Residual deviance:  51.42  on 54  degrees of freedom
AIC: 388.74

Number of Fisher Scoring iterations: 4
```

A sample R code to obtain them is shown below.

```
# Venables, W. N. and Ripley, B. D. (2002)
# Modern Applied Statistics with S-PLUS. 4th Edition. Springer.
library(MASS)   # for 'Insurance' data
print(summary(Insurance))
#
# N = Number of Claims
# X_i = Loss amount
# S = Severity, X_1 + X_2 + ... X_N
# (i)  Fit data using Poisson model on the Claims
mle.pois = fitdistr(Insurance$Claims, densfun="poisson")
# (ii) Fit data using Exponential model on the Claims
mle.expn = fitdistr(Insurance$Claims, densfun="exponential")
# Show the result of the fitting in histogram
x = 0:400
hist(Insurance$Claims, freq=FALSE, ylim=c(0,0.02))
lines(density(Insurance$Claims), col="blue", lwd=2)
lines(x,dpois(x,lambda=mle.pois$estimate), col="red", lwd=4, lty=1)
lines(x,dexp(x,rate=mle.expn$estimate), col="red", lwd=4, lty=2)

# (iii) Fit data using Generalised Linear Model with offset
mod.glm = glm(Claims ~ District + Group + Age + offset(log(Holders)),
    data = Insurance, family = poisson)
print(summary(mod.glm))
```

3.3 Multinomial Logistic Regression

Multinomial probit or `https://en.wikipedia.org/wiki/Multinomial_logistic_regression` is a classification method that generalises the LR to multiclass problems.

We apply the odds analysis to the output:

$$\begin{cases} \ln \dfrac{\mathbb{P}(Y=2|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})} = \boldsymbol{\beta}_2 \cdot \boldsymbol{x} \\ \ln \dfrac{\mathbb{P}(Y=3|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})} = \boldsymbol{\beta}_3 \cdot \boldsymbol{x} \\ \cdots\cdots \\ \ln \dfrac{\mathbb{P}(Y=K|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})} = \boldsymbol{\beta}_K \cdot \boldsymbol{x} \end{cases}$$

Since the sum of the probability of the K-class outputs is one:

$$\begin{aligned} &\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x}) + \mathbb{P}(Y=2|\boldsymbol{X}=\boldsymbol{x}) + \cdots + \mathbb{P}(Y=K|\boldsymbol{X}=\boldsymbol{x}) = 1 \\ \Rightarrow\ &\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x}) + \mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})e^{\boldsymbol{\beta}_2\cdot\boldsymbol{x}} + \cdots + \mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})e^{\boldsymbol{\beta}_K\cdot\boldsymbol{x}} = 1, \end{aligned}$$

we obtain the **multinomial LR model**:

$$\begin{aligned} \mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x}) &= \frac{1}{1+\sum_{k=2}^{K} e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}}} \\ \mathbb{P}(Y=j|\boldsymbol{X}=\boldsymbol{x}) &= \frac{e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}}{1+\sum_{k=2}^{K} e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}}}, \quad j=2,\cdots,K. \end{aligned} \tag{3.8}$$

Note that it becomes LR when $K=2$ (we change the labels from 1,2 to 0,1 for LR).

The multinomial LR model requires more data compare to LR, so when we have little data, this model won't work. It is implemented in R and Python as listed below.

```
library(nnet)                    # R
multinom(formula, data, weights, subset, na.action,
         contrasts = NULL, Hess = FALSE, summ = 0,
         censored = FALSE, model = FALSE, ...)
```

```
### Easier to use (following https://www.andrewvillazon.com/logistic-regression-pyth
import statsmodels.formula.api as smf
m = smf.logit("Y ~ X1 + X2 + X3", data=d_f).fit()
print(m.summary())
print("Odds ratios:\n", np.exp(m.params))
### More complex to use, need to work with pasty
import statsmodels.api as sm    # Python
sm.GLM(endog, exog, family=None, offset=None,
  exposure=None, freq_weights=None,
  var_weights=None, missing='none', **kwargs)
sm.MNLogit(endog, exog, check_rank=True, **kwargs)
import pasty
y, X = patsy.dmatrices("Y ~ X1+X2+X3", data=d_f, return_type="dataframe")
```

Note that `endog` stands for endogenous response variable while `exog` stands for exogenous variables (Section 1.7.3).

Example 3.3.1. Analyse the iris flower data using multinomial LR.

Solution: Let $Y = 1$ be `setosa`, $Y = 2$ be `versicolor`, $Y = 3$ be `virginica` and X_1 be `Sepal.Length`, X_2 be `Sepal.Width`, X_3 be `Petal.Length`, X_4 be `Petal.Width`.

The R script to fit the data is listed below.

```
library(nnet)
m = multinom(Species ~ ., data=iris)  # multinomial LR
print(summary(m))
```

The coefficients of the trained model are listed below.

```
Call:
multinom(formula = Species ~ ., data = iris)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    18.69037    -5.458424   -8.707401     14.24477   -3.097684
virginica    -23.83628    -7.923634  -15.370769     23.65978   15.135301

Std. Errors:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    34.97116     89.89215    157.0415     60.19170    45.48852
virginica     35.76649     89.91153    157.1196     60.46753    45.93406

Residual Deviance: 11.89973
AIC: 31.89973
```

The mathematical formulation for the iris flower data with $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$ according to (3.8) is

$$\begin{aligned}
\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) &= \frac{1}{D} \\
\mathbb{P}(Y = 2|\boldsymbol{X} = \boldsymbol{x}) &= \frac{e^{18.69037 - 5.458424x_1 - 8.707401x_2 + 14.24477x_3 - 3.097684x_4}}{D} \\
\mathbb{P}(Y = 3|\boldsymbol{X} = \boldsymbol{x}) &= \frac{e^{-23.83628 - 7.923634x_1 - 15.370769x_2 + 23.65978x_3 + 15.135301x_4}}{D}
\end{aligned}$$

where the denominator is

$$\begin{aligned}
D = 1 &+ e^{18.69037 - 5.458424x_1 - 8.707401x_2 + 14.24477x_3 - 3.097684x_4} \\
&+ e^{-23.83628 - 7.923634x_1 - 15.370769x_2 + 23.65978x_3 + 15.135301x_4}
\end{aligned}$$

3.3.1 ElasticNet MLR

A generalisation of MLR is the `https://en.wikipedia.org/wiki/Elastic_net_regularization` which is used in Python.

