MACHINE LEARNING

LINEAR REGRESSION

AGENDA

01 Supervised Learning

Training data, hypotheses, an example.

02 LMS Algorithm

Linear model, cost function, gradient descent.

03 Normal Equations

Matrix linear model, deriving the normal equations

04 Probabilistic Interpretation

Assumptions for errors, likelihood function

05 Base Function

Derivation, functions, maximum likelihood

SUPERVISED LEARNING

DEFINITIONS

NOMENCLATURE

SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

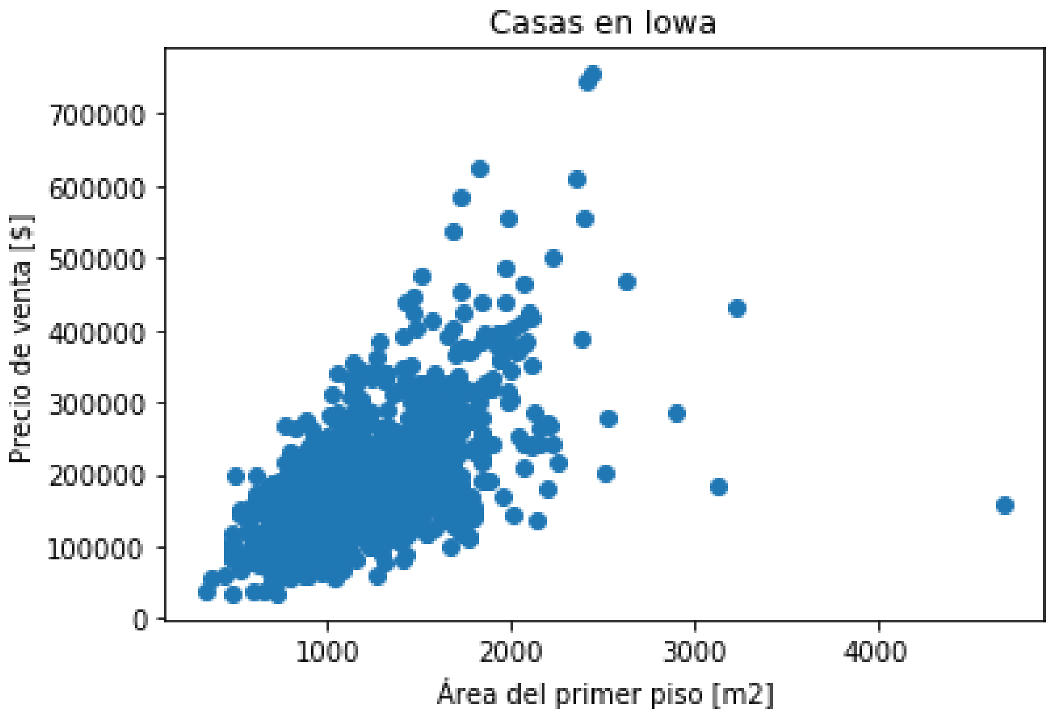
We start with a data set that represents first-floor living space and sales prices for 1,460 homes in Ames, Iowa.

1^{st} floor square feet	Sale Price
856	208500
1262	181500
920	223500
961	140000
1145	250000
⋮	⋮

SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

We graph the 1,460 elements using Python:



SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

The question that naturally arises would be:

HOW DO WE PREDICT THE PRICES FOR OTHER HOMES IN AMES, IOWA?

SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

To answer the question, we must first define the nomenclature that will be used:

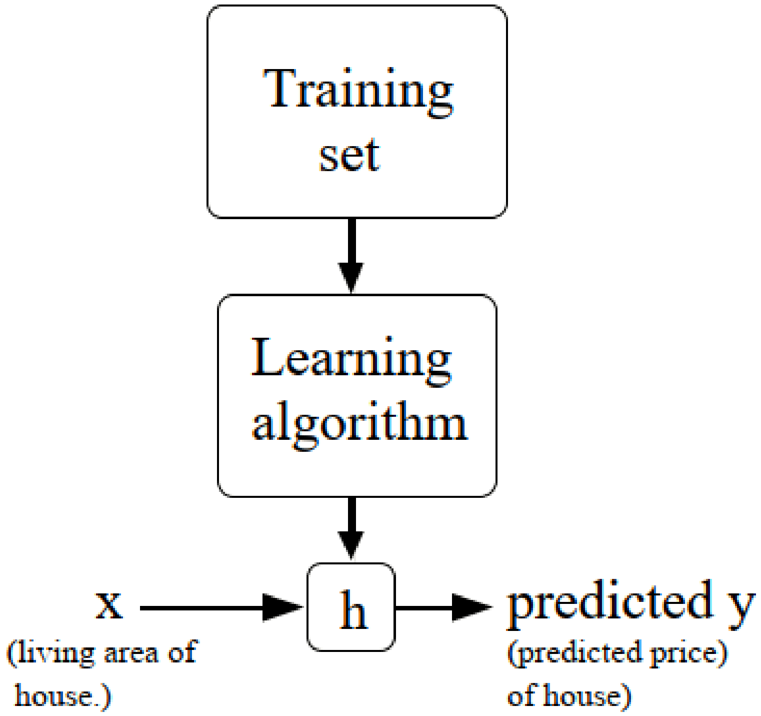
Variable / Symbol	Description	Example
$\boldsymbol{x^{(i)}}$	Input variable or characteristics	The house's first floor area
$\boldsymbol{y^{(i)}}$	Output, target, or response variable	The house's sale price
$(\boldsymbol{x^{(i)}}, \boldsymbol{y^{(i)}})$	Training example	(area, price)
$\boldsymbol{m}$	Number of training examples	1460 houses with their first floor área and their price.
$\{(\boldsymbol{x^{(i)}}, \boldsymbol{y^{(i)}}); \boldsymbol{i} = \boldsymbol{1}, \ldots, \boldsymbol{m}\}$	Training data set	NA
$\boldsymbol{\chi}$	Input value space	NA
$\boldsymbol{\gamma}$	Output value space	

SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

The **main goal** of any **supervised algorithm** is to **learn** a **function** $\boldsymbol{h}$: $\chi \rightarrow \Upsilon$, such that $\boldsymbol{h}\ (\boldsymbol{x})$ can **predict** the corresponding value of $\boldsymbol{y}$.

Where $\boldsymbol{h}$ is defined as the **hypothesis**.



SUPERVISED LEARNING

DEFINITIONS AND NOMENCLATURE

There are **2 types** of **supervised learning problems**:

Type of problem	Description
Regression problem	**y takes continuous values.**
Classification problem	**y takes discrete values**

LMS ALGORITHM

LINEAR MODEL

COST FUNCTION

GRADIENT DESCENT

LMS ALGORITHM

LINEAR MODEL

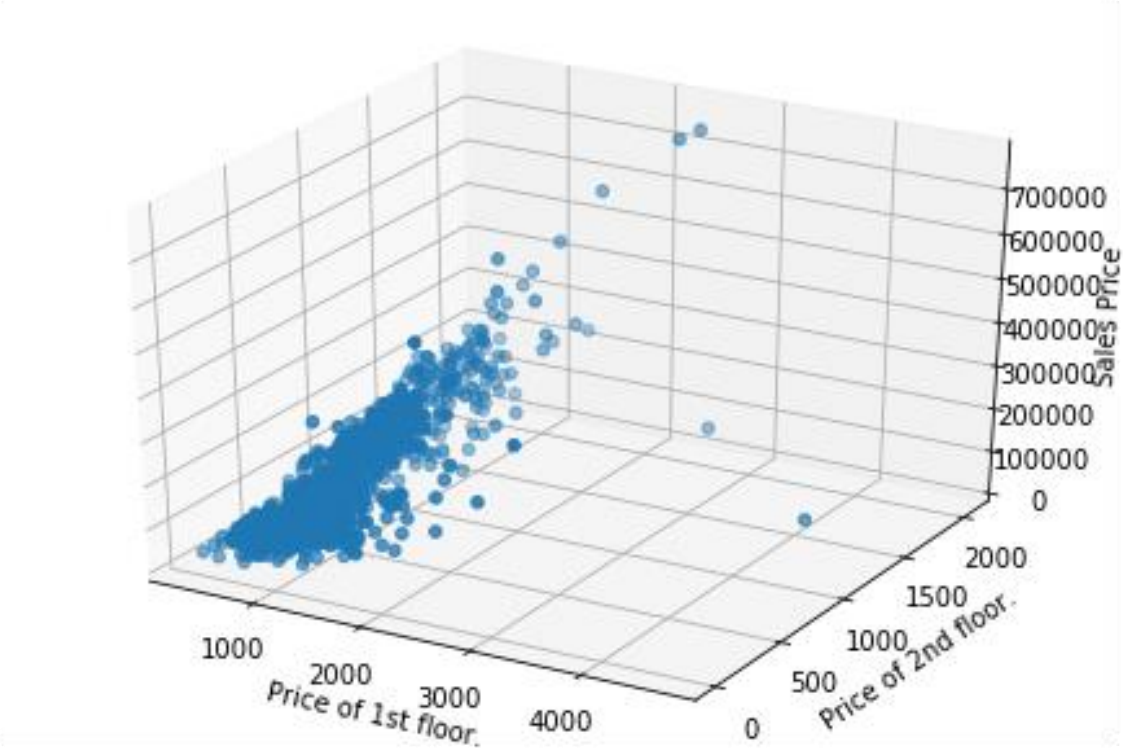
Now let's look at the same model but with another variable: the living space on the second floor.

