Fundaments of Machine learning for and with engineering applications

Enrico Riccardi[1]

Department of Mathematics and Physics, University of Stavanger (UiS).[1]

Sep 15, 2025

1 Linear Regression

2 Linear Regression

3 Model evaluation

4 Neural Network -NN-

A linear model

Considering a univariate case, we have:

$$q = f(x)$$

which relates the **independent variable** x to the **true dependent variable** q.

Assuming a linear model

$$q = \beta_0 + \beta_1 x$$

where β_i are the arbitrary selected coefficients.

A linear model

Considering a univariate case, we have:

$$q = f(x)$$

which relates the **independent variable** x to the **true dependent variable** q.

Assuming a linear model

$$q = \beta_0 + \beta_1 x$$

where β_i are the arbitrary selected coefficients.

A linear model

Considering a univariate case, we have:

$$q = f(x)$$

which relates the **independent variable** x to the **true dependent variable** q.

Assuming a linear model

$$q = \beta_0 + \beta_1 x$$

where β_i are the arbitrary selected coefficients.

Model set-up

For a given x we do not know the true response q, only the measurementa y_i for experiment i.

We have that:

$$y_i = q_i + \epsilon_i$$

NOTE: do not proceed if you do not fully understand this equation.

which is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Discussion point

Does ϵ_i matter? And why so?

Model set-up

For a given x we do not know the true response q, only the measurementa y_i for experiment i.

We have that:

$$y_i = q_i + \epsilon_i$$

NOTE: do not proceed if you do not fully understand this equation.

which is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Discussion point

Does ϵ_i matter? And why so?

Model set-up

For a given x we do not know the true response q, only the measurementa y_i for experiment i.

We have that:

$$y_i = q_i + \epsilon_i$$

NOTE: do not proceed if you do not fully understand this equation.

which is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Discussion point

Does ϵ_i matter? And why so?

Model set-up

For a given x we do not know the true response q, only the measurementa y_i for experiment i.

We have that:

$$y_i = q_i + \epsilon_i$$

NOTE: do not proceed if you do not fully understand this equation.

which is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Discussion point

Does ϵ_i matter? And why so?

Estimated model paramters

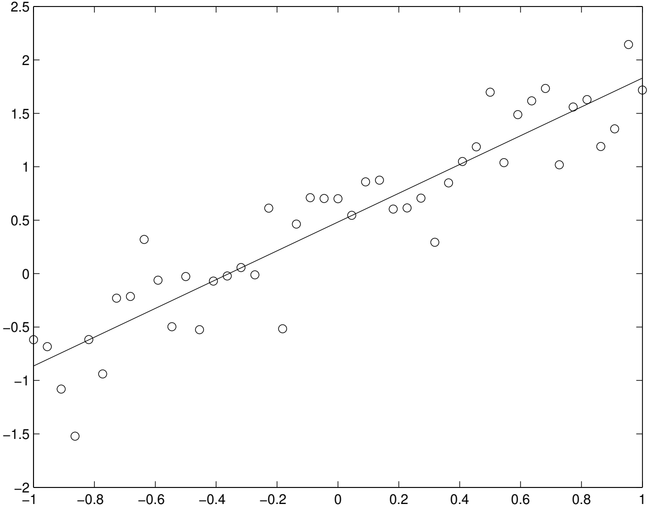
The model parameters β_0, β_1 are unknown, but they can be **estimated**. To distinguish estimates from true model parameters we call them b_0, b_1. These estimates are calculated such that the model

$$\hat{y} = b_0 + b_1 x$$

fits the n different experimental observations as well as possible.

Linear model example

We would like to find the TRUTH (What it it?)



Python Example

```
import numpy as np
import matplotlib.pyplot as plt

def generate_linear_data(n_random_points, noise=16):
    x = np.random.rand(n_random_points) * 10

    # Make 'perfect' data
    true_slope,  true_intercept = 2, 5
    y = true_slope * x + true_intercept

    # Add noise
    y += np.random.randn(n_random_points)*noise

    return x, y, true_slope, true_intercept

# Use the function to generate data
x, y, true_slope, true_intercept = generate_linear_data(
        n_random_points=166,
        noise=3)

# Plot all
plt.plot(x, true_slope*x + true_intercept,
         color='red', label='Truth Line')
plt.scatter(x, y, color='blue', label='Data Points')
plt.show()
```

Estimation of linear regression parameters

For the 1-dimensional problem, we have

$$\hat{y} = b_0 + b_1 x$$

where $\hat{y}$ is the estimated y-value from the approximate model that has been generated from a set of measurements (x_i, y_i). We aim to find the b_i parameters such that the regression line fits the observed data as well as possible.

This means we want to minimise the residuals

$$e_i = y_i - \hat{y}_i$$

Estimation of linear regression parameters

For the 1-dimensional problem, we have

$$\hat{y} = b_0 + b_1 x$$

where $\hat{y}$ is the estimated y-value from the approximate model that has been generated from a set of measurements (x_i, y_i). We aim to find the b_i parameters such that the regression line fits the observed data as well as possible.