```
library(glmnet)                    # R
glmnet(x, y, family = c("gaussian", "binomial", "poisson",
  "multinomial", "cox", "mgaussian"), weights = NULL, offset = NULL,
  alpha = 1, nlambda = 100,
  lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL, standardize = TRUE, intercept = TRUE,
  thresh = 1e-07, dfmax = nvars + 1,
  pmax = min(dfmax * 2 + 20, nvars), exclude = NULL,
  penalty.factor = rep(1, nvars),
  lower.limits = -Inf, upper.limits = Inf, maxit = 1e+05,
  type.gaussian = ifelse(nvars < 500, "covariance", "naive"),
  type.logistic = c("Newton", "modified.Newton"),
  standardize.response = FALSE,
```

```
  type.multinomial = c("ungrouped", "grouped"),
  relax = FALSE, trace.it = 0, ...)
```

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *,
  dual=False, tol=0.0001, C=1.0, fit_intercept=True,
  intercept_scaling=1, class_weight=None, random_state=None,
  solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
  warm_start=False, n_jobs=None, l1_ratio=None)  # C -> oo, mode -> LR
```

Python used the name `sklearn.linear_model.LogisticRegression` instead of ElasticNet and is misleading. The coefficient `C` is the control of regularisation on the GLM.

According to `https://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/`, the following are things to consider when applying the multinomial LR model:

- The Independence of Irrelevant Alternatives (IIA) assumption: Roughly, the IIA assumption means that adding or deleting alternative outcome categories does not affect the odds among the remaining outcomes. There are alternative modelling methods, such as alternative-specific multinomial probit model, or nested logit model to relax the IIA assumption.
- Diagnostics and model fit: Unlike logistic regression where there are many statistics for performing model diagnostics, it is not as straightforward to do diagnostics with multinomial logistic regression models. For the purpose of detecting outliers or influential data points, one can run separate logit models and use the diagnostics tools on each model.
- Sample size: Multinomial regression uses a maximum likelihood estimation method, it requires a large sample size. It also uses multiple equations. This implies that it requires an even larger sample size than ordinal or binary logistic regression.
- Complete or quasi-complete separation: Complete separation means that the outcome variable separate a predictor variable completely, leading perfect prediction by the predictor variable.
- Perfect prediction means that only one value of a predictor variable is associated with only one value of the response variable. But you can tell from the output of the regression coefficients that something is wrong. You can then do a two-way tabulation of the outcome variable with the problematic variable to confirm this and then rerun the model without the problematic variable.
- Empty cells or small cells: You should check for empty or small cells by doing a cross-tabulation between categorical predictors and the outcome variable. If a cell has very few cases (a small cell), the model may become unstable or it might not even run at all.

3.4 Feed-forward ANN / MLP

Artificial neural networks (ANNs, `https://en.wikipedia.org/wiki/Artificial_neural_network` are machine learning models that are the foundation of generative AI (GenAI, large language models (LLM) is a special case) and artificial general intelligence (AGI) which are built using principles of neuronal organisation discovered by connectionism in the biological neural networks constituting animal brains.

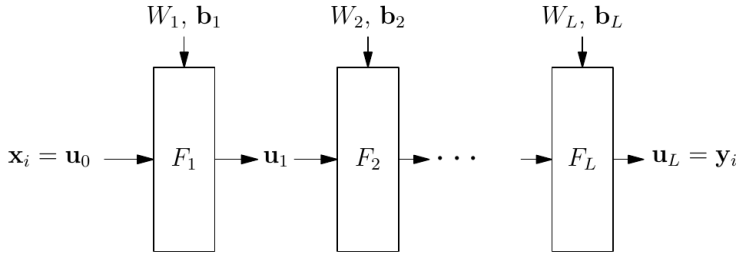
A **feed-forward ANN** is known as as `https://en.wikipedia.org/wiki/Multilayer_perceptron` **(MLP)** is a generalisation of the multinomial LR.

A MLP with input $\boldsymbol{x}_i \in \mathbb{R}^p$ and output is $\boldsymbol{y}_i \in \mathbb{R}^m$:

$$\begin{aligned} \boldsymbol{u}_1 &= F_1(W_1\boldsymbol{u}_0 + \boldsymbol{b}_1), \quad \boldsymbol{u}_0 = \boldsymbol{x}_i \\ \boldsymbol{u}_2 &= F_2(W_2\boldsymbol{u}_1 + \boldsymbol{b}_2) \\ &\dots \\ \hat{\boldsymbol{y}}_i = \boldsymbol{u}_L &= F_L(W_L\boldsymbol{u}_{L-1} + \boldsymbol{b}_L). \end{aligned} \tag{3.9}$$

where L is the number of layers of ANN (with $L-1$ hidden layers).

Horizontal pictorial representation:



When $L = 1$, we obtain a `https://en.wikipedia.org/wiki/Perceptron`:

$$\boldsymbol{u}_1 = F_1(W_1\boldsymbol{x}_i + \boldsymbol{b}_1). \tag{3.10}$$

We can see that when $m = 1$, $\boldsymbol{b}_1 = \beta_0$, $W_1 = (\beta_1, \dots, \beta_p)$ and $F_1(x) = S(x)$, we obtain the LR. When $m = K - 1$ ($K \geq 2$), we obtain the multinomial LR (3.8) which is how the R's `nnet::multinom` is implemented.