1^{st} floor square feet	2^{nd} floor square feet	Sale Price
856	854	208500
1262	0	181500
920	866	223500
961	756	140000
1145	1053	250000
$\vdots$	$\vdots$	$\vdots$

In this case each $\boldsymbol{x}^{(i)} \in \mathbb{R}^{\mathbf{2}}$, so it is defined as a vector $\boldsymbol{x}^{(i)} = [\boldsymbol{x}^{(i)}{}_{\mathbf{1}} \ \ \boldsymbol{x}^{(i)}{}_{\mathbf{2}}]$.

LMS ALGORITHM
LINEAR MODEL

We graph the 1,460 elements again using Python:



LMS ALGORITHM

LINEAR MODEL

It is **hypothesized** that the data is distributed in a **linear fashion**, therefore:

$$\boldsymbol{h_w(x) = w_0 + w_1 x_1 + w_2 x_2}$$

In this case the **variables** $\boldsymbol{w_i}$ are defined as the **parameters** or **weights** that **parameterize** the **space** of all **linear functions** that map from $\boldsymbol{\chi}$ to $\boldsymbol{\Upsilon}$.

The expression is **simplified** into **vector form**, where $\boldsymbol{n}$ represents the **number** of **input variables** (omitting x_0),:

$$\boldsymbol{h(x) = \sum_{i=1}^{n} w_i x_i = w^T x}$$

LMS ALGORITHM
COST FUNCTION

Then, the question would be:

WHAT IS THE BEST COMBINATION OF PARAMETERS $\boldsymbol{w}$ THAT RESULTS IN THE BEST HYPOTHESIS $\boldsymbol{h}$?

LMS ALGORITHM

COST FUNCTION

To **answer** the **question**, we need to **measure** the **error** produced by the **estimates** produced by $\boldsymbol{h}$.

We know that we can **measure** the **error** of a single **estimate** $\boldsymbol{i}$ as a **difference between** the estimate $\boldsymbol{h_w}(\boldsymbol{x}^{(i)})$ and the response variable $\boldsymbol{y}^{(i)}$.

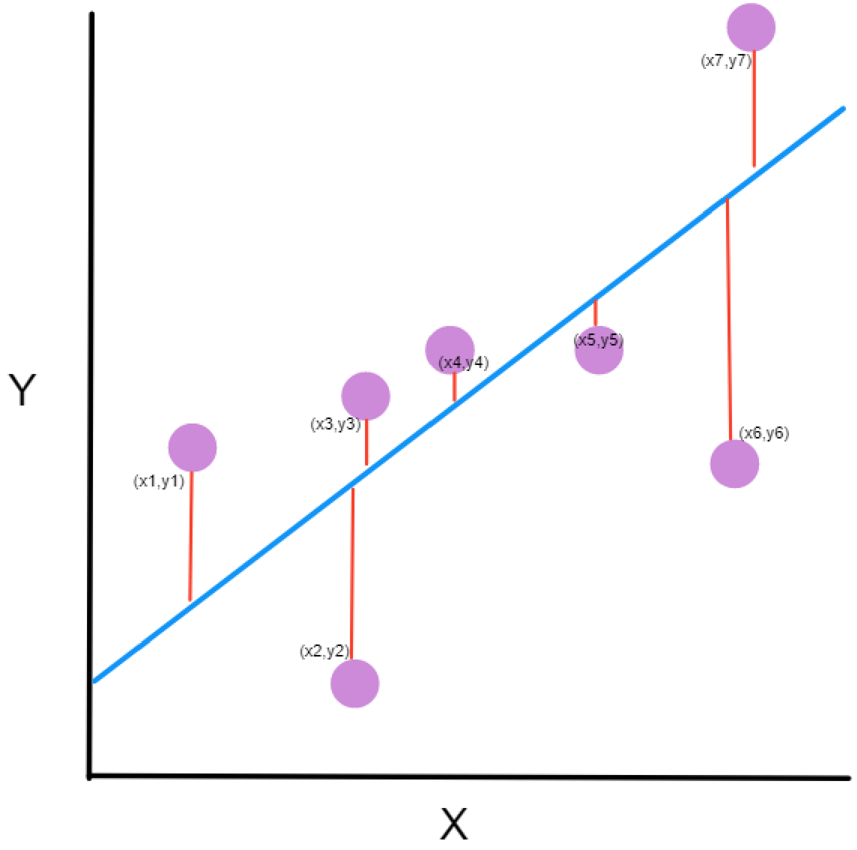
$$\boldsymbol{e = h_w\left(x^{(i)}\right) - y^{(i)}}$$

To ensure positive amounts of error, the answer is squared:

$$\boldsymbol{e^2 = (h_w\left(x^{(i)}\right) - y^{(i)})^2}$$

LMS ALGORITHM
COST FUNCTION

The **error** that we are **calculating graphically** is **interpreted** with an **example**:



The linear model represented by $\boldsymbol{h}$

Response variables $y^{(i)}$

LMS ALGORITHM

COST FUNCTION

So far, we have calculated the **squared error** of a **single training data**. The **mean square error MSE** is calculated **for all training data**.

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}(h_w\left(x^{(i)}\right) - y^{(i)})^2$$

$$MSE = J(w)$$

Where $J(w)$ is the **cost function**.

LMS ALGORITHM

GRADIENT DESCENT

Therefore, the **best combination** of weights $\boldsymbol{w}$ is the one that **minimizes** the **cost function** $\boldsymbol{J}$ ($\boldsymbol{w}$), which **measures** our **mean square error.**

To **find** the **best combination**, let's design a **search algorithm** that **starts** with a **random initial value** of $\boldsymbol{w}$, and **updates** the **values** of $\boldsymbol{w}$ until it **converges** to a **minimum value** of $\boldsymbol{J}$ ($\boldsymbol{w}$). The **constant** $\boldsymbol{\alpha}$ is defined as the **learning rate**.

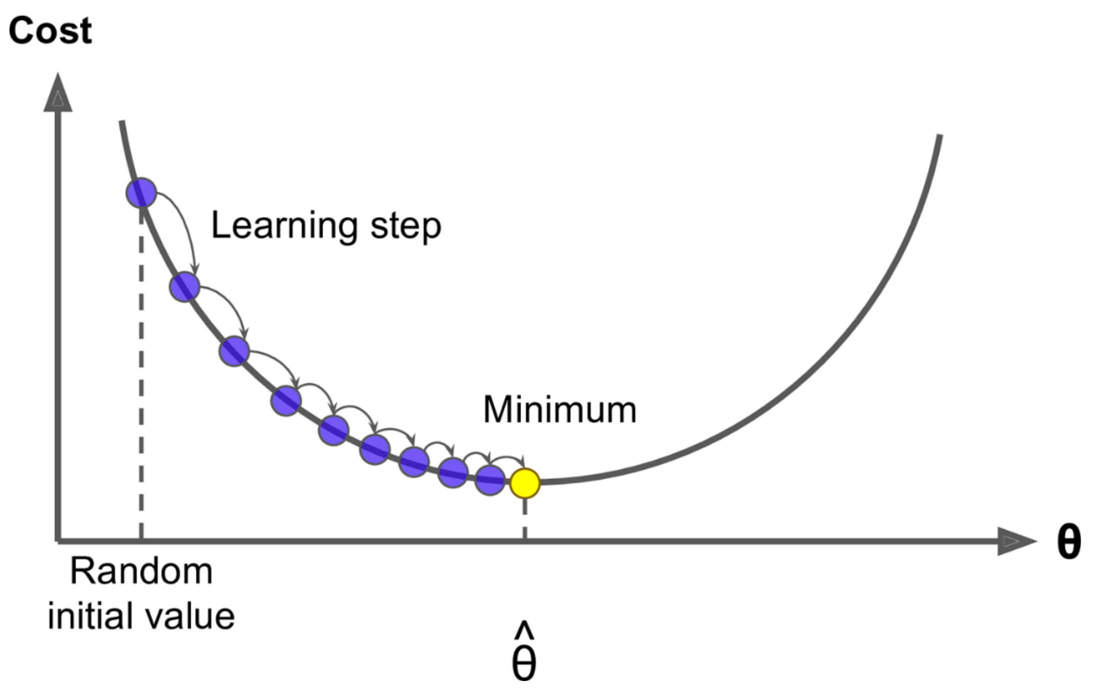
$$\boldsymbol{w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)}$$

NOTE: the above equation **only updates** the **value** of a **single weight** $\boldsymbol{w_j}$ of the $\boldsymbol{n}$ weights that **parameterize** the **linear model**. In reality, **all weights** $\boldsymbol{w_j}$ are **updated simultaneously**.

LMS ALGORITHM

GRADIENT DESCENT

In the **gradient descent optimization algorithm**, we **update** the **weights** in the **direction** with the **greatest decrement** of $\boldsymbol{J}\ (\boldsymbol{w})$.



LMS ALGORITHM

GRADIENT DESCENT

The **derivative** of the **cost function** $\boldsymbol{J}$ $(\boldsymbol{w})$ with respect to a **specific weight** $\boldsymbol{w_j}$ is calculated.