This means we want to minimise the residuals

$$e_i = y_i - \hat{y}_i$$

Estimation of linear regression parameters

For the 1-dimensional problem, we have

$$\hat{y} = b_0 + b_1 x$$

where $\hat{y}$ is the estimated y-value from the approximate model that has been generated from a set of measurements (x_i, y_i). We aim to find the b_i parameters such that the regression line fits the observed data as well as possible.

This means we want to minimise the residuals

$$e_i = y_i - \hat{y}_i$$

Estimation of linear regression parameters

For the 1-dimensional problem, we have

$$\hat{y} = b_0 + b_1 x$$

where $\hat{y}$ is the estimated y-value from the approximate model that has been generated from a set of measurements (x_i, y_i). We aim to find the b_i parameters such that the regression line fits the observed data as well as possible.

This means we want to minimise the residuals

$$e_i = y_i - \hat{y}_i$$

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Estimation of linear regression parameters

- Cannot sum e_i values since they might be positive and negative and thus cancel
- Could use e.g. $\sum_{i=1}^{n} |e_i|$, but is mathematically more difficult to handle
- Residual"smallness" measured by $\sum_{i=1}^{n} e_i^2$.

Thus, we find the linear regression coefficients by *minimising*

$$R = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

How?

Regression!

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Regression

To minimize the sum of the square residuals, we can try to solve the following equations:

$$\frac{\partial R}{\partial b_0} = 0$$
$$\frac{\partial R}{\partial b_1} = 0$$

where:

$$R = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 =$$
$$\sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 = \sum_{i=1}^{n} u_i^2$$

Regression

To minimize the sum of the square residuals, we can try to solve the following equations:

$$\frac{\partial R}{\partial b_0} = 0$$
$$\frac{\partial R}{\partial b_1} = 0$$

where:

$$R = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 =$$
$$\sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 = \sum_{i=1}^{n} u_i^2$$

Regression

To minimize the sum of the square residuals, we can try to solve the following equations:

$$\begin{aligned}\frac{\partial R}{\partial b_0} &= 0\\ \frac{\partial R}{\partial b_1} &= 0\end{aligned}$$

where:

$$R = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad =$$

$$\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 = \sum_{i=1}^{n} u_i^2$$

Regression

Skipping the math (but you are more than welcome to try), here are the results:

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Straightforward derivation becomes very cumbersome for multiple variables. Thus, a different approach must be used. Yet, it is important to understand that there is an analytical solution (even if not all the time).

Regression

Skipping the math (but you are more than welcome to try), here are the results:

$$
\begin{aligned}
b_0 &= \bar{y} - b_1 \bar{x} \\
b_1 &= \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}
\end{aligned}
$$

Straightforward derivation becomes very cumbersome for multiple variables. Thus, a different approach must be used. Yet, it is important to understand that there is an analytical solution (even if not all the time).

Regression

Skipping the math (but you are more than welcome to try), here are the results:

$$
\begin{aligned}
b_0 &= \bar{y} - b_1 \bar{x} \\
b_1 &= \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}
\end{aligned}
$$

Straightforward derivation becomes very cumbersome for multiple variables. Thus, a different approach must be used. Yet, it is important to understand that there is an analytical solution (even if not all the time).

Linear model(s)

Does a linear model mean only straight lines (or hyperplanes in general)?

That answer to this is *no*. different. In general for a model f to be defined linear, it has to be linear with respect to the unknown parameters $\beta_0, \cdots, \beta_n$. The general linear model is

$$q = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots + \beta_n f_n(x_n)$$

where $f_i(x_i)$ may be non-linear functions. It is the **form** of the equation which makes it linear, i.e. that $f_i(x_i)$ does not depend on the parameters β_i.

Linear model(s)

Does a linear model mean only straight lines (or hyperplanes in general)?

That answer to this is *no*. different. In general for a model f to be defined linear, it has to be linear with respect to the unknown parameters $\beta_0, \cdots, \beta_n$. The general linear model is

$$q = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots + \beta_n f_n(x_n)$$

where $f_i(x_i)$ may be non-linear functions. It is the **form** of the equation which makes it linear, i.e. that $f_i(x_i)$ does not depend on the parameters β_i.

Linear model(s)

Does a linear model mean only straight lines (or hyperplanes in general)?

That answer to this is *no*. different. In general for a model f to be defined linear, it has to be linear with respect to the unknown parameters $\beta_0, \cdots, \beta_n$. The general linear model is

$$q = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots + \beta_n f_n(x_n)$$

where $f_i(x_i)$ may be non-linear functions. It is the **form** of the equation which makes it linear, i.e. that $f_i(x_i)$ does not depend on the parameters β_i.

Linear model(s)

Consider the following model example - is it linear?

$$q = \beta_0 + \beta_1 x_1^2 + \beta_2 x_2^{-1} + \beta_3 \log x_3$$

The answer is *yes* because by simple substitution it is possible to convert this formula into a linear form.

With $h_1 = x_1^2$, $h_2 = x_2^{-1}$ and $h_3 = \log x_3$, then we can formulate the new model:

$$q = \beta_0 + \beta_1 h_1 + \beta_2 h_2 + \beta_3 h_3$$

which is in the standard linear form.