Theorem 3.4.1. *ANNs are **universal approximators** (like **support vector machines (SVMs)** [Hammer and Gersmann, 2003], **Gaussian processes (GPs) with a RBF kernel** [Sotiropoulos and Asada, 2020], Kolmogorov-Arnold network [Schmidt-Hieber, 2021], ...). Theoretically, a ANN with one hidden layer is as expressive as one with many hidden layers in practice if many nodes are used.*

When $L = 2$, we obtain an ANN with a single hidden-layer.

$$\begin{aligned} \boldsymbol{u}_1 &= F_1(W_1\boldsymbol{x}_i + \boldsymbol{b}_1) \\ \boldsymbol{y} = \boldsymbol{u}_2 &= F_1(W_2\boldsymbol{u}_1 + \boldsymbol{b}_2). \end{aligned} \tag{3.11}$$

Single-Hidden-Layer NN in R

```
nnet(x, y, weights, size, Wts, mask,
     linout = FALSE, entropy = FALSE, softmax = FALSE,
     censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,
     maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,
     abstol = 1.0e-4, reltol = 1.0e-8, ...)
```

MLP in R

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
  stepmax = 1e+05, rep = 1, startweights = NULL,
  learningrate.limit = NULL, learningrate.factor = list(minus=0.5,
  plus = 1.2), learningrate = NULL, lifesign = "none",
  lifesign.step = 1000, algorithm = "rprop+", err.fct = "sse",
  act.fct = "logistic", linear.output = TRUE, exclude = NULL,
  constant.weights = NULL, likelihood = FALSE)
```

MLP in Python

```
class sklearn.neural_network.MLPClassifier(
  hidden_layer_sizes=(100,), activation='relu', *, solver='adam',
  alpha=0.0001, batch_size='auto', learning_rate='constant',
  learning_rate_init=0.001, power_t=0.5, max_iter=200,
  shuffle=True, random_state=None, tol=0.0001, verbose=False,
  warm_start=False, momentum=0.9, nesterovs_momentum=True,
  early_stopping=False, validation_fraction=0.1, beta_1=0.9,
  beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)
```

The algorithm to estimate the parameters W_ℓ and $\boldsymbol{b}_\ell$ for the layer $\ell = 1, \ldots, L$ is the improvement of **back-propagation algorithm**:

1. $t = 0$;
2. Using the guess parameters $W_\ell^{(t)}$, $\boldsymbol{b}_\ell^{(t)}$, calculate all the intermediate states

$$\boldsymbol{u}_\ell^{(t)} = F_\ell(W_\ell^{(t)} \boldsymbol{u}_{\ell-1}^{(t)} + \boldsymbol{b}_\ell^{(t)})$$

   and the output $\hat{\boldsymbol{y}}_i$;
3. The output layer

$$\delta_L = \hat{\boldsymbol{y}}_i - \boldsymbol{y}_i$$

4. Back-Propagation (roughly): For ℓ from L to 1, do

$$\delta_{\ell-1} = \frac{\partial F_\ell}{\partial W_\ell}(\boldsymbol{u}_{\ell-1}^{(t)})\delta_\ell$$
$$W_\ell^{(t+1)} = W_\ell^{(t)} + \alpha \times \boldsymbol{u}_{\ell-1}^{(t)} \times \delta_{\ell-1}$$

5. $t = t + 1$ and go to step 2.

ANNs learn lots of parameters and therefore are prone to overfitting. This is not necessarily a problem as long as regularisation is used. Two popular **reglarisers** are the following:

- Weight Decay: Use ℓ_2 regularisation on all weights (including bias terms).
- Dropout: For each input (or mini-batch) randomly remove each hidden node with probability p (e.g. $p = 0.5$) these nodes stay removed during the backprop pass, however are included again for the next input. It is theoretically related to bagging (Section 7.1.1).

In the training of ANN, local minima need to be avoided by

- Using momentum:

$$\nabla \boldsymbol{w}_t = \Delta \boldsymbol{w}_t + \mu \nabla \boldsymbol{w}_{t-1}, \quad \boldsymbol{w} = \boldsymbol{w} - \alpha \nabla \boldsymbol{w}_t$$

  i.e. still use some portion of previous gradient to keep you pushing out of small local minima.
- Initialising weights cleverly (not all that important). For example, use Autoencoders for unsupervised pre-training.
- Using ReLU instead of sigmoid/tanh (weights don't saturate).

Other tricks and tips in the training of ANN are

- Rescale the data so that all features are within [0,1];
- Lower learning rate;
- Use mini-batch, i.e. stochastic gradient descent with maybe 100 inputs at a time – make sure you shuffle inputs randomly first;
- For image input, use convolution neural network.

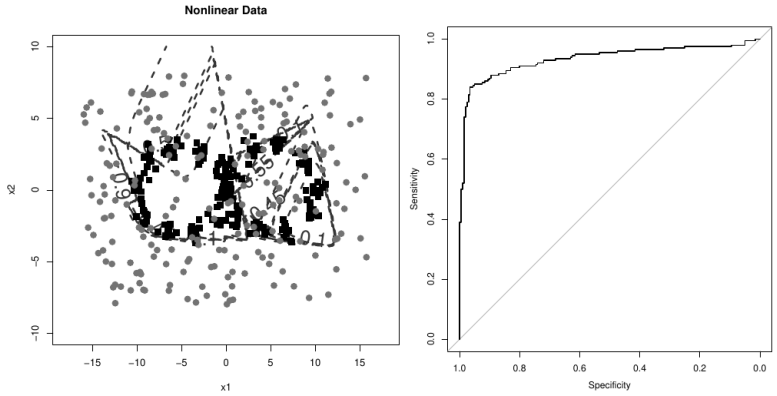
Example 3.4.2 (Decision Boundaries and ROC Curve for ANN with Strongly Nonlinear Data)**.** Consider the data from Example 3.1.18. If we use neural network instead of LR, we will face the following problems:

- What neural network to use? 1 hidden layer? 2? 3?
- If we pick 1-hidden layer, how many internal nodes should we have? 5? 10?
- What should the initial guess of the parameters be? I tried the same model with 1 hidden layer, 10 nodes, it sometimes go OK, and some times cannot converge because I forgot to set seed. So far, seed=1 is OK, seed=2023 and seed=202301 are not OK.

The R script is as follows:

