POLYTECH ENERGY SCHOOL
OIL&GAS IN ENERGY INDUSTRY

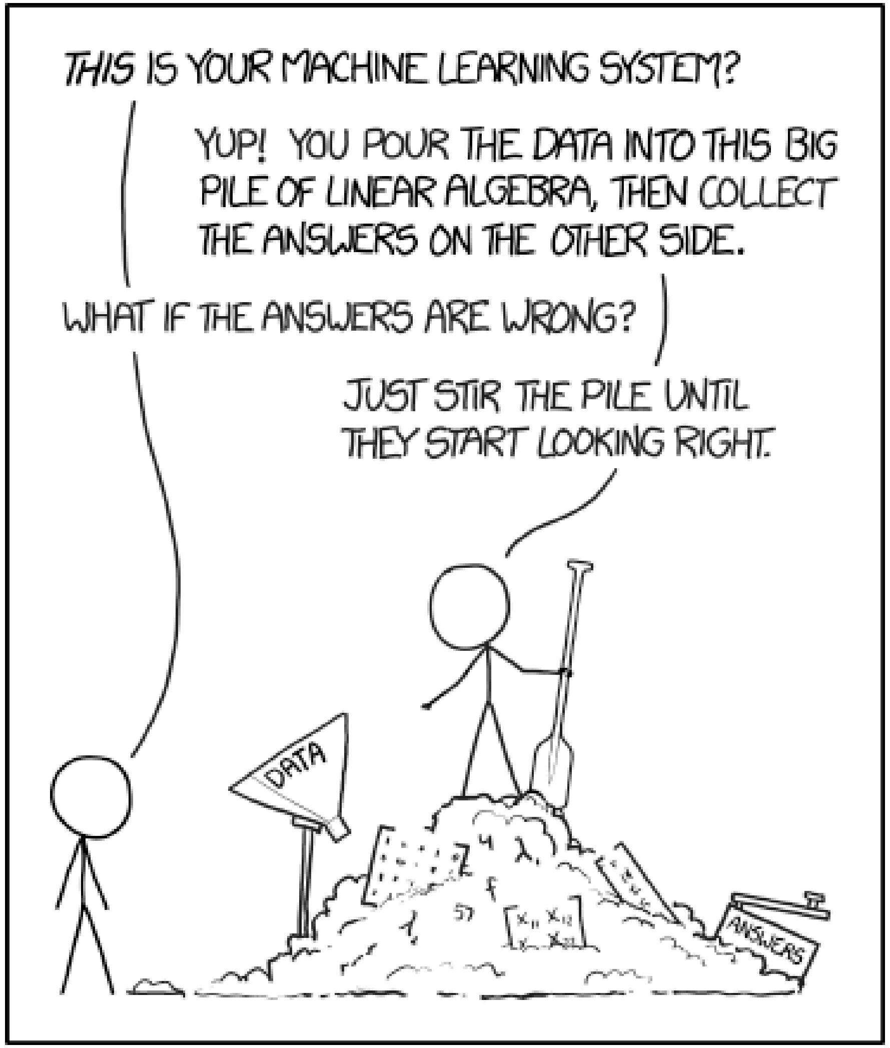
POLYTECH
Peter the Great
St.Petersburg Polytechnic
University

Institute of Applied
Mathematics and Mechanics

Machine learning in Oil&Gas Industry

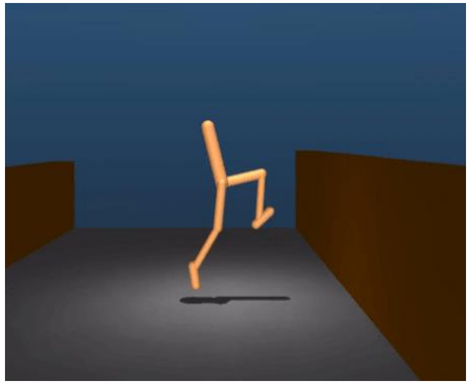
Andrey Murachev

What is Machine learning?