Derive the result:

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m}(h(x) - y)x_j$$

LMS ALGORITHM
GRADIENT DESCENT

Therefore, the **gradient descent weights update** for a **single training data** would look like this:

$$w_j := w_j + \frac{\alpha}{m}\left(y^{(i)} - h\left(x^{(i)}\right)\right)x_j$$

This **equation** is called the **LMS update rule** (“**Least Mean Squares**”) or also the **Widrow-Hoff learning rule**.

This **method** is also called **batch gradient descent**, where **all training data** is **analyzed simultaneously**.

LMS ALGORITHM
GRADIENT DESCENT

For $\boldsymbol{m}$ **training data**, and defining $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$ as **vectors**, the learning rule would look like this:

$$\boldsymbol{w} := \boldsymbol{w} + \frac{\boldsymbol{\alpha}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} \left(\boldsymbol{y}^{(\boldsymbol{i})} - \boldsymbol{h}\left(\boldsymbol{x}^{(\boldsymbol{i})}\right)\right)\boldsymbol{x}^{(\boldsymbol{i})}$$

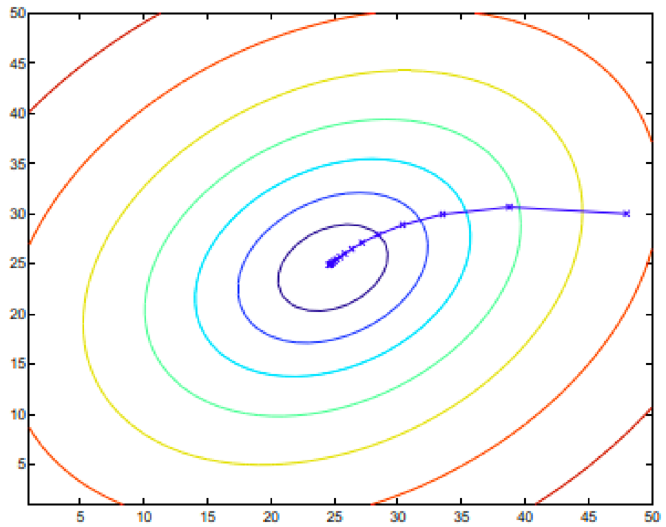
This **equation** is called the **LMS update rule** (“**Least Mean Squares**”) or also the **Widrow-Hoff learning rule**.

LMS ALGORITHM

GRADIENT DESCENT

In general, the **gradient descent method** can suffer from **several local minima**. In this case, for **linear regression models** there is only a **single global minimum**.

Consequently, the **algorithm will always converge** assuming that the **learning rate is not too high.**



LMS ALGORITHM

GRADIENT DESCENT

When the **method** only **observes one training data** at a time, and **updates** the **weights with a single data**, it is called **stochastic or incremental gradient descent**.

$$\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{\alpha}\left(\boldsymbol{y}^{(i)} - \boldsymbol{h}\left(\boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}$$

This **variant** of the algorithm is **used** when it is **very expensive to evaluate** the **update** for **very large data sets** (when $\boldsymbol{m}$ is **very large**), but it has the minimal **divergence** problem.

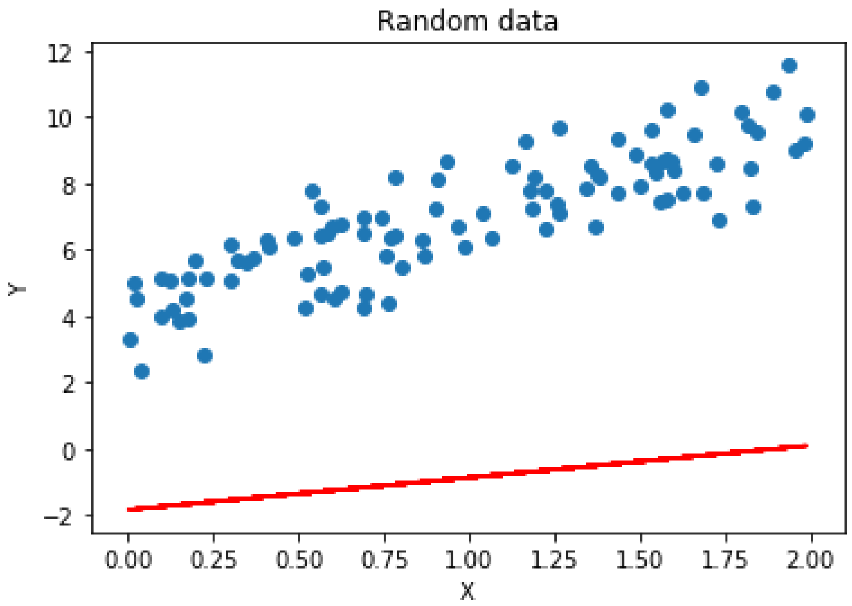
LMS ALGORITHM
GRADIENT DESCENT

Example:

Random training data was generated with a certain degree of error e:

$$y = 3x + 4 + e$$

and both weights were randomly initialized: w_0 and w_1

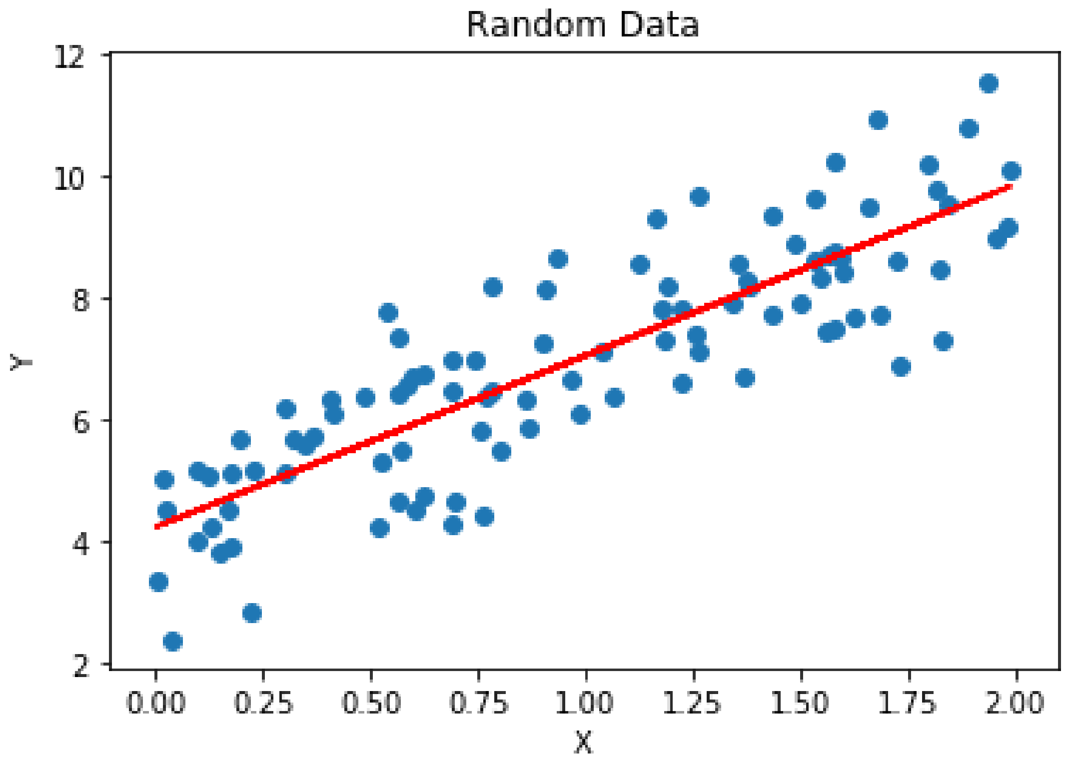


LMS ALGORITHM

GRADIENT DESCENT

Example:

After 1000 iterations the following model was obtained:

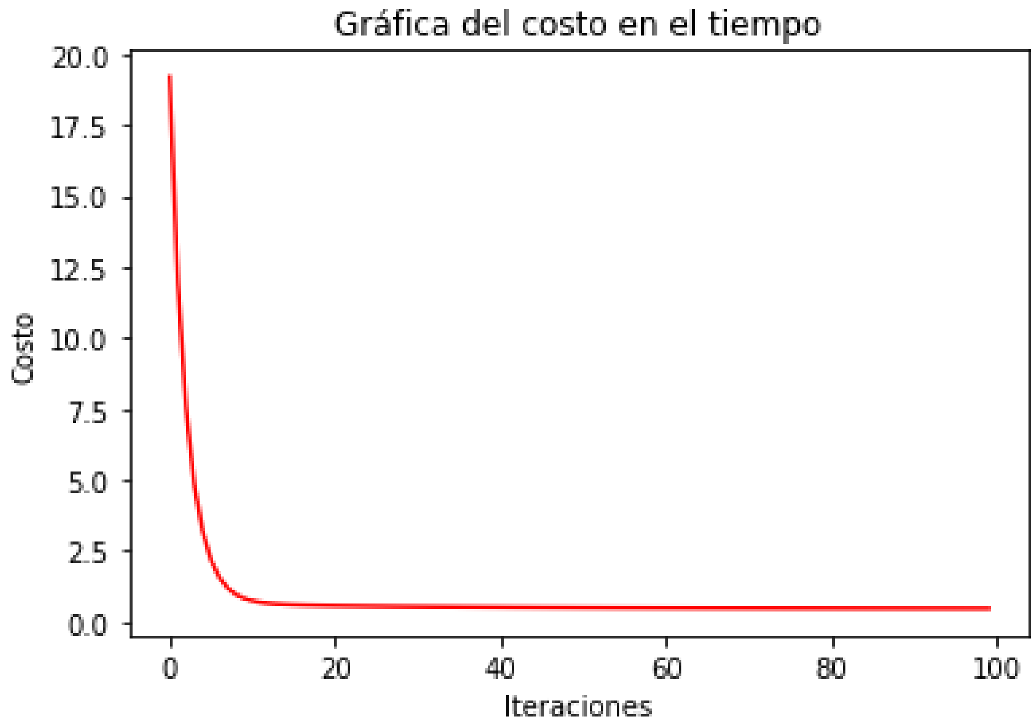


$$w_0 = 4.2174$$
$$w_1 = 2.816$$

LMS ALGORITHM
GRADIENT DESCENT

Example:

Error based on 100 iterations.