```
library(neuralnet)
set.seed(1)   # This works for c(10)
model = neuralnet((y=="1") ~ x1+x2, data=d.f, hidden=c(10),
  linear.output=FALSE)
g.x1 = seq(x1min-2, x1max+2, by=0.1); g.x2 = seq(x2min-2, x2max+2, by=0.1)
d.grid = expand.grid(x1=g.x1, x2=g.x2)
pred = predict(model, newdata=d.grid)
```

The contour plot is complex and the ROC indicates the ANN is a good enough model.



According to `https://github.com/udlbook/udlbook`, best results are created by deep networks with many layers, 50–1000 layers for most applications. The training hyperparameters include the choice of learning algorithms, learning rate and momentum.

3.5 Neural Network Architectures

References:

- https://h2o.ai/wiki/neural-network-architectures/#:~:text=What%20Is%20Neural%20Network%20Architecture,power%20of%20a%20human%20brain.
- Makosso et al. [2024] (https://ijeeas.utem.edu.my/ijeeas/article/view/6213), Mienye and Swart [2024].
- Samaya Madhavan, M. Tim Jones, *Deep learning architectures: The rise of artificial intelligence*, 24 April 2024 (https://developer.ibm.com/articles/cc-machine-learning-deep-le
- https://www.topbots.com/a-brief-history-of-neural-network-architectures/
- https://www.coursera.org/articles/neural-network-architecture

All the figures in this section are taken from Wikipedia unless otherwise stated.

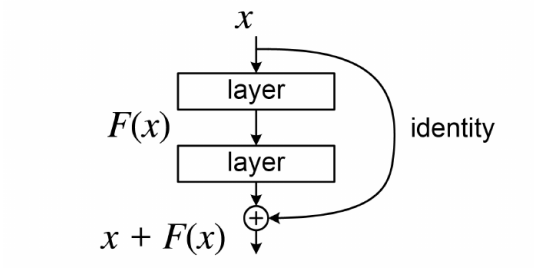
According to the above references, the neural network based architectures are roughly classified to

- Supervised Learning:
  - **Standard Artificial Neural Network**
    - Feed-Forward Network / Multi-Layer Perceptron (MLP): Section 3.4
    - Residual Networks (ResNet): Section 3.5.1
  - **Convolutional Neural Network (CNN)**: Section 3.5.2 (mentioning AlexNet, Visual geometry group (VGG), YOLO, ...)
  - **Recurrent Network Network (RNN)**: Section 3.5.3 (mentioning Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Echo state network (ESN), ...)
  - **Transformer**: Section 3.5.4
- Unsupervised Learning:
  - **Generative Adversarial Networks (GAN)**: Section 4.6.2
  - **Encoder-Decoder Architectures**: Section 4.6.3 (mentioning Autoencoders, Variational Autoencoders)
  - **Self Organising Maps (SOM)**: Section 8.3

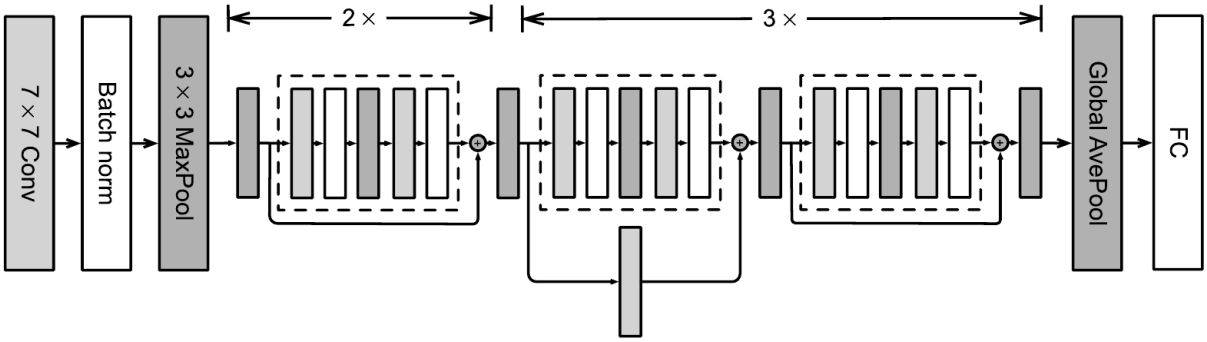
Remark 3.5.1. You can visit http://www.asimovinstitute.org/neural-network-zoo/ for alternative visualisation of neural network architectures.

3.5.1 Residual Networks (ResNet)

https://en.wikipedia.org/wiki/Residual_neural_network (also referred to as a residual network or ResNet) is a seminal deep learning model in which the weight layers learn residual functions with reference to the layer inputs. It was developed in 2015 for image recognition and won that year's ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

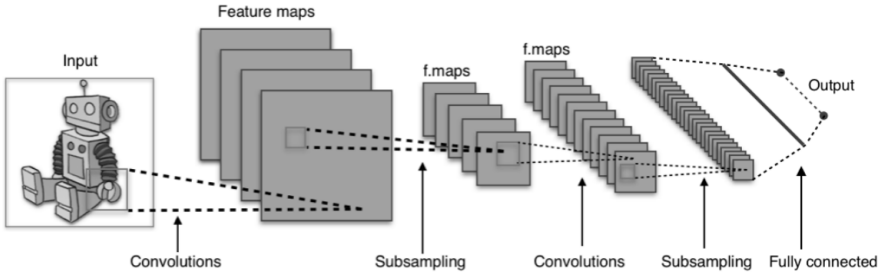


Note that when ResNet are combined with CNN models, an image classification model can be obtained.



3.5.2 Convolutional Neural Network (CNN)

`https://en.wikipedia.org/wiki/Convolutional_neural_network` is a regularised type of feed-forward ANN that learns feature engineering via kernel optimisation on data with grid structures.



They are mostly used in image classification and object detection but can be applied to audio data (e.g. `https://github.com/jeffprosise/Deep-Learning/blob/master/Audio%20Classification%20(CNN).ipynb`) and CT data (e.g. `https://www.nature.com/articles/s41598-020-79336-5`).