Linear model(s)

Consider the following model example - is it linear?

$$q = \beta_0 + \beta_1 x_1^2 + \beta_2 x_2^{-1} + \beta_3 \log x_3$$

The answer is *yes* because by simple substitution it is possible to convert this formula into a linear form.

With $h_1 = x_1^2$, $h_2 = x_2^{-1}$ and $h_3 = \log x_3$, then we can formulate the new model:

$$q = \beta_0 + \beta_1 h_1 + \beta_2 h_2 + \beta_3 h_3$$

which is in the standard linear form.

Linear model(s)

Consider the following model example - is it linear?

$$q = \beta_0 + \beta_1 x_1^2 + \beta_2 x_2^{-1} + \beta_3 \log x_3$$

The answer is *yes* because by simple substitution it is possible to convert this formula into a linear form.

With $h_1 = x_1^2$, $h_2 = x_2^{-1}$ and $h_3 = \log x_3$, then we can formulate the new model:

$$q = \beta_0 + \beta_1 h_1 + \beta_2 h_2 + \beta_3 h_3$$

which is in the standard linear form.

Linear model(s)

Consider the following model example - is it linear?

$$q = \beta_0 + \beta_1 x_1^2 + \beta_2 x_2^{-1} + \beta_3 \log x_3$$

The answer is *yes* because by simple substitution it is possible to convert this formula into a linear form.

With $h_1 = x_1^2$, $h_2 = x_2^{-1}$ and $h_3 = \log x_3$, then we can formulate the new model:

$$q = \beta_0 + \beta_1 h_1 + \beta_2 h_2 + \beta_3 h_3$$

which is in the standard linear form.

Curvilinear models

A special class of linear models which we will investigate later are those which are expressed in terms of *polynomials* (here in only 1D):

$$q = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m = \sum_{i=0}^{n} \beta_i x^i$$

For n-D case there are a wide range of interaction terms and combinations, that can still be converted to standard linear form.

Such models are sometimes referred to as **curvilinear** instead of non-linear

Curvilinear models

A special class of linear models which we will investigate later are those which are expressed in terms of *polynomials* (here in only 1D):

$$q = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m = \sum_{i=0}^{n} \beta_i x^i$$

For n-D case there are a wide range of interaction terms and combinations, that can still be converted to standard linear form.

Such models are sometimes referred to as **curvilinear** instead of non-linear

Curvilinear models

A special class of linear models which we will investigate later are those which are expressed in terms of *polynomials* (here in only 1D):

$$q = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m = \sum_{i=0}^{n} \beta_i x^i$$

For n-D case there are a wide range of interaction terms and combinations, that can still be converted to standard linear form.

Such models are sometimes referred to as **curvilinear** instead of non-linear

Nonlinear models

But there are many other models that cannot be substituted to such a form. For instance:

$$q = \beta_0 + \log(x - \beta_1)$$

No substitution can transform this equation to the linear form.

That is the case for all the model that:

$$q \sim f(x, \beta)$$

Another example is:

$$f(x, \beta) = \frac{x\beta}{x + \beta}$$

Nonlinear models

But there are many other models that cannot be substituted to such a form. For instance:

$$q = \beta_0 + \log(x - \beta_1)$$

No substitution can transform this equation to the linear form.

That is the case for all the model that:

$$q \sim f(x, \beta)$$

Another example is:

$$f(x, \beta) = \frac{x\beta}{x + \beta}$$

Nonlinear models

But there are many other models that cannot be substituted to such a form. For instance:

$$q = \beta_0 + \log(x - \beta_1)$$

No substitution can transform this equation to the linear form.

That is the case for all the model that:

$$q \sim f(x, \beta)$$

Another example is:

$$f(x, \beta) = \frac{x\beta}{x + \beta}$$

Nonlinear models

But there are many other models that cannot be substituted to such a form. For instance:

$$q = \beta_0 + \log(x - \beta_1)$$

No substitution can transform this equation to the linear form.

That is the case for all the model that:

$$q \sim f(x, \beta)$$

Another example is:

$$f(x, \beta) = \frac{x\beta}{x + \beta}$$

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Let's recaps

Before to continue, let's make sure to have all the main elements clear:

1. Splitting the data in dependent and independent variables
2. Assumption of a linear model between them
3. Recognise the difference between the truth and the estimation
4. Aiming to *minimize* the residuals

Discussion

What happen when the sum of residual is 0 ?

What happens where the data is heavily correlated?

Regression

To minimize the sum of the square residuals, we can try to solve the following equations:

$$\begin{aligned}\frac{\partial R}{\partial b_0} &= 0\\ \frac{\partial R}{\partial b_1} &= 0\end{aligned}$$

where:

$$R = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad =$$

$$\sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2 = \sum_{i=1}^{n} u_i^2$$

Regression

Skipping the math (but you are more than welcome to try), here are the results:

$$\begin{aligned} b_0 &= \bar{y} - b_1\bar{x} \\ b_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \end{aligned}$$

Straightforward derivation becomes very cumbersome for multiple variables. Thus, a different approach must be used. Yet, it is important to understand that there is an analytical solution (even if not all the time).