NORMAL EQUATIONS

LINEAR MATRIX MODEL DERIVATION

NORMAL EQUATIONS

MATRIX NOTATION

Now we want to **calculate** the **minimum** of the **cost function analytically**. For this, **matrix notation** is introduced, which **allows** the **derivatives** of the **cost function** to be **expressed** in an **elegant way** without getting into so much **verbiage**.

The $\boldsymbol{m}$ **training data** $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}); \boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{m}\}$ is represented as a matrix $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{m} \, x \, \boldsymbol{n}}$, the set of **outputs** as a **vector** $\vec{\boldsymbol{y}} \in \mathbb{R}^{\boldsymbol{m}}$ and the $\boldsymbol{n}$ **weights** $\boldsymbol{w_j}$ as a vector $\vec{\boldsymbol{w}} \in \mathbb{R}^{\boldsymbol{n}}$.

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_n} \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} -\left(\boldsymbol{x}^{(1)}\right)^{\boldsymbol{T}} - \\ -\left(\boldsymbol{x}^{(2)}\right)^{\boldsymbol{T}} - \\ \vdots \\ -\left(\boldsymbol{x}^{(m)}\right)^{\boldsymbol{T}} - \end{bmatrix} \qquad \vec{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \\ \vdots \\ \boldsymbol{y}^{(m)} \end{bmatrix} \qquad \vec{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \\ \vdots \\ \boldsymbol{w_n} \end{bmatrix}$$

NORMAL EQUATIONS

MATRIX NOTATION

The **absolute error function** is represented in **matrix notation** where $h_w\left(x^{(i)}\right) = \left(x^{(i)}\right)^T \vec{w}$:

$$X\vec{w} - \vec{y} = \begin{bmatrix} -\left(x^{(1)}\right)^T \vec{w} - \\ -\left(x^{(2)}\right)^T \vec{w} - \\ \vdots \\ -\left(x^{(m)}\right)^T \vec{w} - \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$X\vec{w} - \vec{y} = \begin{bmatrix} h_w\left(x^{(1)}\right) - y^{(1)} \\ h_w\left(x^{(2)}\right) - y^{(2)} \\ \vdots \\ h_w\left(x^{(m)}\right) - y^{(m)} \end{bmatrix}$$

NORMAL EQUATIONS

MATRIX NOTATION

The **mean square error term** (cost function):

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2$$

can be **represented** in **vector** form using the **property** $z^T z = \sum_i {z_i}^2$

$$J(w) = \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y})$$

NORMAL EQUATIONS

DERIVATION OF EQUATIONS

We have to **obtain** the **derivatives** of $\boldsymbol{J}$ $(\boldsymbol{w})$ with respect to the vector $\boldsymbol{w}$.

$$\nabla_w J(w) = \nabla_w \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y})$$

The **gradient calculates** all the **derivatives** with **respect** to **all** the weights $\boldsymbol{w_j}$ **simultaneously**.

Also, because we **represent** the **training dataset** in **matrix form**, we can **calculate** the **gradient** for **all training data**.

NORMAL EQUATIONS

DERIVATION OF EQUATIONS

Therefore, we obtain:

$$\nabla_w J(w) = \nabla_w \frac{1}{2m}(X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y})$$

DERIVE THE EQUALITY

$$\nabla_w J(w) = \frac{1}{m}\left(X^T X\vec{w} - X^T\vec{y}\right)$$

NORMAL EQUATIONS

DERIVATION OF EQUATIONS

Equating the **derivative to zero** to find the **minimum**, we obtain the **vector of weights** that gives this **minimum**:

$$\nabla_w J(w) = \frac{1}{m}\left(X^T X\vec{w} - X^T\vec{y}\right) = 0$$

$$\vec{w} = \left(X^T X\right)^{-1} X^T\vec{y}$$

PROBABILISTIC INTERPRETATION

ASSUMPTIONS FOR ERRORS

LIKELIHOOD FUNCTION

PROBABILISTIC INTERPRETATION

M O T I V A T I O N

WHY DO WE MINIMIZE THE MEAN QUADRATIC ERROR FUNCTION AND NOT ANOTHER FUNCTION?

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

The **output variables** $\boldsymbol{y^{(i)}}$ can be defined as the **hypothesis** $\boldsymbol{h_w}$ **plus** an **estimation error** $\boldsymbol{\varepsilon^{(i)}}$ that captures the **effects** that we **did not consider** in our **hypothesis** or contemplates the **random errors**.

$$\boldsymbol{y^{(i)} = h_w\left(x^{(i)}\right) + \varepsilon^{(i)}}$$

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

The **following assumptions** are made for the **errors $\varepsilon^{(i)}$** (**random sampling**): **IID**

- Independent Errors - The probability of one error does not affect the probability of other errors.
- Identically distributed errors: errors are sampled from the same probability distribution.
- **Distributed by a Gaussian probability distribution with mean $\mu = 0$ and variance σ^2.**

  $\varepsilon^{(i)} \sim N(0, \sigma^2)$

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

Therefore, the **probability density function** of $\boldsymbol{\varepsilon^{(i)}}$ **is given by**:

$$\boldsymbol{p\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(\varepsilon^{(i)}\right)^2}{2\sigma^2}\right)}}$$

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

WHY PROPOSE A MODEL THAT IS NORMALLY DISTRIBUTED?

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

Furthermore, the **outputs** $\boldsymbol{y}^{(i)}$ and the **inputs** $\boldsymbol{x}^{(i)}$ are set as **random variables**.

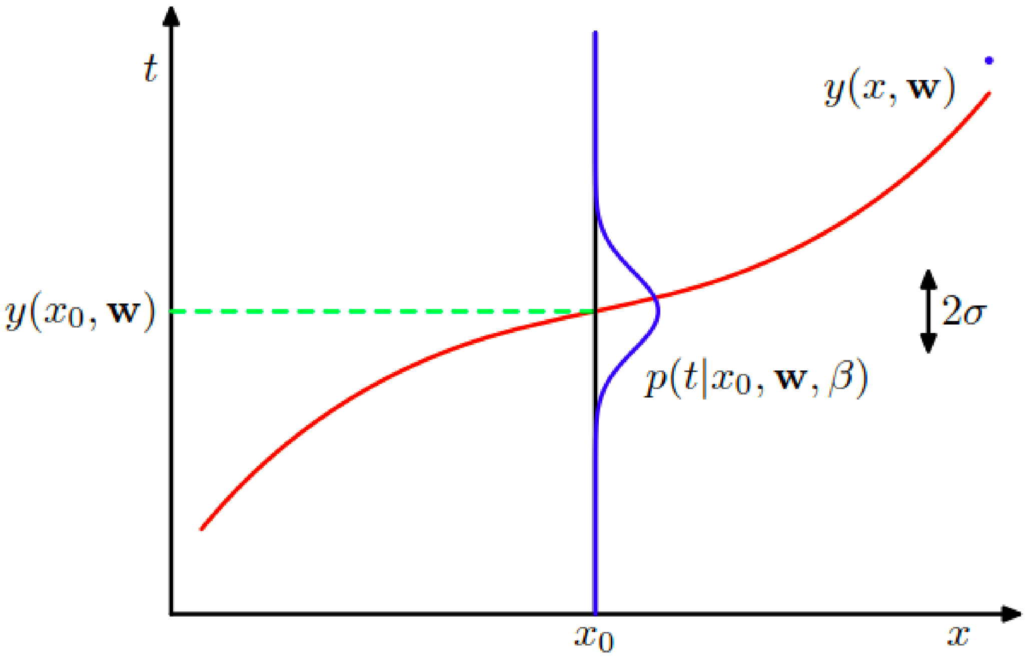
Consequently, the following question is posed:

Given my input data set $\boldsymbol{X}$ and my weight vector $\vec{\boldsymbol{w}}$, what is the probability distribution that models my outputs $\vec{\boldsymbol{y}}$?

PROBABILISTIC INTERPRETATION

ERROR ASSUMPTIONS

Following the **assumptions** from the **normal distribution of errors**, it naturally occurs that the **distribution** of the outputs $\boldsymbol{y^{(i)}}$ follows a **normal form** with mean $\boldsymbol{h_w(x^{(i)})}$ and variance $\boldsymbol{\sigma^2}$.



