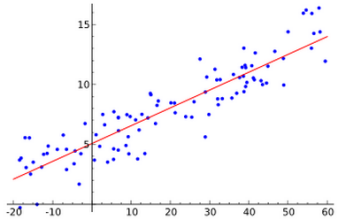
3. Linear models

3.1 Linear regression

Supervised learning: Regression
Input variables: continuous
Output variable: continuous

It is a linear model to predict an output variable based on one or more input variables. It could be:

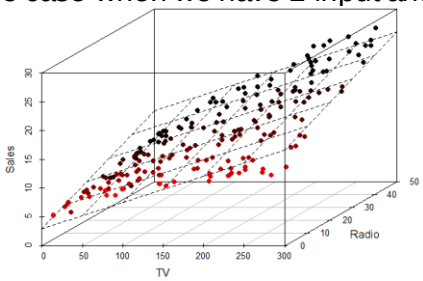
- **A line**, it is the case when we have 1 input and 1 output variable. See examples.



$$\boldsymbol{h(x) = \beta_1 x + \beta_0}$$

For example:
$Grade = 0.5 * hrs\_spent\_study + 4$

- **A plane**, it is the case when we have 2 input and 1 output variable. See examples.



$$\boldsymbol{h(x) = \beta_2 x_2 + \beta_1 x_1 + \theta_0}$$

For example:
$Sales = 0.54 * TV + 3\ Radio + 4$

- A **hyperplane**, it is the case when we have more than 2 input variables and 1 output variable.

$$h(x) = \beta_M x_M + \beta_{M-1} x_{M-1} + \cdots + \beta_2 x_2 + \beta_1 x_1 + \beta_0 = \sum_{i=1}^{M} \beta_i x_i + \beta_0$$

The objective is to minimize a cost function J. It consists into the difference between the real output and the prediction of the model for all the samples:

$$\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i=1}^{M} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Remembering, a dataset is composed by pairs of input-output data $< x^{(i)}, y^{(i)} >$, where $x^{(i)} \in \mathbb{R}^M$ is a vector that contains the values of features, and $y^{(i)} \in \mathbb{R}$ contains the output value.

	fixed acidity	volatile acic	citric acid	residual sug	chlorides	free sulfur c	total sulfur	density	pH	sulphates	alcohol	quality	
	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	
	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5	
$x^{(i)}$	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5	$y^{(i)}$
	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6	
	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	
	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5	
	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5	
	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7	

To optimize the computation process, we create the new variable $\mathcal{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$ and we rewrite the equation $h(x)$ using the dot product.

$$h(x) = \sum_{i=1}^{M} \beta_i x_i + \beta_0$$
$$h(x) = \beta_M x_M + \beta_{M-1} x_{M-1} + \cdots + \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

$$h(x) = [\beta_0 \quad \beta_1 \quad \beta_2 \quad \dots \quad \beta_M] \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

$$h(x) = \beta^T \mathcal{X}$$

For calculating the β values using the samples $< x^{(i)}, y^{(i)} >$ we solve the following optimization problem:

$$\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i=1}^{M} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2$$

It can be solved using Gradient Descent or Conjugate Gradient.

$$\nabla J(\beta) = \begin{bmatrix} \sum_{i=1}^{M} \left(h\left(x^{(i)}\right) - y^{(i)} \right) \\ \sum_{i=1}^{M} \left(h\left(x^{(i)}\right) - y^{(i)} \right) \left(x_1^i\right) \\ \vdots \\ \sum_{i=1}^{M} \left(h\left(x^{(i)}\right) - y^{(i)} \right) \left(x_n^i\right) \end{bmatrix}$$

$$\nabla^2 J(\beta) = \begin{bmatrix} M & \sum_{i=1}^{M} x_1^{(i)} & \dots & \sum_{i=1}^{M} x_n^{(i)} \\ \sum_{i=1}^{M} x_1^{(i)} & \sum_{i=1}^{M} x_1^{(i)} x_1^{(i)} & \dots & \sum_{i=1}^{M} x_1^{(i)} x_n^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{M} x_n^{(i)} & \sum_{i=1}^{M} x_1^{(i)} x_n^{(i)} & \dots & \sum_{i=1}^{M} x_n^{(i)^2} \end{bmatrix}$$

Variables transformation

Sometimes, a variable transformation can be very helpful to improve the regression.

For example: Original data on house sale.

x_1	x_2	x_3	x_4	y
Lot front	Lot depth	# of bedrooms	# floors	Price ($1000)
10	10	3	2	1200
6	15	2	2	700
15	20	3	1	1500
20	20	4	2	4500
10	6	1	1	350

It could be a good idea to transform the variables x1 and x2 to create a new one, in this case, we can have this data set:

x_1	x_2	x_3	y
Lot area	# of bedrooms	# floors	Price ($1000)
100	3	2	1200
90	2	2	700
300	3	1	1500
400	4	2	4500
60	1	1	350

The variables transformation can be performed using: multiplication, sum, trigonometric functions, etc.
Important: The transformation needs to be done **before** to fit the linear regression model.

Feature Scaling and Normalization

For improving the optimization process (Gradient Descent or Conjugate Gradient), a preprocessing step could be performed.