Famous CNN-based architectures for **image classification** are summarised in the figure below.

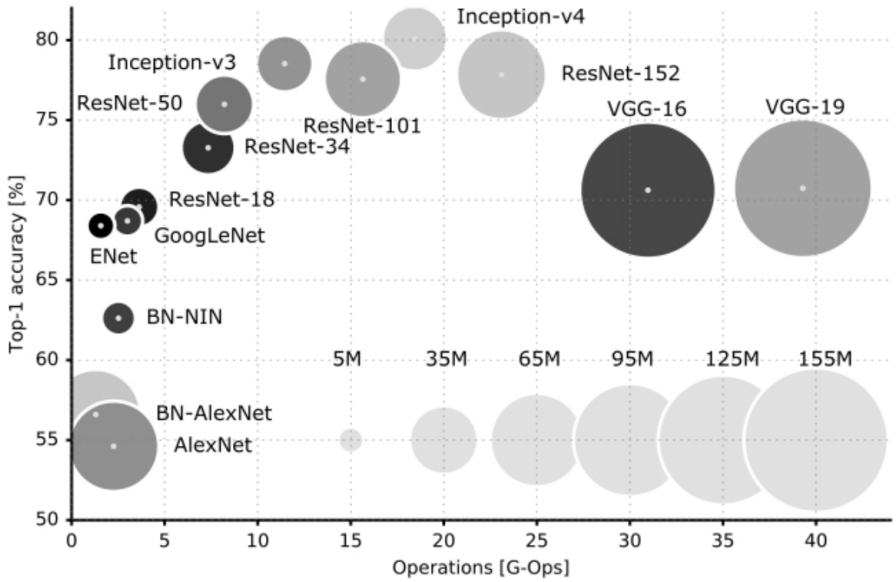
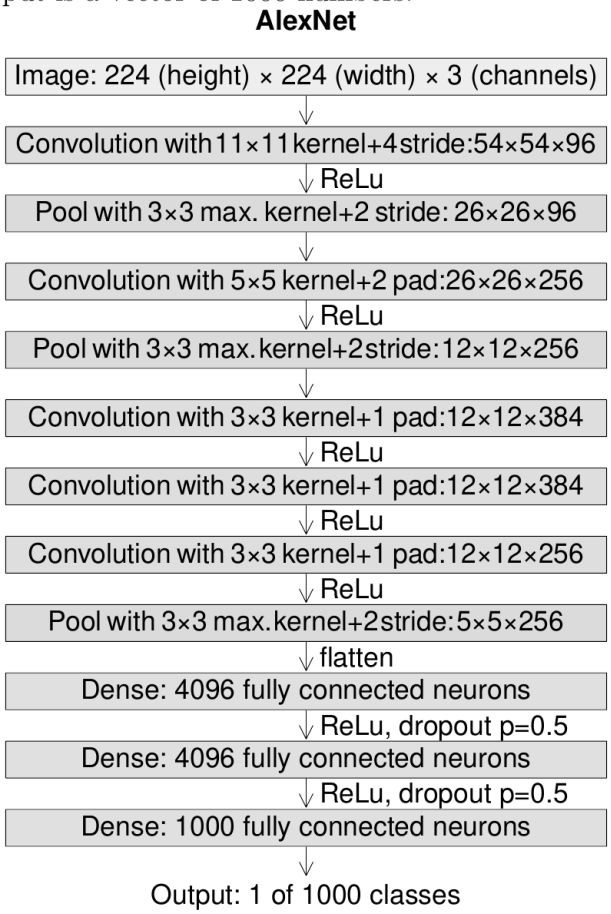


Figure CNN.
Source:
`https://www.topbots.com/a-brief-history-of-neural-network-architectures/`

Famous CNN-based architectures for object detection and image segmentation are YOLOv1 to YOLOv10.

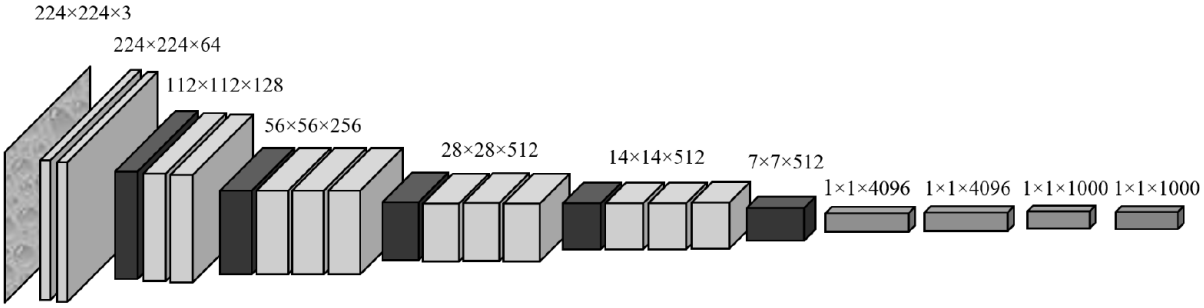
AlexNet

`https://en.wikipedia.org/wiki/AlexNet` was introduced in 2012 with the following 8-layer CNN + 3-layer ANN architecture for computer vision where the input is an image and the output is a vector of 1000 numbers.



Visual geometry group (VGG)

VGG (2014) is a CNN model up to 19 layers, $\geq 6 \times 10^6$ parameters that uses multiple smaller kernel-sized filters that provides more accuracy when classifying images with the following architecture.



GoogLeNet, Inception and Xception

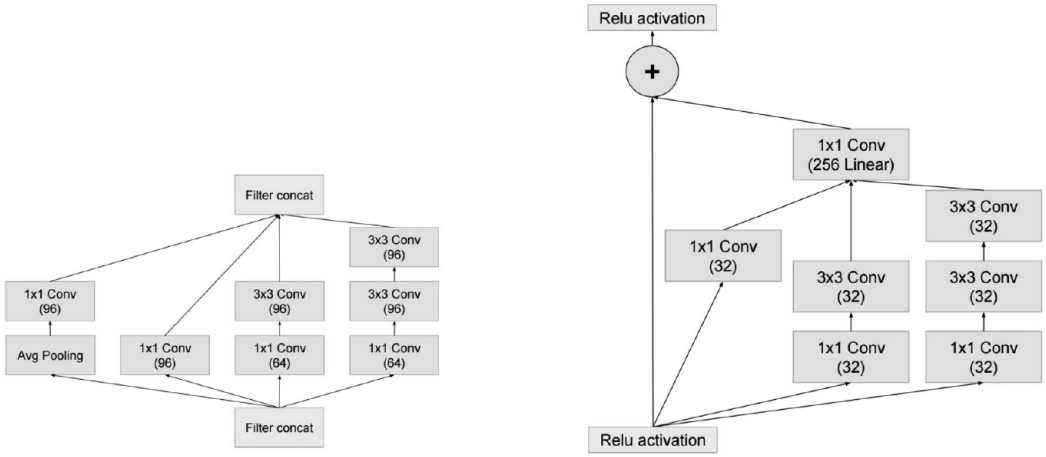
Google's Christian Szegedy began a quest aimed at reducing the computational burden of deep neural networks, and devised the GoogLeNet, the first Inception architecture, a CNN model up

to 22 layers. It is comparatively smaller and faster than VGG and more accurate in detailing than AlexNet (Ref: Figure CNN).

In December 2015, Inception v3 modules and the corresponding architecture were released with a list of original ideas:

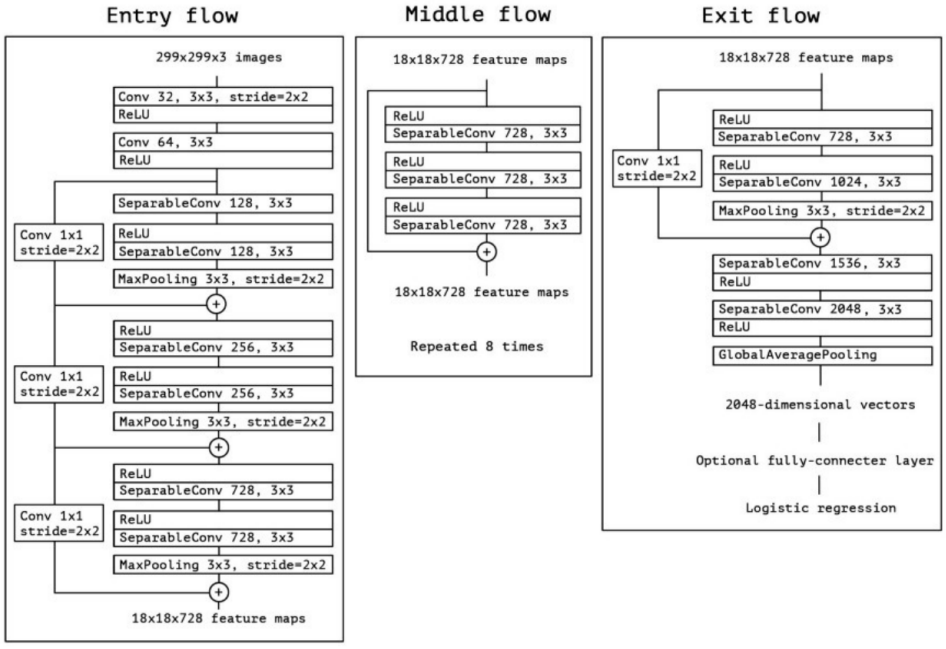
- maximise information flow into the network, by carefully constructing networks that balance depth and width. Before each pooling, increase the feature maps.
- when depth is increased, the number of features, or width of the layer is also increased systematically
- use width increase at each layer to increase the combination of features before next layer
- use only 3×3 convolution, when possible, given that filter of 5×5 and 7×7 can be decomposed with multiple 3×3. ...

Inception V4 were later released with the inception module on the left combined with the ResNet module on the right.



Source:
`https://www.topbots.com/a-brief-history-of-neural-network-architectures/`

Xception, short for Extreme Inception, is a Deep Learning model developed by Francois Chollet at Google, continuing the popularity of Inception architecture, and further perfecting it to the following architecture.

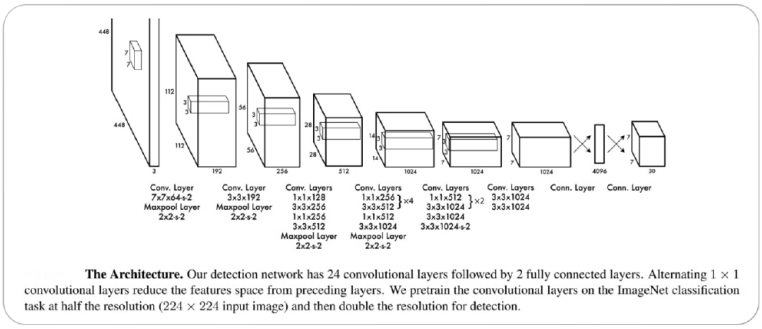


Source:
https://www.topbots.com/a-brief-history-of-neural-network-architectures/

YOLO (You Only Look Once)

YOLO is a one-stage architecture for object detection in comparison to RNN based models (e.g. RCNN, Fast-RCNN, RFCN, Mask RCNN, etc.) which are two-stage architectures.

YOLOv1 has an architecture shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Source: https://arxiv.org/abs/1506.02640

3.5.3 Recurrent Neural Network (RNN)

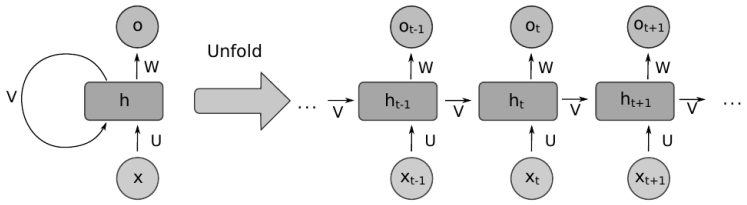
https://en.wikipedia.org/wiki/Recurrent_neural_network is a **bi-directional** ANN, i.e. it allows the output from some nodes to affect subsequent input to the same nodes such as

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Its ability to use internal state (memory) to process arbitrary sequences of inputs makes it applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. It is difficult to parallelised due to the sequential processing of data in the model.