PROBABILISTIC INTERPRETATION

LIKELIHOOD FUNCTION

Formally it would look like this:

$$p(y^{(i)}/x^{(i)};w) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(y^{(i)}-h_w(x^{(i)})\right)^2}{2\sigma^2}\right)}$$

$$p(y^{(i)}/x^{(i)};w) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(y^{(i)}-w^T x^{(i)}\right)^2}{2\sigma^2}\right)}$$

PROBABILISTIC INTERPRETATION

LIKELIHOOD FUNCTION

We define the **conditional probability** for **all training data** (assuming the **samples** were **collected independently**):

$$p(\vec{y}/X; w) = \prod_{i=1}^{m} p(y^{(i)}/x^{(i)}; w) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(y^{(i)} - w^T x^{(i)}\right)^2}{2\sigma^2}\right)}$$

PROBABILISTIC INTERPRETATION

LIKELIHOOD FUNCTION

This equation is called the **likelihood function**, because it describes the **probability** that our **beliefs** about the **reality** of the **observations** (data) are **true**. In other words, that the **hypothesis** we proposed is **correct**.

$$p(\vec{y}/X;\vec{w}) = \prod_{i=1}^{m} p(y^{(i)}/x^{(i)};\vec{w}) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(y^{(i)}-w^T x^{(i)}\right)^2}{2\sigma^2}\right)}$$

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

PROBABILISTIC INTERPRETATION

LIKELIHOOD FUNCTION

Therefore, we want to **maximize** the **probability** that **our beliefs** about how the **data** is **distributed** (in a **normal** way).

That is, we want to **maximize** the **likelihood function**. From a **frequentist** perspective (the value of $\vec{w}$ is **not random**), we want to **find** the **value** of $\vec{w}$ that **maximizes** the **probability** that the **observations** made of **reality** are **true**.

$$\arg\max_{\vec{w}} L(\vec{w}; X, \vec{y}) = \arg\max_{\vec{w}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(y^{(i)} - w^T x^{(i)}\right)^2}{2\sigma^2}\right)}$$

PROBABILISTIC INTERPRETATION

LOG- LIKELIHOOD

Maximizing a **summation** is much easier than doing so with a **multiplication**

$$\arg\max_{\vec{w}} \, log\big(L(\vec{w})\big) = \arg\max_{\vec{w}} \, log\Big(\prod_{i=1}^{m} p(y^{(i)}/x^{(i)};\vec{w})\Big)$$

$$\arg\max_{\vec{w}} \, log(L(\vec{w})) = \arg\max_{\vec{w}} \sum_{i=1}^{m} log(p(y^{(i)}/x^{(i)};\vec{w}))$$

INTERPRETACIÓN PROBABILÍSTICA

L O G - L O S S

The **maximization problem** is **converted** into a **minimization** one by **scaling** the function with a **minus sign**.

This **function** $-\boldsymbol{log}(\boldsymbol{L}(\vec{\boldsymbol{w}}))$ is called **logarithmic loss.**

$$\underset{\vec{w}}{\arg\min}\ -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} -\sum_{i=1}^{m} \log(p(y^{(i)}/x^{(i)};\vec{w}))$$

$$\underset{\vec{w}}{\arg\min}\ -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} -\sum_{i=1}^{m} \log(\frac{1}{\sqrt{2\pi\sigma}} e^{\left(-\frac{\left(y^{(i)}-w^{T}x^{(i)}\right)^{2}}{2\sigma^{2}}\right)})$$

INTERPRETACIÓN PROBABILÍSTICA

L O G - L O S S

We develop the equation:

$$\underset{\vec{w}}{\arg\min}\ -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} - \sum_{i=1}^{m} log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + log(e^{\left(-\frac{\left(y^{(i)} - w^{T}x^{(i)}\right)^{2}}{2\sigma^{2}}\right)})$$

$$\underset{\vec{w}}{\arg\min}\ -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} - m\, log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \sum_{i=1}^{m} -\frac{\left(y^{(i)} - w^{T}x^{(i)}\right)^{2}}{2\sigma^{2}}$$

INTERPRETACIÓN PROBABILÍSTICA

L O G - L O S S

We develop the equation:

$$\underset{\vec{w}}{\arg\min} \; -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} c + \sum_{i=1}^{m} \frac{\left(y^{(i)} - w^T x^{(i)}\right)^2}{2\sigma^2}$$

$$\underset{\vec{w}}{\arg\min} \; -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left(y^{(i)} - w^T x^{(i)}\right)^2$$

INTERPRETACIÓN PROBABILÍSTICA

MEAN SQUARED ERROR

Minimizing the **logarithmic loss** is the **same** as **minimizing** the **mean square error** $MSE = J(\vec{w})$**:**

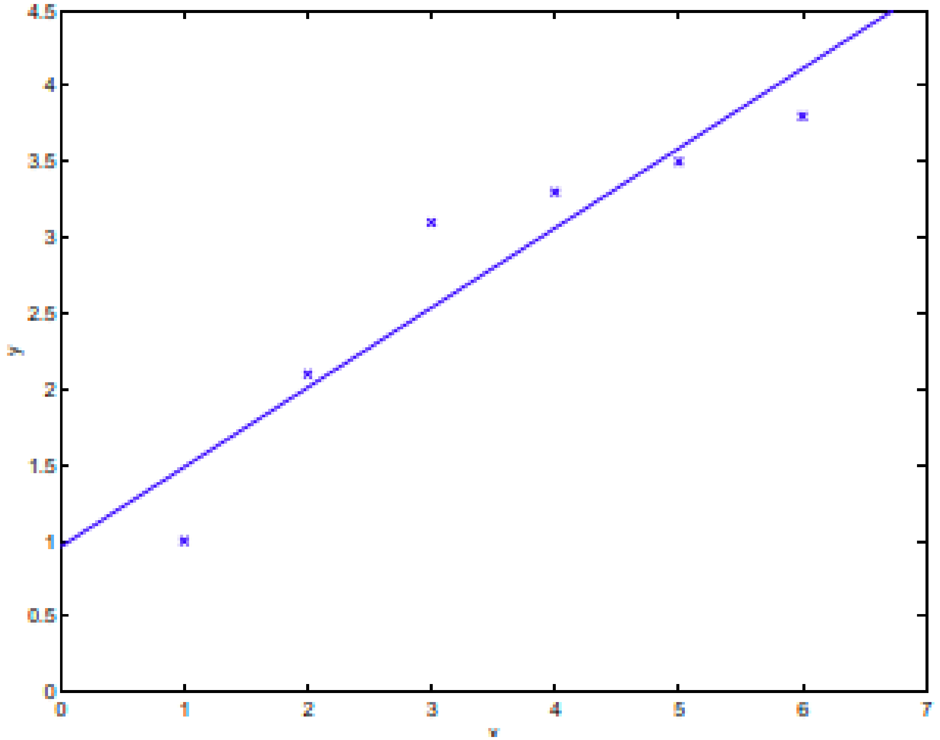
$$\arg\min_{\vec{w}} MSE = \arg\min_{\vec{w}} \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2$$

$$\arg\min_{\vec{w}} -log(L(\vec{w})) =_{\arg\min_{\vec{w}}} \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left(y^{(i)} - w^T x^{(i)}\right)^2$$

BASIS FUNCTIONS

BASIS FUNCTIONS
LINEARITY PROBLEM

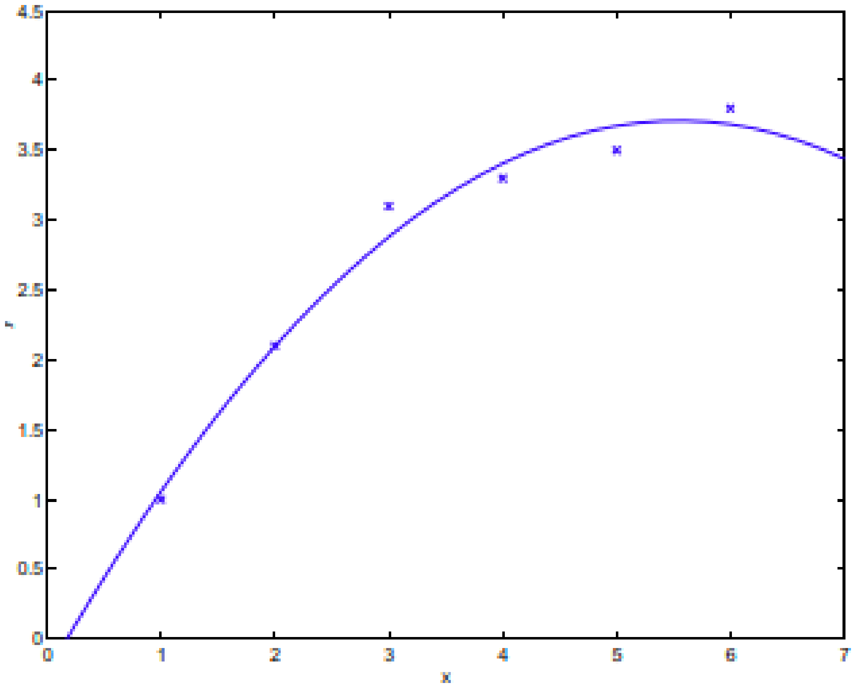
The **real world** has **non-linear** behaviors, so the **linear regression models** seen so far have their **limitations**.