Feature scaling: It consists into transform the variables to have a range of 1:

$$x = \frac{x}{range\ (x)}$$

Normalization: It consists into transform the variables for having a normalized range (based on the normal distribution):

$$x = \frac{x - average\ (x)}{standard\ deviation\ (x)}$$

Systems of linear equations

Another way to solve the linear regression problem is using systems of linear equations:

	x_1 Size (feet2)	x_2 Number of bedrooms	x_3 Number of floors	x_4 Age of home (years)	y Price ($1000)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	1035	3	2	20	250

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

$$X\ \beta = Y$$

$$\begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 1035 & 3 & 2 & 20 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 250 \end{bmatrix}$$

We can solve that system using:

First approach:

$$X\ \beta = Y$$
$$\cancel{X^{-1}X}\ \beta = X^{-1}Y$$
$$\beta = X^{-1}Y$$

But X is not always is square, so, it is not possible to calculate its inverse X^{-1}

Second approach:

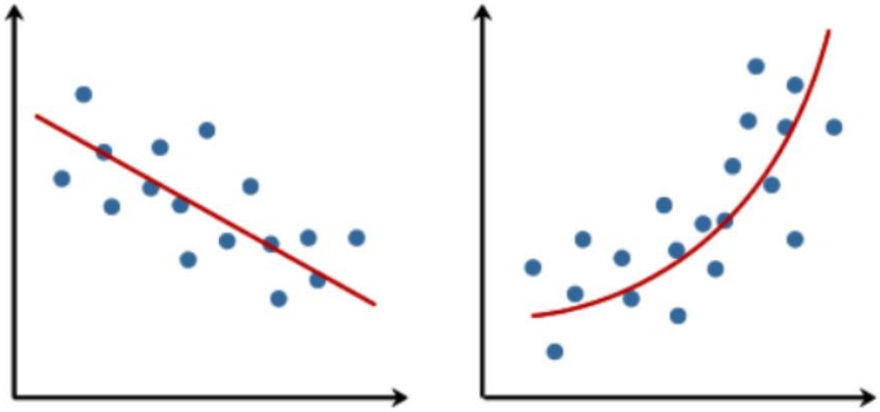
$$X\ \beta = Y$$
$$X^T X\ \beta = X^T Y$$
$$\cancel{(X^T X)^{-1} X^T X}\ \beta = (X^T X)^{-1} X^T Y$$
$$\boldsymbol{\beta = (X^T X)^{-1} X^T Y}$$

It is a simple approach, but the complexity is higher than Conjugate Gradient:

	Optimization with Conjugate Gradient	Linear system
Complexity	$O(n^2)$	$O(n^3)$

It is important to take care with the matrix $\boldsymbol{X^T X}$, sometimes it can be singular (it does not have an inverse matrix). It could be for having features or samples linearly dependents.

Nonlinear problems



Sometimes the problems are nonlinear. In those cases, we can prove feature transformation **before** fitting the model.

For example, if we have the features input features: x_1, x_2 the linear model could be:

$$h(x_1,x_2) = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

And a nonlinear model could be:

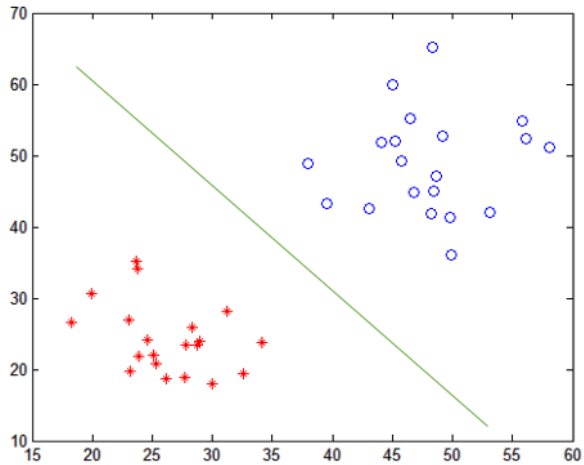
$$h(x_1,x_2) = \beta_5 {x_1}^2 {x_2}^2 + \beta_4 {x_2}^2 + \beta_3 {x_1}^2 + \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

3.2 Logistic Regression

Supervised learning: Classification
Input variables: continuous
Output variable: discrete

Logistic regression is a model for binary classification. It consists into fitting a point (1d), a line (2d), a plane (3d) or a hyperplane (+3D) to separate the data of different classes.

The correct name should be "**logistic classification**", but the name has been keept because was the original proposal of the author, Joseph Berkson.



We can see the models as follows:

1D, a point: $h(X) = w_0 + w_1 x$

2D, a line $h(X) = w_0 + w_1 x_1 + w_2 x_2$

En 3D, un plano $h(X) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_2$

In general, a hyperplane:

$$\mathbf{h(X) = W^T X}$$

where:

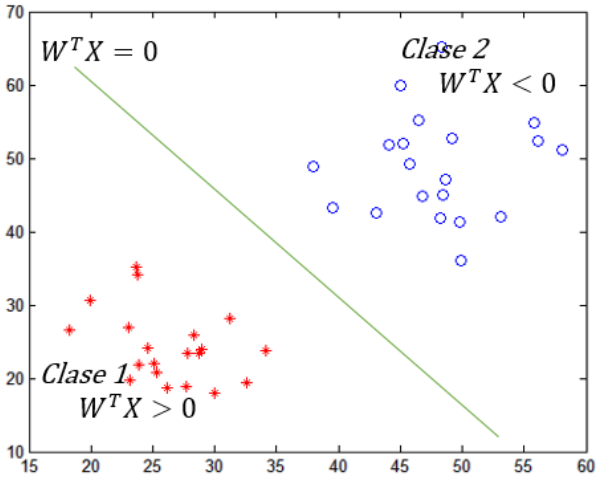
$\boldsymbol{W = [w_0, w_1, \dots, w_n]^T}$

$\boldsymbol{X = [1, x_1, \dots, x_n]^T}$

The objective is to identify the class using the sign of the dot product:

$\boldsymbol{W^T X > 0 \rightarrow \ Class\ 1}$

$\boldsymbol{W^T X < 0 \rightarrow \ Class\ 2}$



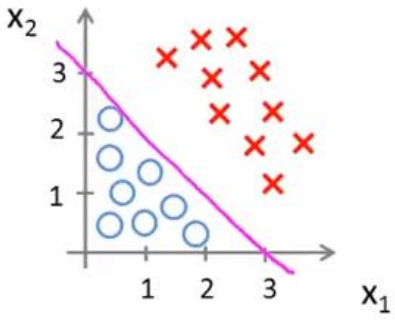
Transformation of the decision limit

The original model of logistic regression is a hyperplane:

$$h(x) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

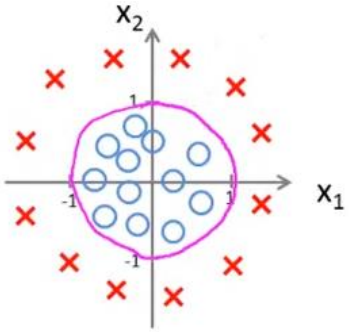
$$= [w_0 \quad w_1 \quad \dots \quad w_n] \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= W^T X$$



However, if we transform the variables **before** to fitting the linear model, we can have a nonlinear model:

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$$

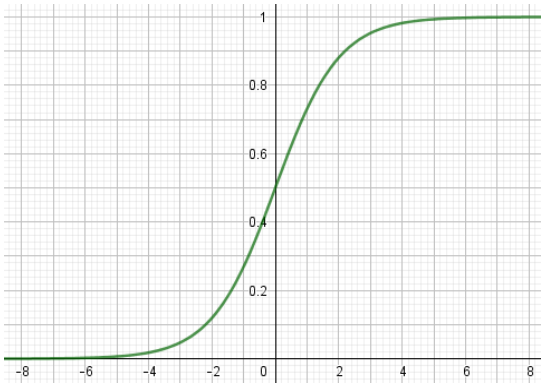


Likelihood and sigmoid function

To see the problem as the likelihood (probability) of belonging to a class, then we follow the next rules:

- The classes' labels are 0 and 1.
- The sigmoid function is used to add the probability effect.
- The log function is used to define the optimization problem.

$$h(x) = \frac{1}{1 + e^{-W^T X}}$$



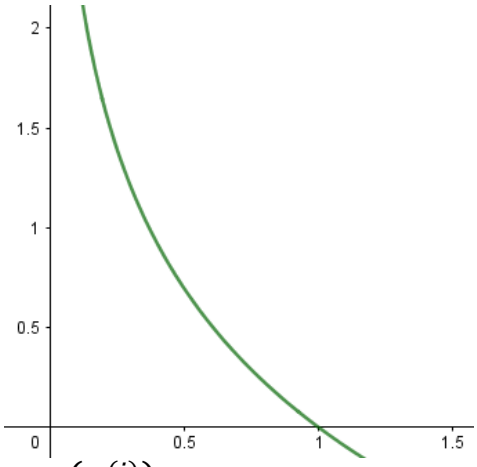
Optimization

To calculate the parameter values W it is necessary to solve the next optimization problem:

$$\min_{\theta} J(\theta) = \frac{1}{2}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right)$$

$If\ y = 1$

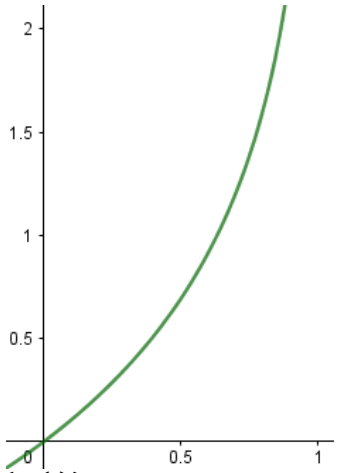
$$Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) = -\log\left(h\left(x^{(i)}\right)\right)$$



If $h\left(x^{(i)}\right) \to 0$ then $cost \to \infty$

$If\ y = 0$

$$Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) = -\log\left(1 - h\left(x^{(i)}\right)\right)$$



If $h\left(x^{(i)}\right) \to 1$ then $cost \to \infty$

To simplify:

$$Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(h\left(x^{(i)}\right)\right) & if\ y = 1 \\ -\log\left(1 - h\left(x^{(i)}\right)\right) & if\ y = 0 \end{cases}$$

It is used:

$$Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) = -\left[y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

If $y^{(i)} = 1$ only the green term will be different to 0
Si $y^{(i)} = 0$ only the blue term will be different to 0

The optimization problem is redefined as:

$$\min_{W} J(W) = -\sum_{i=1}^{M}\left[y^{(i)}\log\left(\frac{1}{1+e^{-W^T X^i}}\right) + (1-y^{(i)})\log\left(1-\frac{1}{1+e^{-W^T X^{(i)}}}\right)\right]$$

$$\nabla J(W) = \begin{bmatrix} \sum_{i=1}^{M}\left(\frac{1}{1+e^{-W^T X^{(i)}}} - y^{(i)}\right) \\ \sum_{i=1}^{M}\left(\frac{1}{1+e^{-W^T X^{(i)}}} - y^{(i)}\right)(x_1^i) \\ \vdots \\ \sum_{i=1}^{M}\left(\frac{1}{1+e^{-W^T X^{(i)}}} - y^{(i)}\right)(x_n^i) \end{bmatrix}$$

Proof

$$J(W) = -\sum_{i=1}^{M}\left[y^{(i)}\log\left(\frac{1}{1+e^{-W^T X^{(i)}}}\right) + (1-y^{(i)})\log\left(1-\frac{1}{1+e^{-W^T X^{(i)}}}\right)\right]$$

Simplifying:

➔ $\log\left(\frac{1}{1+e^{-W^T X^{(i)}}}\right) = -\log\left(1+e^{-W^T X^{(i)}}\right)$