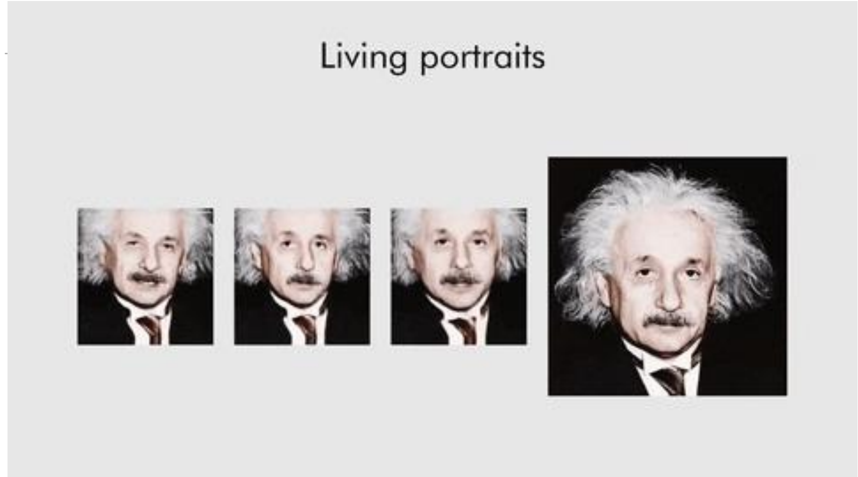


Amazing machine learning

Movement in difficult environments

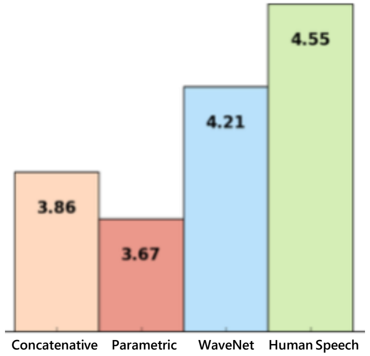


Deep fake. Human image synthesis

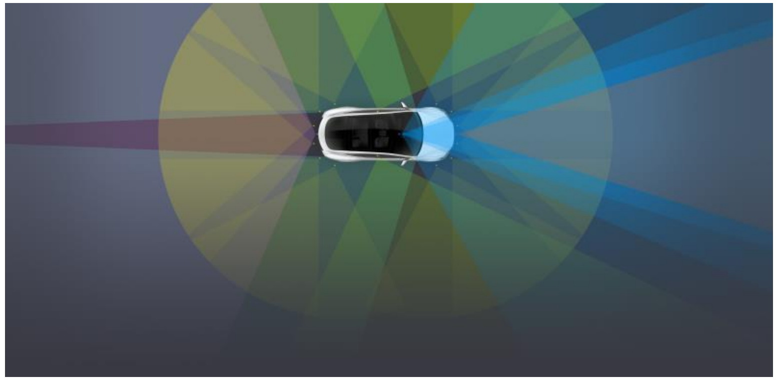


Significantly increasing the quality of machine translation