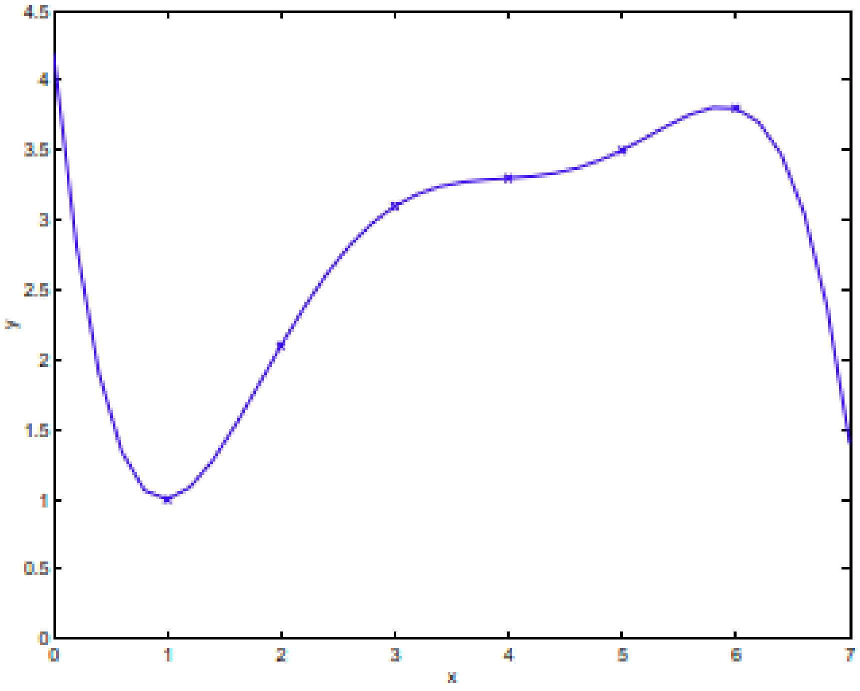
BASIS FUNCTIONS

LINEARITY PROBLEM

We can **establish** our **hypothesis** as a **polynomial model**.



$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$



$$y = \sum_{j=0}^{5} \theta_j x^j$$

B A S I S   F U N C T I O N S

INTRODUCTION TO BASIS FUNCTIONS

If we take this kind of reasoning further, we can **build** models as **linear combinations** of **nonlinear functions** $\boldsymbol{\phi}_j(\boldsymbol{x})$, called **base functions**. Where $\boldsymbol{\phi}: \mathbb{R}^n \rightarrow \mathbb{R}^k$

$$\boldsymbol{h_w}\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=1}^{k} \boldsymbol{w_j \phi_j}\left(\boldsymbol{x}^{(i)}\right)$$

$$\boldsymbol{h_w}\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)$$

NOTA

$\boldsymbol{w} \in \mathbb{R}^k$

$\boldsymbol{x}^{(i)} \in \mathbb{R}^n$

Thus we can **build** a **non-linear model**, which is still **parametrized** by **linear weights** $\boldsymbol{w}$.

BASIS FUNCTIONS

AN INTRODUCTION TO BASIS FUNCTIONS

In a more detailed fashion, we have that:

$$\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) = \begin{bmatrix} \boldsymbol{\phi}_1\left(\boldsymbol{x}^{(i)}\right) \\ \boldsymbol{\phi}_j\left(\boldsymbol{x}^{(i)}\right) \\ \vdots \\ \boldsymbol{\phi}_k\left(\boldsymbol{x}^{(i)}\right) \end{bmatrix}$$

BASIS FUNCTIONS

CLASSICAL LINEAR REGRESSION

In the case of **classical linear regression** we have **for a single** training data $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{\boldsymbol{k}}$ in this case $\boldsymbol{k} = \boldsymbol{n}$:

$$\boldsymbol{\phi}_j\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{x}^{(i)} \qquad \boldsymbol{x}^{(i)} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_k \end{bmatrix}$$

BASIS FUNCTIONS

CLASSICAL LINEAR REGRESSION

In the case of **classical linear regression** we have **for a single** training data $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{\boldsymbol{k}}$ in this case $\boldsymbol{k} = \boldsymbol{n}$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{\phi}_0(\boldsymbol{x}^{(1)}) \\ \boldsymbol{\phi}_1(\boldsymbol{x}^{(i)}) \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{x}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}$$

$$\boldsymbol{h}_{\boldsymbol{w}}\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{w}^{T}\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{w}_0 + \boldsymbol{w}_1(\boldsymbol{x}_1) + \cdots + \boldsymbol{w}_k(\boldsymbol{x}_n)$$

BASIS FUNCTIONS

CLASSICAL LINEAR REGRESSION

In the case of **classical linear regression**, we have **for m** training **data**:

$$\phi(x) = \begin{bmatrix} \phi_0(x^{(1)}) & \phi_1(x^{(1)}) \\ \vdots & \vdots \\ \phi_0(x^{(m)}) & \phi_1(x^{(m)}) \end{bmatrix} = \begin{bmatrix} 1 & {x^{(1)}}^T \\ \vdots & \vdots \\ 1 & {x^{(m)}}^T \end{bmatrix}$$

$$w = [w_0 \quad \cdots \quad w_k]$$

BASIS FUNCTIONS

CLASSICAL LINEAR REGRESSION

In the case of **classical linear regression**, we have **for m** training **data**:

$$h_w(x) = w^T\phi(x) = \begin{bmatrix} w_0 + \cdots + w_k x_n{}^{(1)} \\ \vdots \\ w_0 + \cdots + w_k x_n{}^{(m)} \end{bmatrix}$$

$$h_w(x) = \begin{bmatrix} h_w\left(x^{(1)}\right) \\ \vdots \\ h_w\left(x^{(m)}\right) \end{bmatrix}$$

BASIS FUNCTIONS
DESIGN MATRIX

Therefore, the **design matrix** for any $\boldsymbol{\phi}\ (\boldsymbol{x})$ would be given by:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{\phi}_0(\boldsymbol{x}^{(1)}) & \cdots & \boldsymbol{\phi}_{k-1}(\boldsymbol{x}^{(1)}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_0(\boldsymbol{x}^{(m)}) & \cdots & \boldsymbol{\phi}_{k-1}(\boldsymbol{x}^{(m)}) \end{bmatrix}$$

$$\boldsymbol{x}^{(i)} \in \mathbb{R}^{n}$$

Where each row of the matrix $\boldsymbol{\phi}\ (\boldsymbol{x})$ is given by $\boldsymbol{\phi}_j = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^T$

BASIS FUNCTIONS

LINEAR BASIS FUNCTIONS

Classic linear functions:

$$\phi_j\left(x^{(i)}\right) = x^{(i)}; x^{(i)} \in \mathbb{R}^{n+1}$$

$$w = \begin{bmatrix} w_0 \\ \vdots \\ w_k \end{bmatrix}; k = n$$

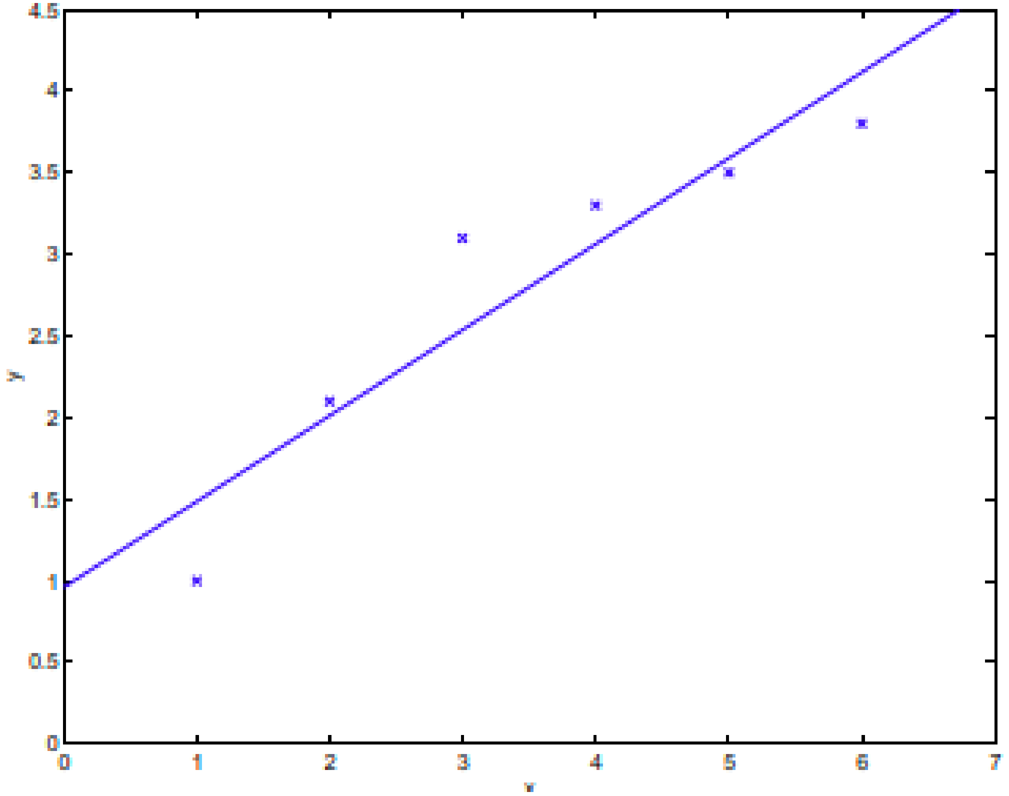
$$\phi(x) = \begin{bmatrix} 1 & {x^{(1)}}^T \\ \vdots & \vdots \\ 1 & {x^{(m)}}^T \end{bmatrix} = \begin{bmatrix} 1 & \cdots & {x_n}^{(1)} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & {x_n}^{(m)} \end{bmatrix}$$

$$h_w(x) = w^T \phi(x) = \begin{bmatrix} w_0 + w_1 {x_1}^{(1)} + \cdots + w_k {x_n}^{(1)} \\ \vdots \\ w_0 + w_1 {x_1}^{(m)} + \cdots + w_k {x_n}^{(m)} \end{bmatrix}$$