➔ $\log\left(1-\frac{1}{1+e^{-W^T X^{(i)}}}\right) = \log\left(\frac{1+e^{-W^T X^{(i)}}}{1+e^{-W^T X^{(i)}}} - \frac{1}{1+e^{-W^T X^{(i)}}}\right) = \log\left(\frac{e^{-W^T X^{(i)}}}{1+e^{-W^T X^{(i)}}}\right)$

$$= \log\left(e^{-W^T X^{(i)}}\right) - \log\left(1+e^{-W^T X^{(i)}}\right) = -W^T X^{(i)} - \log\left(1+e^{-W^T X^{(i)}}\right)$$

Replacing

$$J(W) = -\sum_{i=1}^{M}\left[-y^{(i)}\log\left(1+e^{-W^T X^{(i)}}\right) + \left(1-y^{(i)}\right)\left(-W^T X^{(i)} - \log\left(1+e^{-W^T X^{(i)}}\right)\right)\right]$$

$$= -\sum_{i=1}^{m}\left[\cancel{-y^{(i)}\log\left(1+e^{-W^T X^{(i)}}\right)} - W^T X^{(i)} - \log\left(1+e^{-W^T X^{(i)}}\right) + y^{(i)} W^T X^{(i)} + \cancel{y^{(i)}\log\left(1+e^{-W^T X^{(i)}}\right)}\right]$$

$$= -\sum_{i=1}^{m}\left[-W^T X^{(i)} - \log\left(1+e^{-W^T X^{(i)}}\right) + y^{(i)} W^T X^{(i)}\right]$$

Applying the log properties

$$-W^T X^{(i)} - \log\left(1+e^{-W^T X^{(i)}}\right) = -\left[W^T X^{(i)} + \log\left(1+e^{-W^T X^{(i)}}\right)\right] = -\left[\log\left(e^{W^T X^{(i)}}\right) + \log\left(1+e^{-W^T X^{(i)}}\right)\right]$$

$$= -\log\left[\left(e^{W^T X^{(i)}}\right)\left(1+e^{-W^T X^{(i)}}\right)\right] = -\log\left[e^{W^T X^{(i)}} + 1\right]$$

Replacing

$$J(W) = -\sum_{i=1}^{m} \left[-\log\left[e^{W^T X^{(i)}} + 1\right] + y^{(i)} W^T X^{(i)} \right]$$

Deriving with respect to W_j

$$\frac{\partial J(W)}{\partial W_j} = -\sum_{i=1}^{m} \left[-\frac{X_j^{(i)} e^{W^T X^{(i)}}}{e^{W^T X^{(i)}} + 1} + y^{(i)} X_j^{(i)} \right] = \sum_{i=1}^{m} \left[\frac{X_j^{(i)} e^{W^T X^{(i)}}}{e^{W^T X^{(i)}} + 1} - y^{(i)} X_j^{(i)} \right]$$

$$\frac{e^{W^T X^{(i)}}}{e^{W^T X^{(i)}} + 1} = \frac{1}{\left(e^{-W^T X^{(i)}}\right)\left(e^{W^T X^{(i)}} + 1\right)} = \frac{1}{1 + e^{-W^T X^{(i)}}}$$

Thus

$$\frac{\partial J(W)}{\partial W_j} = \sum_{i=1}^{m} \left[\frac{X_j^{(i)}}{1 + e^{-W^T X^{(i)}}} - y^{(i)} X_j^{(i)} \right] = \sum_{i=1}^{m} \left[\left(\frac{1}{1 + e^{-W^T X^{(i)}}} - y^{(i)} \right) \left(X_j^{(i)} \right) \right]$$

3.2 Multiclass classification: one vs all

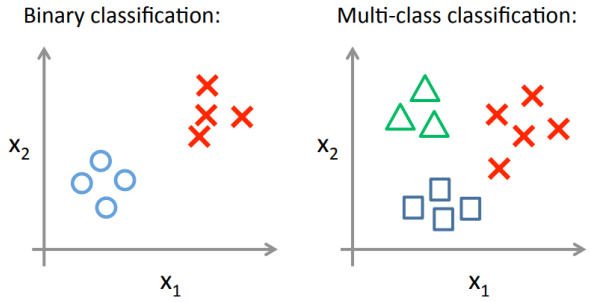
Sometimes, a binary classification is not enough.

Examples of binary classification:

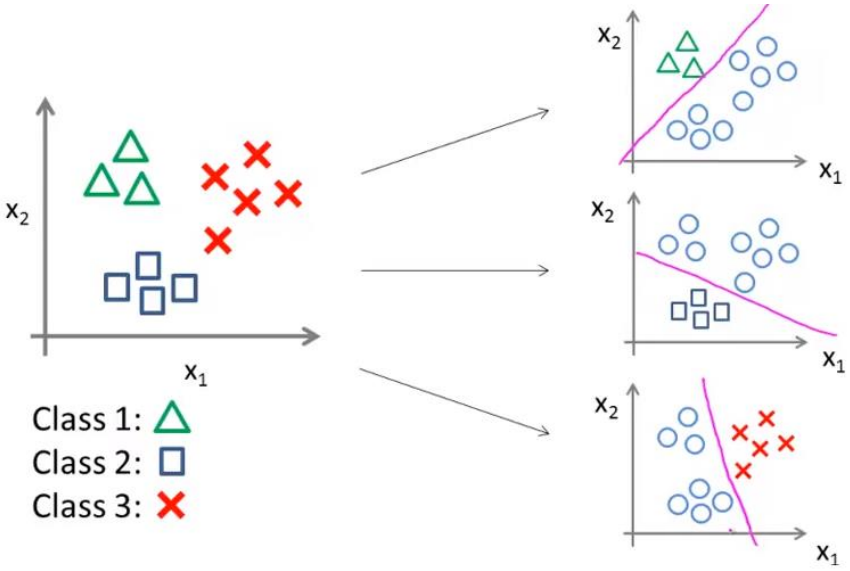
- Identify SPAM email

Examples of multiclass classification:

- Weather: sunny, cloudy, rainy, snowy.
- Face expression recognition: happy, sad, surprise, angry.