Fully RNN (FRNN)

FRNN connects the outputs of all neurons to the inputs of all neurons.

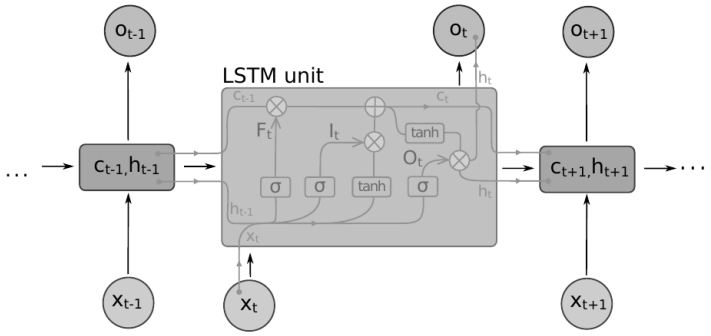


Long Short-Term Memory (LSTM)

https://en.wikipedia.org/wiki/Long_short-term_memory is first introduced in 1997 which deals with the vanishing gradient problem present in traditional RNNs (e.g. FRNN) [Mienye and Swart, 2024]:

$$\begin{aligned}
F_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
I_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
O_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
c_t &= F_t * c_{t-1} + I_t * \tanh(W_c[h_{t-1}, x_t] + b_c) \\
h_t &= O_t * \tanh(c_t)
\end{aligned}$$

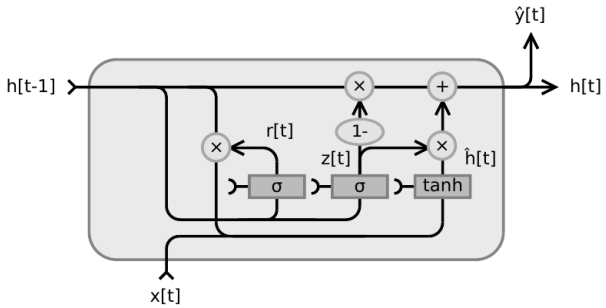
where F_t, I_t and O_t are the forget, input and output gates respectively which control the cell state updates.



Gated Recurrent Unit (GRU)

`https://en.wikipedia.org/wiki/Gated_recurrent_unit` is a gating mechanism in RNN, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a LSTM with a gating mechanism to input or forget certain features, but lacks a context vector or output gate, resulting in fewer parameters than LSTM [Mienye and Swart, 2024].

$$
\begin{aligned}
z_t &= \sigma(W_z[h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r[h_{t-1}, x_t] + b_r) \\
\widetilde{h}_t &= \tanh(W_h[r_t * h_{t-1}, x_t] + b_h) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t
\end{aligned}
$$

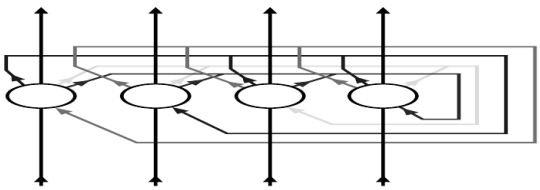


Neural Turing Machine (NTM)

`https://en.wikipedia.org/wiki/Neural_Turing_machine`) has an ANN controller coupled to external memory resources, which it interacts with through attentional mechanisms. The memory interactions are differentiable end-to-end, making it possible to optimise them using gradient descent. An NTM with a LSTM network controller can infer simple algorithms such as copying, sorting, and associative recall from examples alone.

Hopfield Network

`https://en.wikipedia.org/wiki/Hopfield_network` is an RNN in which all connections across layers are equally sized. It requires stationary inputs and is thus not a general RNN, as it does not process sequences of patterns. However, it guarantees that it will converge. If the connections are trained using Hebbian learning, then the Hopfield network can perform as robust content-addressable ("associative") memory, resistant to connection alteration.

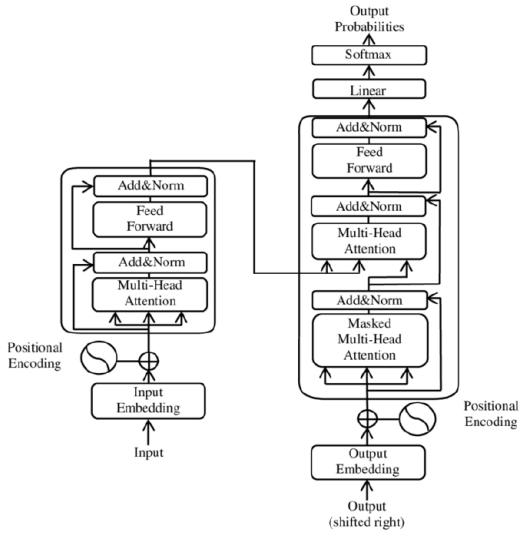


`https://en.wikipedia.org/wiki/Boltzmann_machine` can be thought of as a noisy Hopfield network. However, Boltzmann machines with unconstrained connectivity have not been proven useful for practical problems in machine learning or inference.

3.5.4 Transformer

`https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)` is an ANN based on the multi-head attention mechanism (no recurrent units) and requires less training time than RNN. It has had great success in natural language processing (NLP), e.g. the tasks of machine translation and time series prediction [Mienye and Swart, 2024].

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



As pointed out by Mienye and Swart [2024], the transformer architecture has revolutionised the sequence processing task making the processing of data with mixed texts, images, audios and videos possible using the variations of transformer architectures:

- Large language models (LLMs):
  - `https://grok.com/` (X/Twitter, closed source)
  - `https://chatgpt.com/` (OpenAI, GPT-3 and onwards are closed source while GPT-1 (512 tokens, 0.117B params, 12 decoders) and GPT-2 (1024 tokens, 1.5B params, 48 decoders) are open source). Note that GPT stands for Generative Pre-trained Transformers
  - `https://copilot.microsoft.com/` (closed source)
  - `https://gemini.google.com/app` (Google, closed source),
  - `https://chat.qwen.ai/` (Alibaba, some models are open source)
  - `https://llama.online/` (Meta/Facebook, some models are open source)

  - Mistral AI (https://mistral.ai/technology/)
  - Anthropic Claude
  - `https://www.dola.com/chat/` (ByteDance, closed source)
  - `https://chat.z.ai/`, `https://glm45.org/` (Zhipu AI, `https://en.wikipedia.org/wiki/Z.ai`, closed source)
  - `https://kimi-ai.chat/` (closed source) (`https://en.wikipedia.org/wiki/Moonshot_AI`)
  - `https://deep-seek.com/` (`https://www.deepseek.com/`, open source)
  - `https://yuanbao.tencent.com/chat/` (Tencent, using DeepSeek, closed source)
  - `https://www.genspark.ai/` (closed source)

- Speech Generation AI:
  - Alibaba Qwen3-TTS (`https://github.com/QwenLM/Qwen3-TTS`)

- Image Generation AI:
  - `https://stablediffusion.com/` (open source, `https://huggingface.co/stabilityai`)
  - `https://invoke.ai/` (open source, `https://github.com/invoke-ai/InvokeAI`)
  - `https://openjourney.net/` (open source, `https://huggingface.co/prompthero/openjourney`)
  - `https://localai.io/` (open source? `https://github.com/mudler/LocalAI`)
  - `https://fooocusai.ai/` (`https://github.com/lllyasviel/Fooocus`)

- Audio and Video Generation AI:
  - `https://stablediffusionweb.com/`
  - HunyuanVideo (Tencent, requires login with WeChat)

3.5.5 Applications

According to `https://www.coursera.org/articles/neural-network-architecture`, we have been studying and implementing ANN since the 1940s, advancements in deep learning have guided us to work with ANN in new and advanced ways. Today, researchers and scientists can use ANN for real-world **applications** in various fields, including the automotive industry, finance, agriculture, insurance, health care, and utilities.

- **Vibe Coding**: A chatbot-based approach to creating software where the developer describes a project or task to a LLM, which generates source code based on the prompt.

- **Transport / Automotive**: Self-driving cars use ANNs to make decisions based on the data they receive from their surroundings. ANNs can also optimise vehicle parts and functions or estimate how many vehicles you need to make to meet demand.

- **Health care**: In a health care setting, doctors, health care administrators, and researchers use ANNs to make informed decisions about patient care, organisational decisions, and developing new medications.

- **Agriculture**: Unmanned aircraft can be used to distribute seeds, fertilisers and monitor the growth of crops, plants, etc.

- **Finance**: ANNs have many uses in the finance industry, from predicting the performance of the stock market or exchange rates between monetary denominations to determining credit scores and default risks.
- **Insurance**: Insurance providers can use ANNs to model how often customers file insurance claims and the size of those claims.