BASIS FUNCTIONS

LINEAR BASIS FUNCTIONS

Classic linear functions:



BASIS FUNCTIONS

POLYNOMIAL BASIS FUNCTIONS

Polynomial basis functions:

$$\boldsymbol{\phi_j(x) = \left(x^{(i)}\right)^j ; x^{(i)} \in \mathbb{R}^n}$$

$$\boldsymbol{w = \begin{bmatrix} w_0 \\ \vdots \\ w_{k*n+1} \end{bmatrix}}$$

$$\boldsymbol{\phi(x) = \begin{bmatrix} 1 & {x^{(1)}}^T & \dots & \left({x^{(1)}}^T\right)^{(k-1)} \\ \vdots & \vdots & & \vdots \\ 1 & {x^{(m)}}^T & \dots & \left({x^{(m)}}^T\right)^{(k-1)} \end{bmatrix}}$$

$$\boldsymbol{h_w(x) = w^T\phi(x) = \begin{bmatrix} w_0 + w_1 {x_1}^{(1)} + \dots + w_{k*n}\left({x_n}^{(1)}\right)^{k-1} \\ \vdots \\ w_0 + w_1 {x_1}^{(m)} + \dots + w_{k*n}\left({x_n}^{(m)}\right)^{k-1} \end{bmatrix}}$$

BASIS FUNCTIONS

POLYNOMIAL BASIS FUNCTIONS

Polynomial basis functions: concrete example polynomial degree 2 and two training data $\boldsymbol{m} = \mathbf{2}$

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \boldsymbol{w}_3 \\ \boldsymbol{w}_4 \end{bmatrix}$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{1} & \boldsymbol{x}_1{}^{(1)} & \boldsymbol{x}_2{}^{(1)} & \left(\boldsymbol{x}_1{}^{(1)}\right)^2 & \left(\boldsymbol{x}_2{}^{(1)}\right)^2 \\ \mathbf{1} & \boldsymbol{x}_1{}^{(2)} & \boldsymbol{x}_2{}^{(2)} & \left(\boldsymbol{x}_1{}^{(2)}\right)^2 & \left(\boldsymbol{x}_2{}^{(2)}\right)^2 \end{bmatrix}$$

BASIS FUNCTIONS

POLYNOMIAL BASIS FUNCTIONS

Polynomial basis functions: concrete example **polynomial degree** 2 and two **training data** m = 2 with **2 characteristics**.

$$h_w(x) = w^T \phi(x)$$

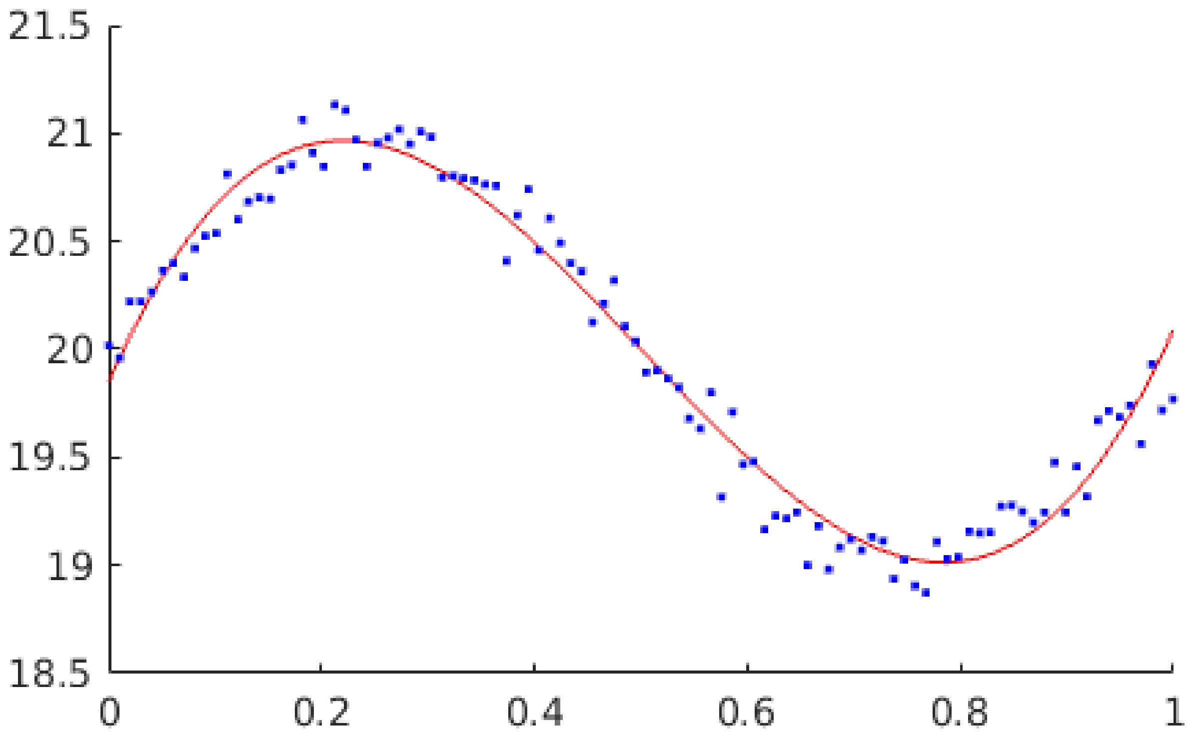
$$h_w(x) = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \left(x_1^{(1)}\right)^2 & \left(x_2^{(1)}\right)^2 \\ 1 & x_1^{(2)} & x_2^{(2)} & \left(x_1^{(2)}\right)^2 & \left(x_2^{(2)}\right)^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$h_w(x) = \begin{bmatrix} w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + w_3\left(x_1^{(1)}\right)^2 + w_4\left(x_2^{(1)}\right)^2 \\ w_0 + w_1 x_1^{(2)} + w_2 x_2^{(2)} + w_3\left(x_1^{(2)}\right)^2 + w_4\left(x_2^{(2)}\right)^2 \end{bmatrix}$$

B A S I S   F U N C T I O N S

POLYNOMIAL BASIS FUNCTIONS

Polynomial basis functions:



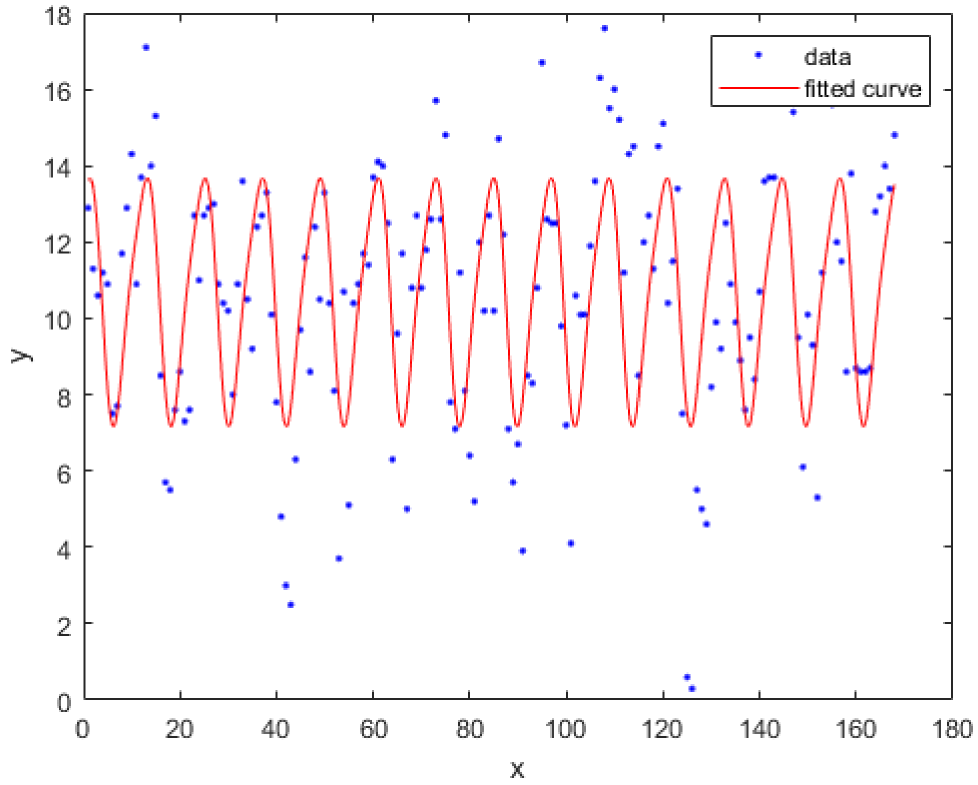
BASIS FUNCTIONS

FOURIER SERIES

Fourier series

$$\boldsymbol{\phi_0(x) = 1}$$

$$\boldsymbol{\phi_j(x)} = \cos(\boldsymbol{\varpi_j x^{(i)} + \psi_j})\,;\boldsymbol{j > 0}$$