Classification one vs all:

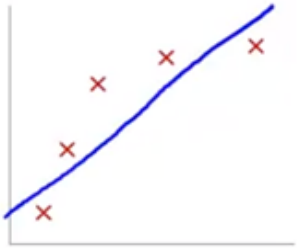


One classifier is created by class. To create the class i classifier, the data of that class are marked as 1 and the rest with 0. To classify a new data, this is evaluated in all the classifiers and the label assigned is the one who corresponds with the classifier where the value gets the highest probability.

3.3 Overfitting

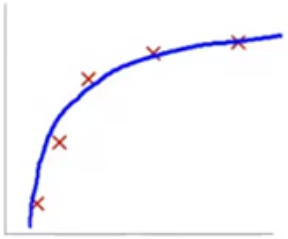
The overfitting in a supervised learning algorithm is how much the model "learns" or "fits" the training data.

Underfitting



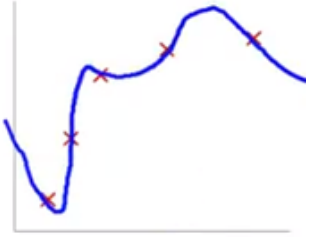
$\theta_0 + \theta_1 x$

Just right



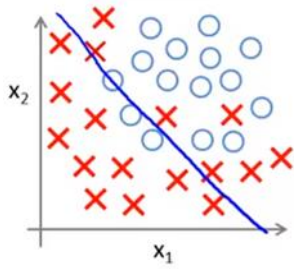
$\theta_0 + \theta_1 x + \theta_2 x^2$

Overfitting



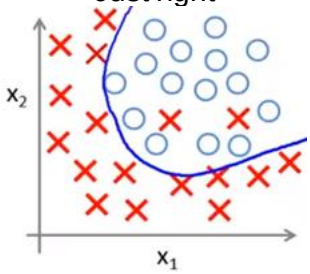
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Underfitting



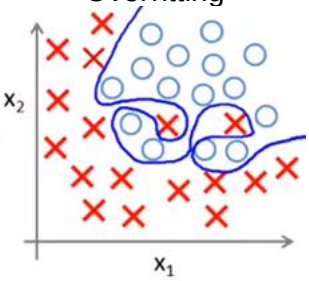
$\theta_0 + \theta_1 x_1 + \theta_2 x_2$

Just right



$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2$

Overfitting

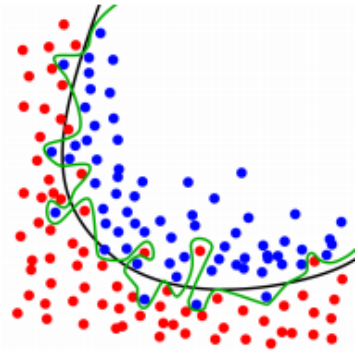


$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \theta_6 x_1^3 + \theta_7 x_1^3 x_2 + \cdots$

It is a clear tendency:

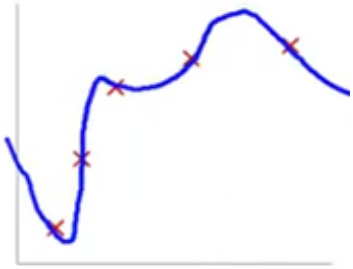
- If the model is too simple -> there is an underfitting
- If the model is too complex -> there is an overfitting

The difference between the correct model and the overfitted model is very clear:

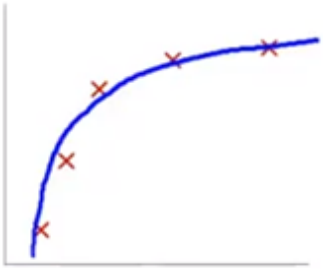


How to solve the overfitting?

From a complex model to a simpler one:



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



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

We can penalize some features in the optimization problem to simplify the model:

$$\min_{\boldsymbol{\theta}} J(\theta) = \frac{1}{2}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) + 1000\theta_3 + 1000\theta_4$$

And then, some of them will have small values, or even some of them will be 0.

However, it is difficult (or impossible) to know which parameters need to be 0. For that reason, the optimization problem is rewritten as:

$$\min_{\boldsymbol{\theta}} J(\theta) = \frac{1}{2}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right) + \lambda \sum_{j=1}^{n} \theta_j^2$$

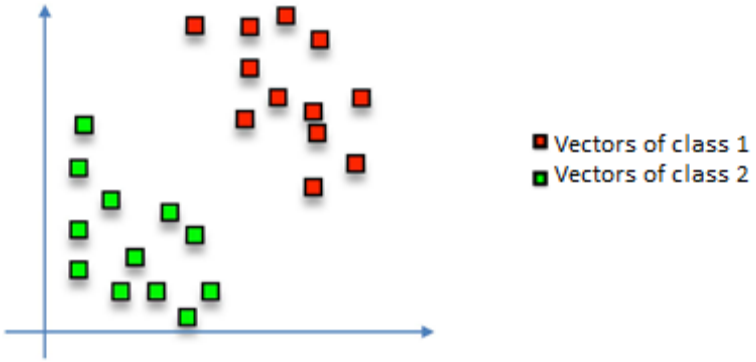
In this case, λ is and hyperparameter that defines the importance of having a simple model. If $\lambda = 0$ then we have a model that can be overfitting, if $\lambda \gg 0$ then the most important in the optimization problem is to have a simple model.