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Many variable equation

The general equation would look like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n = \sum_{j=0}^{m} x_{ij} b_j$$

where we will have to solve all the $n+1$ equations (called the **normal equations**) of the form:

$$\frac{\partial R}{\partial b_j} = 0 \quad \forall j \in [0, n]$$

Is there a way for us to simplify this?

We can use vector and matrix algebra.

Regression via Matrix operation

Remember that

$$R = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

then we can define the vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$

thus

$$\mathbf{e}^T = [(y_1 - \hat{y}_1)\ (y_2 - \hat{y}_2) \cdots\ (y_N - \hat{y}_N)]$$

and can then write

$$R = \mathbf{e}^T \mathbf{e}$$

Regression via Matrix operation

Remember that

$$R = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

then we can define the vector **e**:

$$\mathrm{e} = \mathrm{y} - \hat{\mathrm{y}}$$

thus

$$\mathrm{e}^T = [(y_1 - \hat{y}_1)\ (y_2 - \hat{y}_2) \cdots\ (y_N - \hat{y}_N)]$$

and can then write

$$R = \mathrm{e}^T \mathrm{e}$$

Regression via Matrix operation

Remember that

$$R = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

then we can define the vector **e**:

$$\mathrm{e} = \mathrm{y} - \hat{\mathrm{y}}$$

thus

$$\mathrm{e}^T = [(y_1 - \hat{y}_1)\ (y_2 - \hat{y}_2) \cdots\ (y_N - \hat{y}_N)]$$

and can then write

$$R = \mathrm{e}^T \mathrm{e}$$

Regression via Matrix operation

Remember that

$$R = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

then we can define the vector **e**:

$$\mathrm{e} = \mathrm{y} - \hat{\mathrm{y}}$$

thus

$$\mathrm{e}^T = [(y_1 - \hat{y}_1)\ (y_2 - \hat{y}_2)\ \cdots\ (y_N - \hat{y}_N)]$$

and can then write

$$R = \mathrm{e}^T \mathrm{e}$$

From the following equation:

$$\hat{y}_i = b_0 + \sum_{j=1}^{m} x_{ij} b_j = \sum_{j=0}^{m} x_{ij} b_j$$

where $x_{i0} = 1$
we make the matrix equation:

$$\hat{\mathrm{y}} = \mathrm{X}\mathrm{b}$$

where the first column in **X** consists of ones only.

Residual

$$\begin{aligned}
R &= e^T e \\
&= (y - \hat{y})^T (y - \hat{y}) \\
&= (y - Xb)^T (y - Xb) \\
&= (y^T - b^T X^T)(y - Xb) \\
&= y^T y - y^T X b - b^T X^T y \\
&+ b^T X^T X b
\end{aligned}$$

All the parts of this equation are scalar values. This means e.g. that

$$y^T X b = b^T X^T y$$

This gives

$$R = y^T y - 2 y^T X b + b^T X^T X b$$

Residual

But how can we now compute $\frac{\partial R}{\partial \boldsymbol{b_j}}$ more efficiently in matrix form?

Vector differentiation ! Let

$$y = \mathrm{a}^T \mathrm{x} = a_1 x_1 + \cdots + a_n x_n$$

If

$$\frac{\partial y}{\partial \mathrm{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = \mathrm{a}$$

and $y = \mathrm{x}^T \mathrm{a}$, then:

$$\frac{\partial y}{\partial \mathrm{x}} = \mathrm{a}$$

General solution

In general, when $y = \mathrm{x}^T \mathrm{A}\mathrm{x}$, then

$$\frac{\partial y}{\partial \mathrm{x}} = 2\mathrm{A}\mathrm{x}$$

if **A** is symmetric
(check *Matrix calculus* for more properties)

We can use this to compute

$$\frac{\partial R}{\partial \mathrm{b}}$$

General solution

We have from above:

$$\begin{aligned} R &= \mathrm{y}^T\mathrm{y} - 2\mathrm{y}^T\mathrm{X}\mathrm{b} \\ &+ \mathrm{b}^T\mathrm{X}^T\mathrm{X}\mathrm{b} \end{aligned}$$

Vector differentiation gives

$$\frac{\partial R}{\partial \mathrm{b}} = 0 - 2\mathrm{X}^T\mathrm{y} + 2\mathrm{X}^T\mathrm{X}\mathrm{b} = 0$$

Solving this for b we get:

$$\begin{aligned} \mathrm{X}^T\mathrm{X}\mathrm{b} &= \mathrm{X}^T\mathrm{y} \\ (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{X}\mathrm{b} &= (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} \\ \mathrm{b} &= (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y} \end{aligned}$$

Multiple linear regression

Previous equation make the solution of MLR rather obvious!

When we have a matrix of y-variables $\mathbf{Y}$:

$$B = (X^T X)^{-1} X^T Y$$

in the equation:

$$Y = XB.$$

These equations give us the **multiple linear regression** (MLR) solution.

Multiple linear regression

Previous equation make the solution of MLR rather obvious!

When we have a matrix of y-variables **Y**:

$$B = (X^T X)^{-1} X^T Y$$

in the equation:

$$Y = XB.$$

These equations give us the **multiple linear regression** (MLR) solution.

Model evaluation

We can measure the accuracy of a prediction only when we have a 'truth' to compare with.