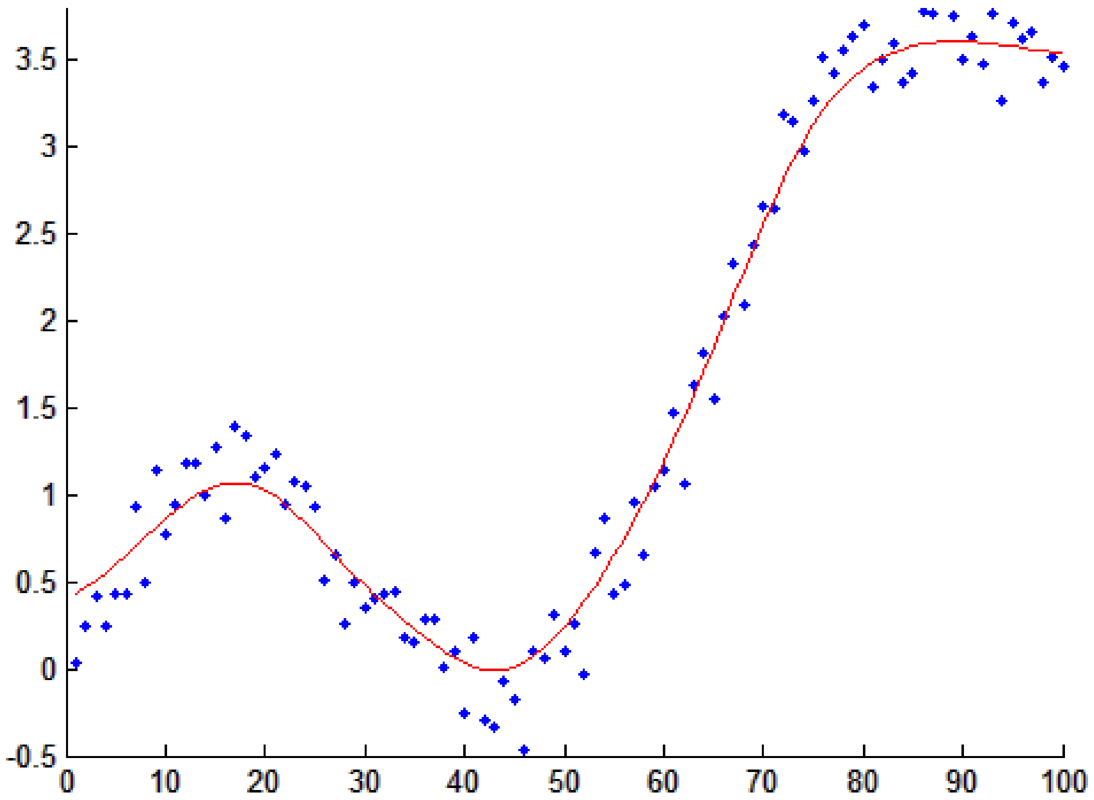


BASIS FUNCTIONS

RADIAL BASIS

Radial basis function:

$$\phi_j(x) = e^{-\frac{1}{2l}\sum_{r=1}^{n}\left(x_r^{(i)}-\mu_{j,r}\right)^2}$$



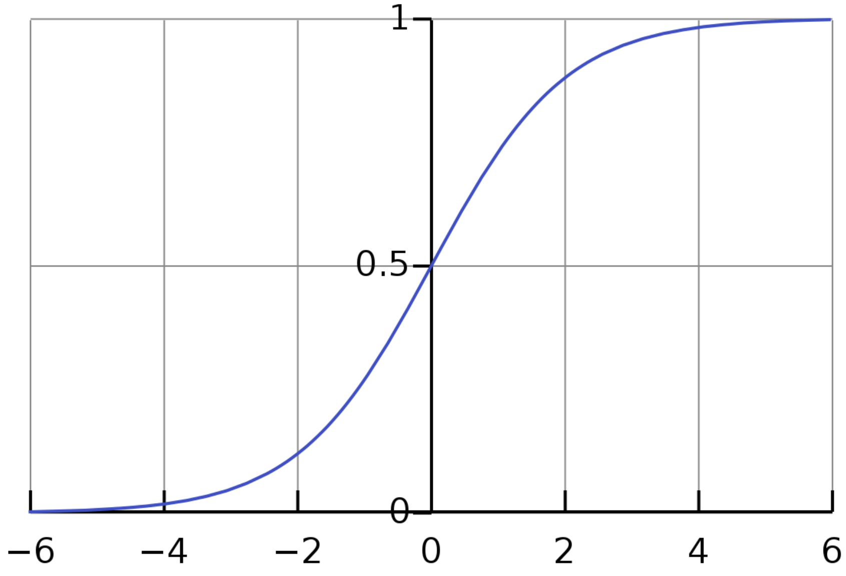
BASIS FUNCTIONS

SIGMOID FUNCTION

Sigmoid function:

$$\phi_j\left(x^{(i)}\right) = \sigma\left(\frac{x^{(i)} - \mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$



B A S I S   F U N C T I O N S

M A X I M U M   L I K E L I H O O D

The **same** estimation **model** is proposed **plus** an **error**:

$$\boldsymbol{y}^{(i)} = \boldsymbol{h}_{\boldsymbol{w}}\left(\boldsymbol{x}^{(i)}\right) + \boldsymbol{\varepsilon}^{(i)}$$

Where:

$$\boldsymbol{h}_{\boldsymbol{w}}\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{w}^{T}\boldsymbol{\phi}(\boldsymbol{x})$$

BASIS FUNCTIONS

MAXIMUM LIKELIHOOD

Assuming again that the **error** is **normally distributed** with **mean 0** and **variance $\boldsymbol{\beta^{-1}}$**. It is defined for $\boldsymbol{m}$ **training data**:

$$p(\vec{y}/X, w, \beta) = \prod_{i=1}^{m} p(y^{(i)}/x^{(i)}; w) = \prod_{i=1}^{m} \sqrt{\frac{\beta}{2\pi}}\, e^{\left(-\frac{\beta}{2}\left(y^{(i)} - w^T \phi\left(x^{(i)}\right)\right)^2\right)}$$

BASIS FUNCTIONS

MAXIMUM LIKELIHOOD

Since the **likelihood function** is **parameterized** in $\boldsymbol{w}$ and $\boldsymbol{\beta}$ the logarithmic likelihood is written like this:

$$\log p(\vec{y}/w, \beta) = \log \prod_{i=1}^{m} \sqrt{\frac{\beta}{2\pi}} e^{\left(-\frac{\beta}{2}\left(y^{(i)} - w^T \phi\left(x^{(i)}\right)\right)^2\right)}$$

$$\log p(\vec{y}/w, \beta) = \sum_{i=1}^{m} \log \sqrt{\frac{\beta}{2\pi}} e^{\left(-\frac{\beta}{2}\left(y^{(i)} - w^T \phi\left(x^{(i)}\right)\right)^2\right)}$$

BASIS FUNCTIONS

MAXIMUM LIKELIHOOD

Developing:

$$\log p(\vec{y}/w,\beta) = \sum_{i=1}^{m} \log\sqrt{\frac{\beta}{2\pi}} + \log e^{\left(-\frac{\beta}{2}\left(y^{(i)} - w^T\phi\left(x^{(i)}\right)\right)^2\right)}$$

$$\log p(\vec{y}/w,\beta) = m\log\sqrt{\frac{\beta}{2\pi}} - \sum_{i=1}^{m}\frac{\beta}{2}\left(y^{(i)} - w^T\phi\left(x^{(i)}\right)\right)^2$$

BASIS FUNCTIONS

MAXIMUM LIKELIHOOD

Developing:

$$\log p(\vec{y}/w,\beta) = \frac{m}{2}\log\beta - \frac{m}{2}\log 2\pi - \sum_{i=1}^{m}\frac{\beta}{2}\left(y^{(i)} - w^T\phi\left(x^{(i)}\right)\right)^2$$

$$\log p(\vec{y}/w,\beta) = \frac{m}{2}\log\beta - \frac{m}{2}\log 2\pi - \beta E_D(w)$$

$$E_D(w) = \frac{1}{2}\left(y^{(i)} - w^T\phi\left(x^{(i)}\right)\right)^2$$

BASIS FUNCTIONS

MAXIMUM LIKELIHOOD

Developing:

$$-\log p(\vec{y}/w,\beta) = -\frac{m}{2}\log\beta + \frac{m}{2}\log 2\pi + \beta E_D(w)$$

$$\underset{\vec{w}}{\arg\min}\; -log(L(\vec{w})) =_{\underset{\vec{w}}{\arg\min}} \frac{\beta}{2}\sum_{i=1}^{m}\left(y^{(i)} - w^T\phi\left(x^{(i)}\right)\right)^2$$

BASIS FUNCTIONS

NORMAL EQUATIONS

Calculating the **gradient**, **setting** it equal to **zero** and solving for the **vector** $\boldsymbol{w}$:

$$\boldsymbol{w} = \left(\boldsymbol{\phi}^T \boldsymbol{\phi}\right)^{-1} \boldsymbol{\phi}^T \vec{\boldsymbol{y}}$$