3.4 Support Vector Machines (SVM)

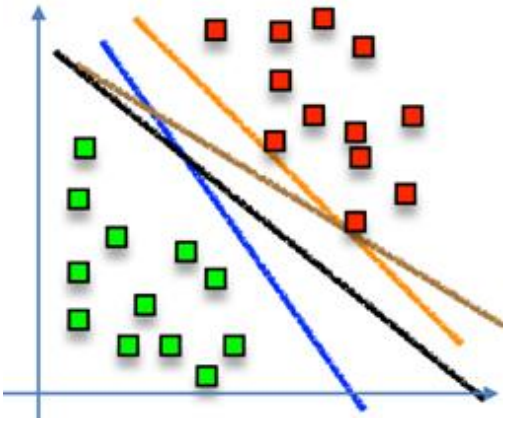
Supervised learning: Classification and regression
Input variables: continuous
Output variable: continuous (regression), discrete (classification)

SVM for classification

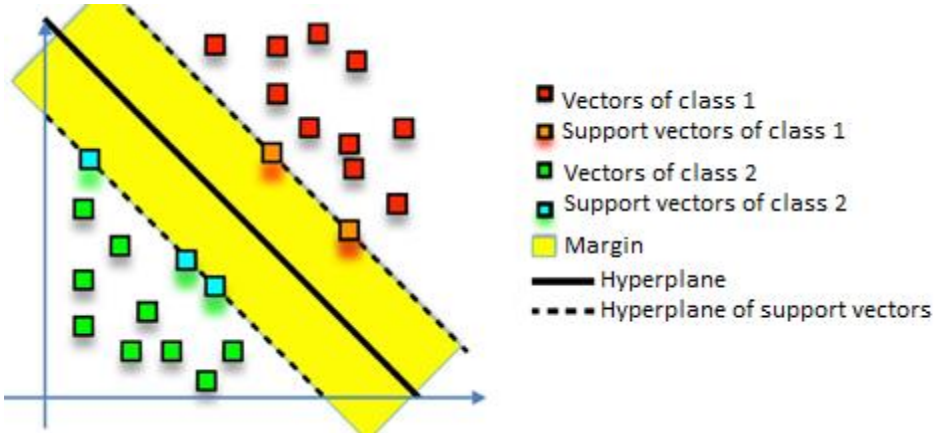
SVM for binary classification is a model that, like linear regression, identifies a line (2d), a plane (3d) or a hyperplane that divide the data into two classes. Its name is because the data are considered as vectors.



There may be many models (lines) that separate the classes' vectors, but, ¿Which one is the best?



The SVM calculates a linear classification model maximizing the margin between the classes.



As logistic regression, in SVM for binary classification, the model is defined as:

$$h(X) = w_0 + w_1x_1 + w_2x_2 + w_3x_2$$

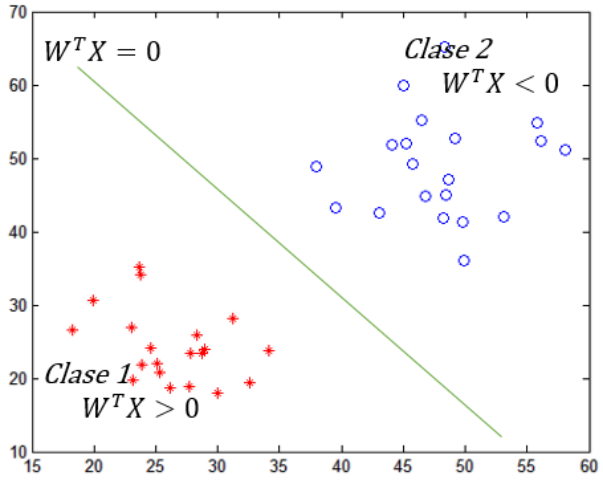
Defining

$W = [w_0, w_1, \dots, w_n]^T$

$X = [1, x_1, \dots, x_n]^T$

$$h(X) = W^T X$$

$$W^T X > 0 \rightarrow \ Class\ 1$$
$$W^T X < 0 \rightarrow \ Class\ 2$$



Optimization

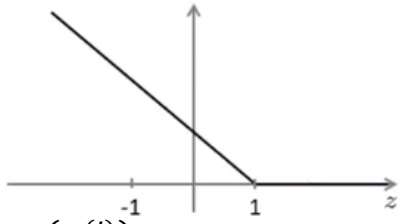
The original modeling of the problem is defined as the following optimization problem:

$$\min_{\theta} J(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1(\text{W}^T\text{X}^{(i)}) + (1-y^{(i)})\text{cost}_2(\text{W}^T\text{X}^{(i)})\right] + \sum_{i=1}^{n}\text{W}_j^2$$

$If\ y = 1$

We want $\text{W}^T\text{X} \geq 1$, not only positive value

$$Cost_1\left(h\left(x^{(i)}\right), y^{(i)}\right)$$

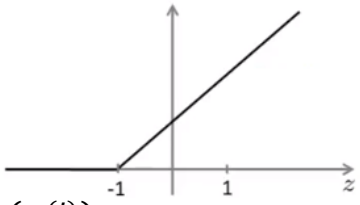


If $h\left(x^{(i)}\right) \geq 1$ then $cost = 0$

$If\ y = 0$

We want $\text{W}^T\text{X} \leq -1$, not only a negative value

$$Cost_0\left(h\left(x^{(i)}\right), y^{(i)}\right)$$



If $h\left(x^{(i)}\right) \leq -1$ then $cost = 0$

Another model of the optimization problem (sci-kit learn implementation) is defined as the following equation. ς_i represents the distance among each sample and the boundary.