Model outcome

The accuracy of a prediction is limited by the available data set (not an universal quantity).

Given a dataset, we split the data in a train and a test set.

One has to be extremely careful to not introduce a **bias** in each of them when splitting. (or to introduce a bias properly, when for example, working with time series.

Model evaluation

We can measure the accuracy of a prediction only when we have a 'truth' to compare with.

Model outcome

The accuracy of a prediction is limited by the available data set (not an universal quantity).

Given a dataset, we split the data in a train and a test set.

One has to be extremely careful to not introduce a **bias** in each of them when splitting. (or to introduce a bias properly, when for example, working with time series.

Train

A train dataset is a subset of the original data. Generally it is circa a 70 % (common practice, not a rule) of the *INSTANCES*.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances; the selection of the train dataset has to be randomised.

The opposite consideration shall be done for time series, where each data point is correlated to the previous.

Train

A train dataset is a subset of the original data. Generally it is circa a 70 % (common practice, not a rule) of the *INSTANCES*.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances; the selection of the train dataset has to be randomised.

The opposite consideration shall be done for time series, where each data point is correlated to the previous.

Validation

A validation dataset is a subset of the original data. Generally it is circa a 15 % (common practice, not a rule) of the *INSTANCES*.

It is aimed to provide a unbiased estimate of the model prediction error. This information is then used to *tune* the model set up (indirectly). I.e. it is not independent from the test phase.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances; the selection of the validation dataset has to be randomised.

The opposite consideration shall be done for time series, where each data point is correlated to the previous.

Validation

A validation dataset is a subset of the original data. Generally it is circa a 15 % (common practice, not a rule) of the *INSTANCES*.

It is aimed to provide a unbiased estimate of the model prediction error. This information is then used to *tune* the model set up (indirectly). I.e. it is not independent from the test phase.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances; the selection of the validation dataset has to be randomised.

The opposite consideration shall be done for time series, where each data point is correlated to the previous.

Test

A test dataset is a subset of the original data. Generally it is circa a 70 % (common practice, not a rule) of the *INSTANCES*.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances, the selection of the alerttest dataset has to be randomised.

Test

A test dataset is a subset of the original data. Generally it is circa a 70 % (common practice, not a rule) of the *INSTANCES*.

For each selected instance, all the features are kept, as their labels.

Random measurements

Some models are able to also consider the sequence of instances, the selection of the alerttest dataset has to be randomised.

Model evaluation

Train-Validation-Test

1. Data is split into train, validation and test dataset
2. A model is trained on the train dataset
3. The model (hyper)parameters are improved on the validation dataset
4. The model performances (e.g. accuracy) are tested on the test dataset

Model evaluation

Train-Validation-Test

1. Data is split into train, validation and test dataset
2. A model is trained on the train dataset
3. The model (hyper)parameters are improved on the validation dataset
4. The model performances (e.g. accuracy) are tested on the test dataset

Model evaluation

Train-Validation-Test

1. Data is split into train, validation and test dataset
2. A model is trained on the train dataset
3. The model (hyper)parameters are improved on the validation dataset
4. The model performances (e.g. accuracy) are tested on the test dataset

Model evaluation

Train-Validation-Test

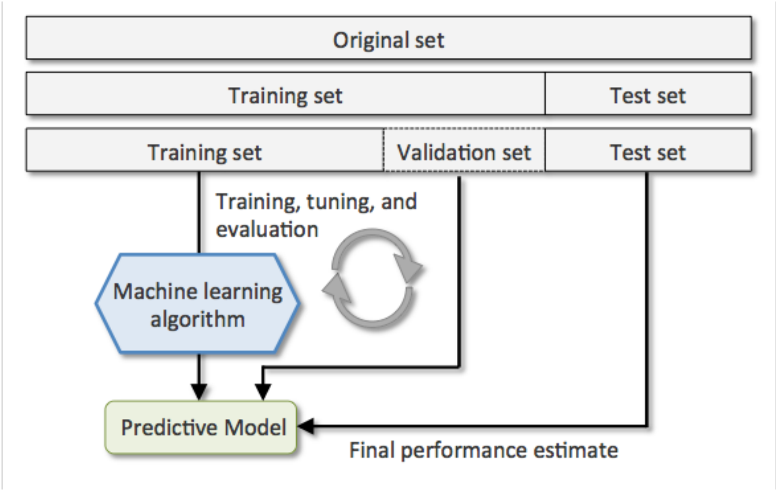
1. Data is split into train, validation and test dataset
2. A model is trained on the train dataset
3. The model (hyper)parameters are improved on the validation dataset
4. The model performances (e.g. accuracy) are tested on the test dataset

Model evaluation

Train-Validation-Test

1. Data is split into train, validation and test dataset
2. A model is trained on the train dataset
3. The model (hyper)parameters are improved on the validation dataset
4. The model performances (e.g. accuracy) are tested on the test dataset