Imitation of human speech

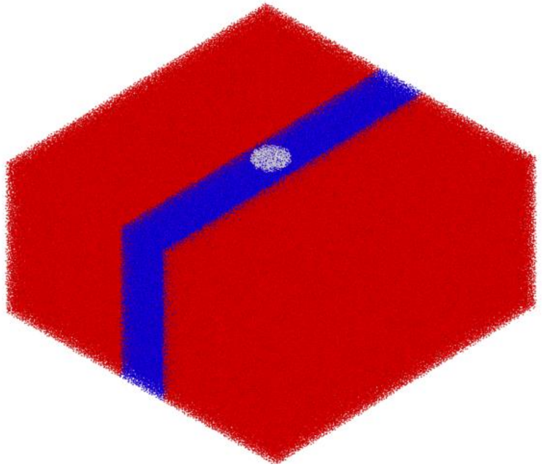


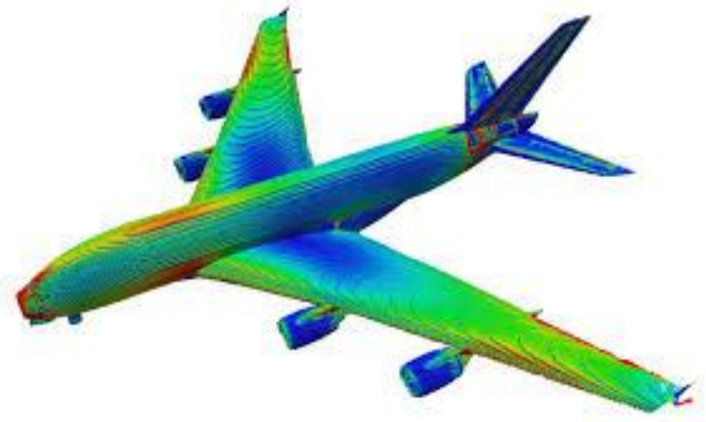
Autopilot

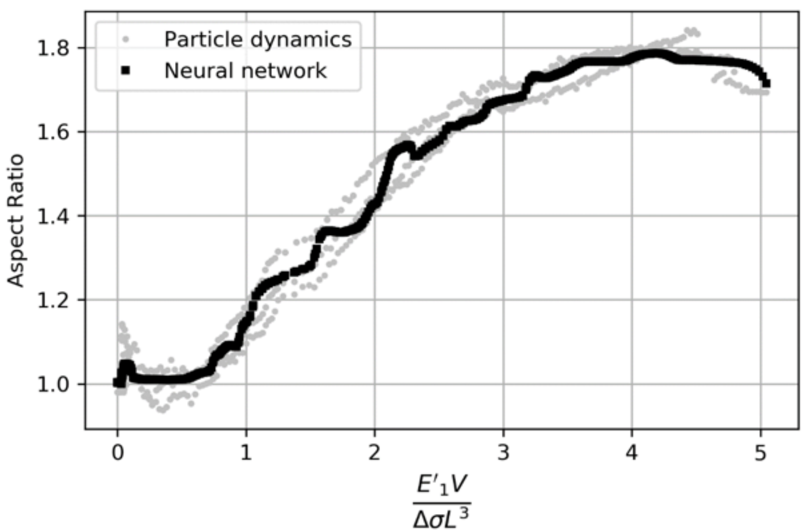


ML in science

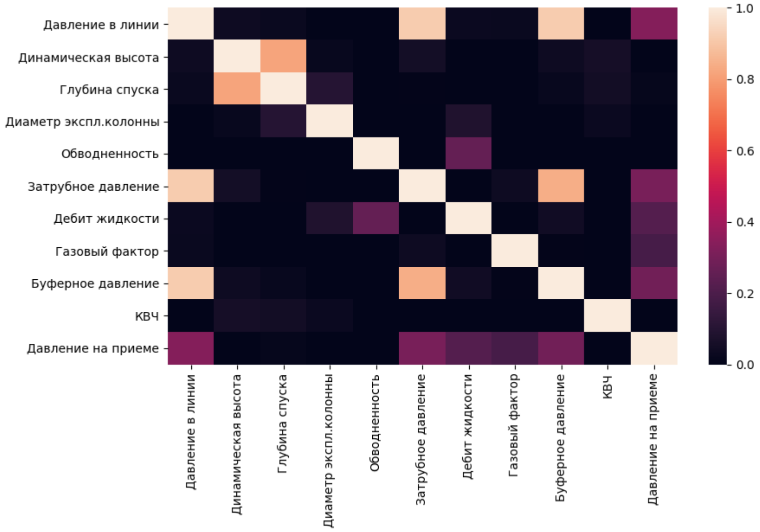
Surrogate modelling approach for optimization complex calculations