$$\min_{w,\varsigma}\frac{1}{2}\text{W}^T\text{W} + C\sum_{i=1}^{n}\varsigma_i$$

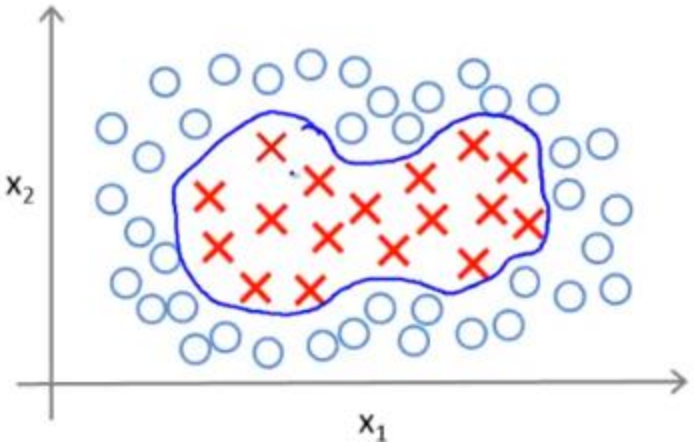
$$subject\ to$$

$$y^{(i)}\left(\text{W}^T\text{X}^{(i)}\right) \geq 1 - \varsigma_i$$

$$\varsigma_i \geq 0, i = 1, \dots, n$$

Kernels

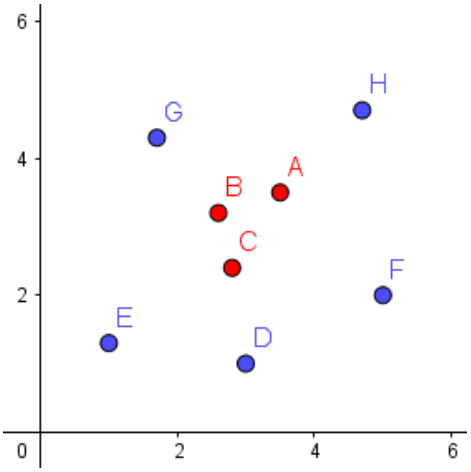
Sometimes a linear model is not enough, like in the next esample:



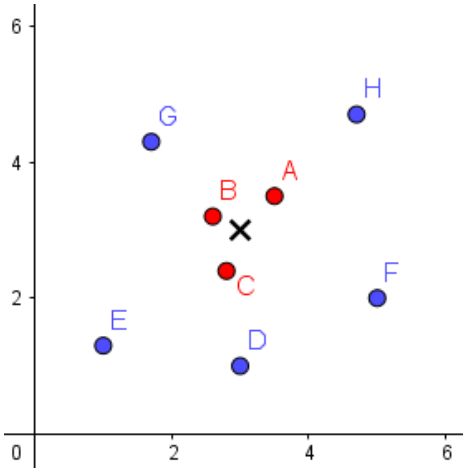
The use of **kernels** allows to transform the data from the **original vectorial space** to **another vectorial space** (the kernel space) where there can exist and hyperplane that separates the classes.

Intuition

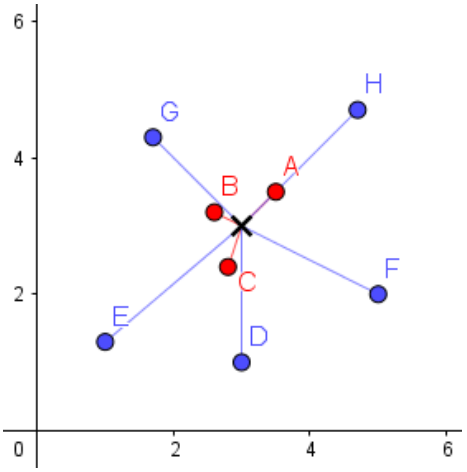
Original data space



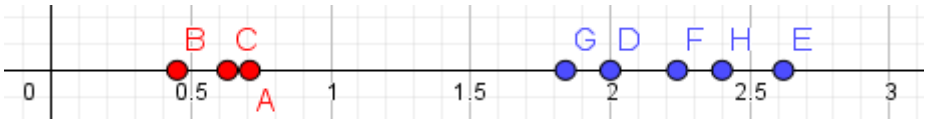
A landmark is added. It needs to be in the data vectorial space.



The distance between the data points and the landmark is measured. Euclidean distance can be used.



Using only those distances, we can transform the points' data from 2d to 1d. In this space, the classes are linearly separable.



In fact, SVM with kernels uses all the points as landmarks. In this case the vectorial space is transform from nD to mD.

Example:

Data in the original space (4D):

	X1	X2	X3	X4
Sample 1	3	1	2	4
Sample 2	5	1	0	9
Sample 3	1	3	6	2
Sample 4	9	2	6	3
Sample 5	0	7	3	5
Sample 6	1	3	2	8

Data in the new space(6D): (one feature for each sample)

	L1 ([3,1,2,4])	L2 ([5,1,0,9])	L3 ([1,3,6,2])	L4 ([9,2,6,3])	L5 ([0,7,3,5])	L6 ([1,3,2,8])
1	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}3\\1\\2\\4\end{bmatrix}\right)$
2	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}5\\1\\0\\9\end{bmatrix}\right)$
3	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}1\\3\\6\\2\end{bmatrix}\right)$
4	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}9\\2\\6\\3\end{bmatrix}\right)$
5	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}0\\7\\3\\5\end{bmatrix}\right)$
6	$Sim\left(\begin{bmatrix}3\\1\\2\\4\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}5\\1\\0\\9\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\6\\2\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}9\\2\\6\\3\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}0\\7\\3\\5\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$	$Sim\left(\begin{bmatrix}1\\3\\2\\8\end{bmatrix},\begin{bmatrix}1\\3\\2\\8\end{bmatrix}\right)$