Model evaluation



Cross validation

Cross validation-Test

1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation

Cross validation-Test

1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation

Cross validation-Test

1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation

Cross validation-Test

1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation

Cross validation-Test

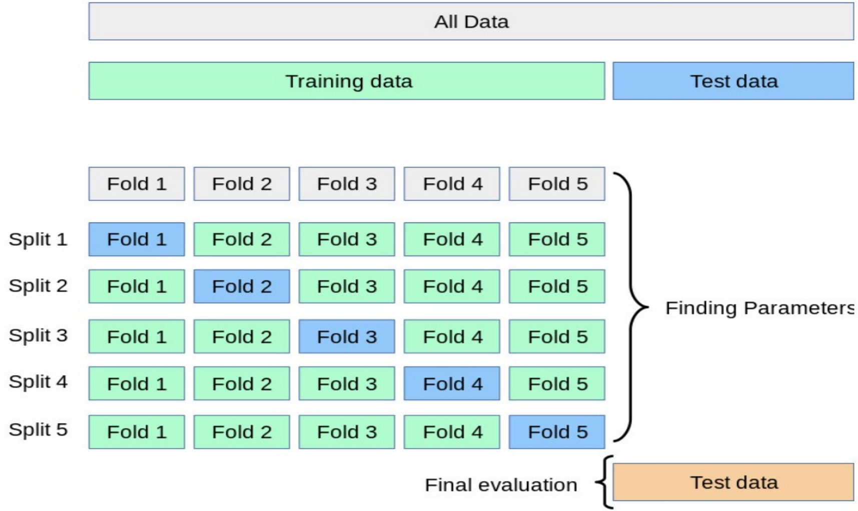
1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation

Cross validation-Test

1. Data is split into train and test.
2. Train dataset is then split into k-folds (different subsets, usually 5 to 10)
3. A model is trained on all k subset but one, sequentially
4. The model is tested on the remaining k data subset and its output averaged
5. The model performances (e.g. accuracy) are tested on the test dataset

Cross validation



Neural Network -NN-

One of the most famous models in Machine Learning is **Neural Network**.

The name comes from how the brain functions: a set of connected neurons that are either off or active.

In its essence,

NN is a large set of (linear) regressions executed both in parallel and in series.

Conventional NN are composed by:

1. Input layer
2. Hidden layers
3. Output layers

which, together, can approximately approximate any function in any dimension.

Neural Network -NN-

One of the most famous models in Machine Learning is **Neural Network**.

The name comes from how the brain functions: a set of connected neurons that are either off or active.

In its essence,

NN is a large set of (linear) regressions executed both in parallel and in series.

Conventional NN are composed by:

1. Input layer
2. Hidden layers
3. Output layers

which, together, can approximately approximate any function in any dimension.

Neural Network -NN-

One of the most famous models in Machine Learning is **Neural Network**.

The name comes from how the brain functions: a set of connected neurons that are either off or active.

In its essence,

NN is a large set of (linear) regressions executed both in parallel and in series.

Conventional NN are composed by:

1. Input layer
2. Hidden layers
3. Output layers

which, together, can approximately approximate any function in any dimension.

Neural Network -NN-

One of the most famous models in Machine Learning is **Neural Network**.

The name comes from how the brain functions: a set of connected neurons that are either off or active.

In its essence,

NN is a large set of (linear) regressions executed both in parallel and in series.

Conventional NN are composed by:

1. Input layer
2. Hidden layers
3. Output layers

which, together, can approximately approximate any function in any dimension.

Neural Network -NN-

One of the most famous models in Machine Learning is **Neural Network**.

The name comes from how the brain functions: a set of connected neurons that are either off or active.

In its essence,

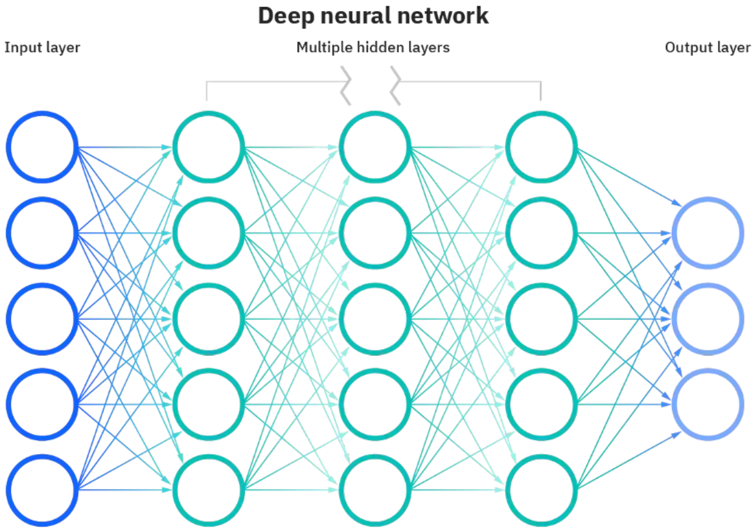
NN is a large set of (linear) regressions executed both in parallel and in series.

Conventional NN are composed by:

1. Input layer
2. Hidden layers
3. Output layers

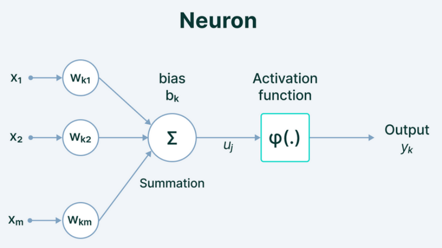
which, together, can approximately approximate any function in any dimension.

deep Neural Network



How do they work?