- **Energy Utilities**: Utility companies can use ANNs to forecast energy demand. Other uses include stabilising electrical voltage or modelling oil recovery from residential areas.
- **Medicine Design**: Protein structure prediction is extremely important in modern medicine design since it provides a much better tool (compare to the classical bioinformatics tools) to understand how proteins interact based on their 3D folding structures [Desai et al., 2024].

According to `https://www.nobelprize.org/prizes/lists/all-nobel-prizes/`,

- The Nobel Prize in Physics 2024 is awarded to John J. Hopfield and Geoffrey E. Hinton **for foundational discoveries and inventions (Hopfield network and Boltzmann machine) that enable machine learning with artificial neural networks**
- The Nobel Prize in Chemistry 2024 is awarded to David Baker **for computational protein design** (`https://www.bakerlab.org/`) and Demis Hassabis and John M. Jumper **for protein structure prediction** (`https://deepmind.google/technologies/alphafold/`)

3.5.6 Model Reduction

Modern neural networks are usually extremely large with billions of parameters. To deploy them in portable/wearable/embedded devices, **AI compression** techniques are needed:

- neural network pruning technique: weight pruning, neuron pruning, filter pruning, layer pruning
- quantisation technique
- knowledge distillation technique: offline distillation, online distillation, self-distillation
- low-rank factorisation technique

3.5.7 Dangers

The dangers of modern AI tools:

- **Expensive Power Consumption**: Large language models (LLMs, e.g. ChatGPT), image generative models (e.g. DALL-E), video generative models (e.g. OpenAI SORA) require a lot of electricity to drive the GPU computations.

  You can watch "How much energy AI really needs. And why that's not its main problem" (`https://www.youtube.com/watch?v=0ZraZPFVr-U`) to understand the detail.

  If you understand Chinese, you can watch `https://www.youtube.com/watch?v=QugfrCwayQU`

- **Misleading / Fake News**: LLMs can generate titles or summaries which are misleading since they don't understand the context of the training data. They are companies who put fake news on the Internet and when AI are trained against the news on the Internet, the wide-spread fake news may have a chance to be integrated into the AI causing summaries of fake news to be generated by AI.

- **Fake photos / videos can be easily generated (coined "deepfake" by some experts)**: This has significant influence in the political arena in many countries. (Ref: Noah Giansiracusa, "The Mathematics of Misinformation") E.g. US can create fake news to influence the politics of many countries (e.g. Arab Spring anti-government protest). Some AI experts have demonstrated the possibility of faking videos with Obama saying something which he has not said using his public speech and using AI to mimic his tone and "say something controlled by AI".

- **Privacy Invasion**: Large companies such as Microsoft (e.g. Win 11 Copile), Google (e.g. Gemini and Android One), Adobe (e.g. Acrobat Reader), Kingsoft (e.g. WPS Office) are using AI to mine companies and users' computer / smart-phone information and privacy protection for companies and users is a disaster.