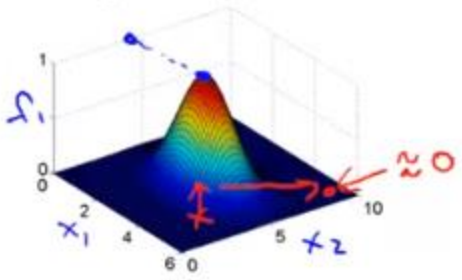
The most popular kernel is the Gaussian kernel. The similarity function is defined as follows:

$$f = similitud(x,l) = \exp\left(-\frac{\left||x-l|\right|^2}{2\sigma^2}\right)$$

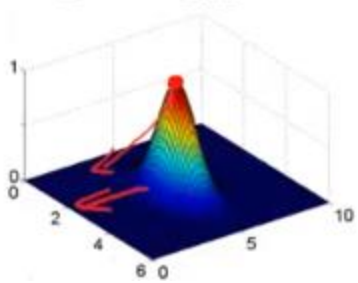
Where l represents the landmark, $\left|\left|.\right|\right|^2$ the squared norm. Remembering $\left||U|\right| = \sqrt{u_1^2+u_2^2+\cdots+u_n^2}$.

σ can change the Gaussian's influence.

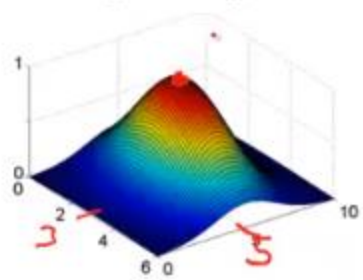
$\sigma^2 = 1$



$\sigma^2 = 0.5$



$\sigma^2 = 3$

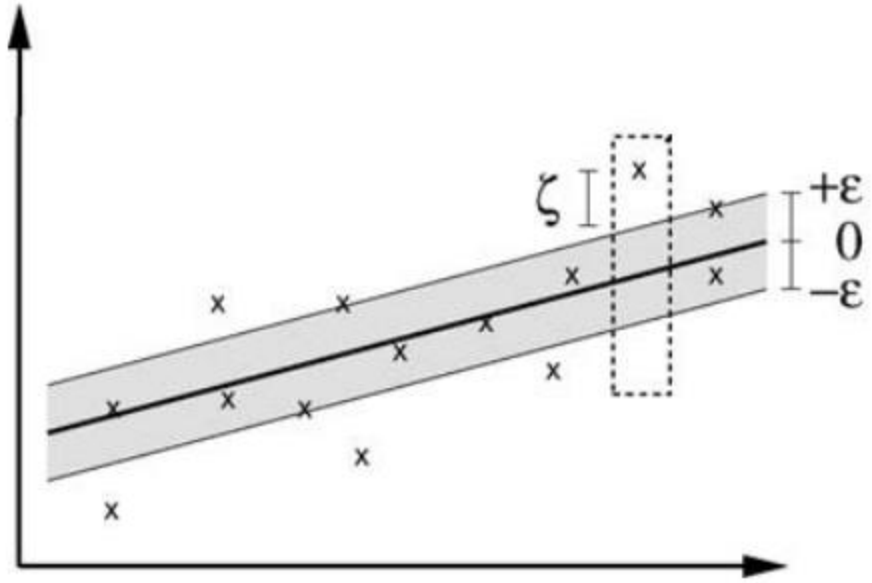


SVM for regression

Smola and Schölkopf (2004) describe how can be used SVM specifically to regression problems. [Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and computing, 14(3), 199-222.]

Given N input vectors $x_1, x_2, \ldots, x_N$ in a d-dimensional space $x_i \in \mathbb{R}^d$, and N output values $y_1, y_2, \ldots, y_N$ where $y_i \in \mathbb{R}$, it is wanted to find the function f that minimizes the difference between $f(x_i)$ and y_i. The model proposed by SVM to solve the previous problem is $f(x_i) = \langle w, x_i \rangle + b$, where $\langle u, v \rangle$ represents the dot product between the vectors u and v, and the vector w, and the scalar value b are the parameters of a linear model.

To find the values of the vector w and the scalar b, SVM solves an optimization problem that constructs a margin that wraps the linear model. SVM tries to minimize the distance between the margin and the value $f(x_i)$ for those input-output training data (x_i, y_i) where $|f(x_i) - y_i| \geq \epsilon$. It means all the pair of input-output training where $|f(x_i) - y_i| < \epsilon$ do not contribute to the optimization function.



The optimization problem is defined as:

$$\min_{w,b,\varsigma,\varsigma^*} \frac{1}{2}\langle w, w \rangle + C \sum_{i=1}^{n} (\varsigma_i + \varsigma_i^*)$$

$$subject\ to$$

$$y_i - f(x_i) \leq \epsilon + \varsigma_i$$

$$f(x_i) - y_i \leq \epsilon + \varsigma_i^*$$

$$\varsigma_i, \varsigma_i^* \geq 0, i = 1, \ldots, n$$

The optimization problem is divided in two terms: the first one $\frac{1}{2}\langle w, w\rangle$ prevents the overfitting and the second $\sum_{i=1}^{n}(\varsigma_i + \varsigma_i^*)$ minimizes the distances between the margin and $f(x_i)$. The value of C gives the relevance to each term (by default $= 1.0$).

One of the principal characteristics of SVM is the use of kernels, which can work in a different vectorial space to construct a non linear model in the original space. The idea is preprocess the data mapping the input variables $x_i \in \mathbb{R}^d$ with new vectors $x_i' \in \mathbb{R}^N$. The vectors x_i' are defined as $x_i' = [K(x_i, x_1), K(x_i, x_2), \dots, K(x_i, x_N)]$. The most common kernel function $K(.)$ is the Gaussian kernel and it is defined in the equation 7.

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

where $\|.\|$ is the norm of a vector and γ is a hiperparameter that is optimized.