Each node is called neuron

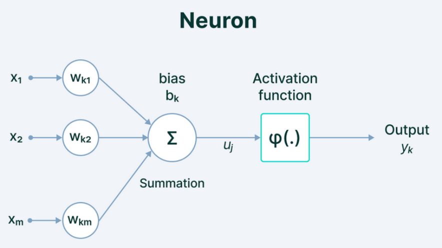


Use many of these, many times, and you have builded a NN!

- It is really just a $Y = f(X)$, where w_i and b_i are the unknowns to solve for.
- As in linear regression, this becomes mainly a number of numerical recipes.
- As the problem's dimensionality is significant, several strategies have been developed.

How do they work?

Each node is called neuron



Use many of these, many times, and you have builded a NN!

- It is really just a $Y = f(X)$, where w_i and b_i are the unknowns to solve for.
- As in linear regression, this becomes mainly a number of numerical recipes.
- As the problem's dimensionality is significant, several strategies have been developed.

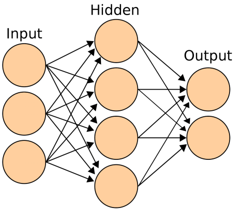
How do they work?

1. We thus need also an activation function.
2. The NN structure
3. Also, we might need to specify the number of epochs.

Using NN is essentially running a simulation campaign: a set of numerical experiments

Please say yes

Have you already heard of experimental design?



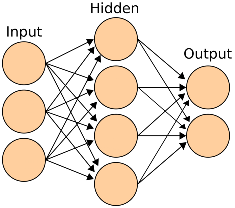
How do they work?

1. We thus need also an activation function.
2. The NN structure
3. Also, we might need to specify the number of epochs.

Using NN is essentially running a simulation campaign: a set of numerical experiments

Please say yes

Have you already heard of experimental design?

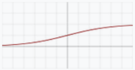


Activation function

Each node applies an activation function on the weighted sum of the previous nodes.

Such functions are called activation functions. There are MANY!

The most popular has been the logistic

	$\sigma(x) \doteq \dfrac{1}{1+e^{-x}}$	$g(x)(1-g(x))$

Now the most common is ReLU (Rectified linear unit)

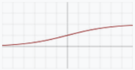
	$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0,x) = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$

Activation function

Each node applies an activation function on the weighted sum of the previous nodes.

Such functions are called activation functions. There are MANY!

The most popular has been the logistic

	$\sigma(x) \doteq \dfrac{1}{1+e^{-x}}$	$g(x)(1-g(x))$

Now the most common is ReLU (Rectified linear unit)

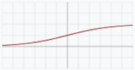
	$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0,x) = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$

Activation function

Each node applies an activation function on the weighted sum of the previous nodes.

Such functions are called activation functions. There are MANY!

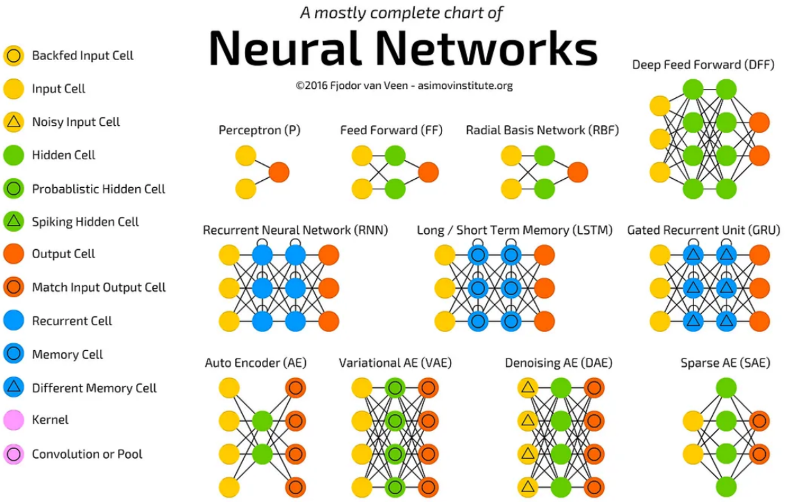
The most popular has been the logistic

	$\sigma(x) \doteq \dfrac{1}{1+e^{-x}}$	$g(x)(1-g(x))$

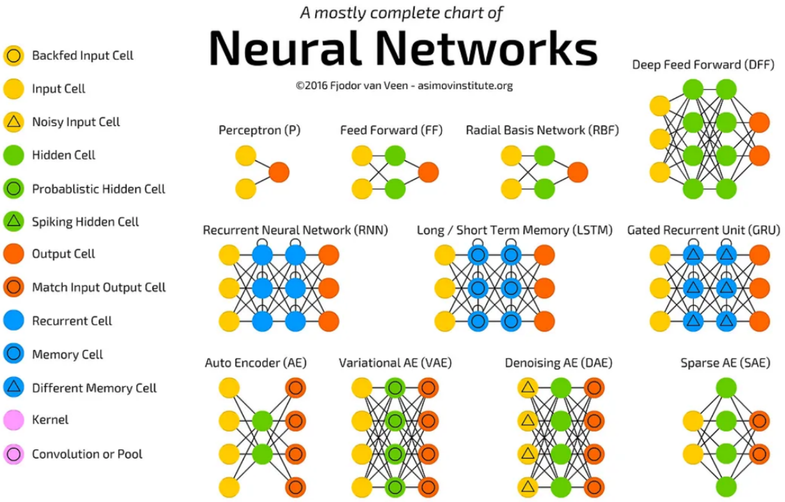
Now the most common is ReLU (Rectified linear unit)

	$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0,x) = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$

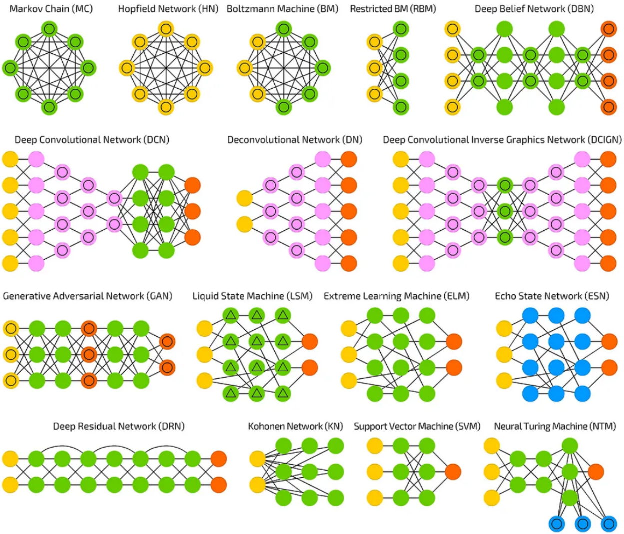
NN Architecture



NN Architecture



NN Architecture



NN Architecture

Neural net suffers from overfitting problems.

Even more parameters

The number of nodes, the architecture, the number of hidden layer etc are NN hyperparameters

Unfortunately, at the current status of knowledge, the best architecture can be found only by trial and error.

The best fitting NN usually has a poor validation (generalizability).

NN is a very expensive approach and it should be used only if really needed.

NN Architecture

Neural net suffers from overfitting problems.

Even more parameters

The number of nodes, the architecture, the number of hidden layer etc are NN hyperparameters

Unfortunately, at the current status of knowledge, the best architecture can be found only by trial and error.

The best fitting NN usually has a poor validation (generalizability).

NN is a very expensive approach and it should be used only if really needed.

NN Architecture

Neural net suffers from overfitting problems.

Even more parameters

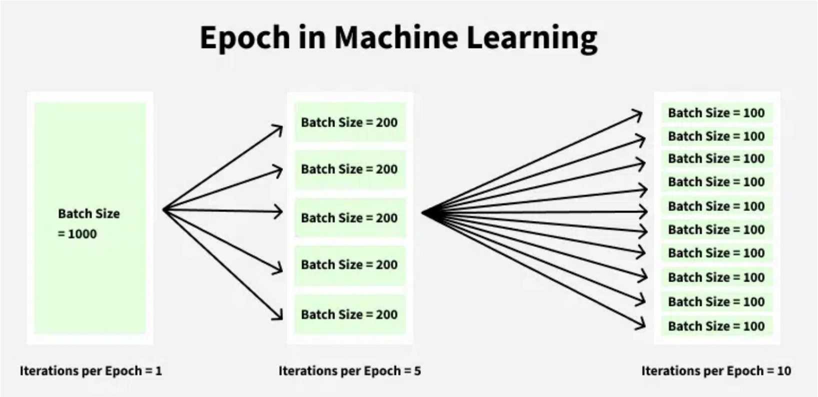
The number of nodes, the architecture, the number of hidden layer etc are NN hyperparameters

Unfortunately, at the current status of knowledge, the best architecture can be found only by trial and error.

The best fitting NN usually has a poor validation (generalizability).

NN is a very expensive approach and it should be used only if really needed.

Epoch



Epoch in Machine Learning
Batch Size
= 1000
Batch Size = 200
Batch Size = 200
Batch Size = 200
Batch Size = 200
Batch Size = 200
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Batch Size = 100
Iterations per Epoch = 1
Iterations per Epoch = 5
Iterations per Epoch = 10

Epoch

Pro

1. Better performance
2. Progress tracking
3. Memory efficiency
4. Improved stopping criteria
5. More effective training

Cons

1. Overfitting risk
2. Computational cost
3. One more hyperparameter

Epoch

Pro

1. Better performance
2. Progress tracking
3. Memory efficiency
4. Improved stopping criteria
5. More effective training

Cons

1. Overfitting risk
2. Computational cost
3. One more hyperparameter

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.

NN types

There are many types of NN:

- ANN (artificial NN), just another name for NN
- DNN (deep) deep neural network
- RNN (recurrent NN) for audio
- CNN (convolutional NN) for images
- Autoencoder (for PCA) to compress to a latent space and decompress data
- Deep autoencoder (for interpretability)
- Physics informed NN (to merge NN to differential equations)
- ... and more ...

tensorflow, pytorch and keras are the most popular and popular libraries for NN.