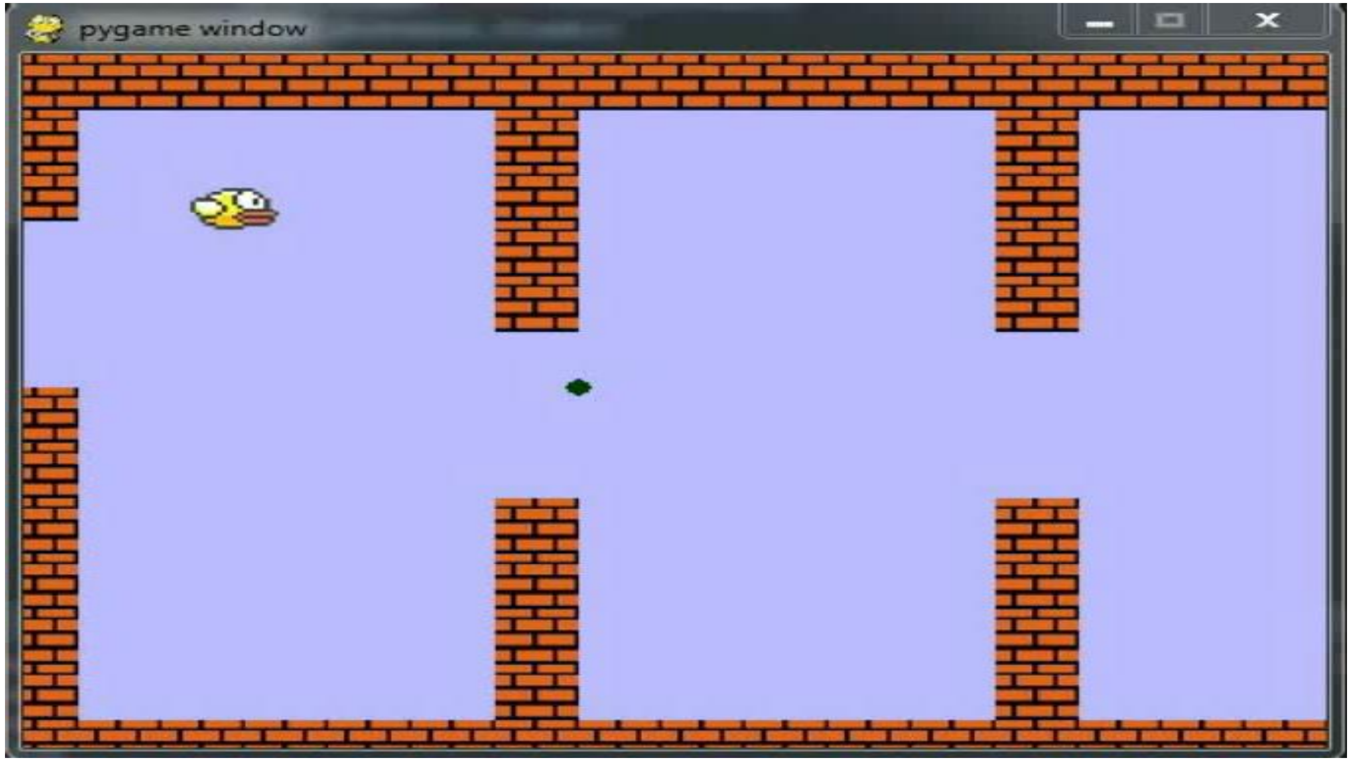




Oil well pressure prediction

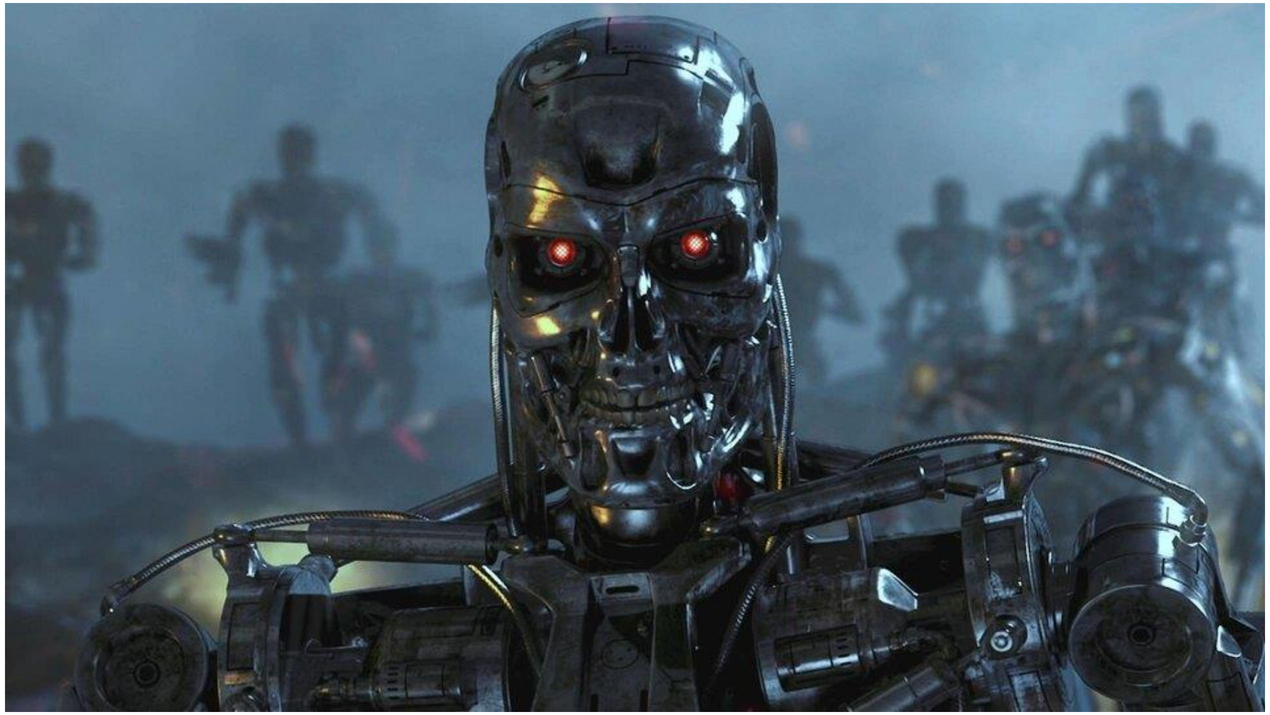


Floppy bird with ML



Machine learning

- *A computer program is said to learn from experience ‘E’, with respect to some class of tasks ‘T’ and performance measure ‘P’ if its performance at tasks in ‘T’ as measured by ‘P’ improves with experience ‘E’.*
- *The extraction of learning from data*
- *Put simply, if a* ***computer program improves itself with experience then we can say that it has learned*****.**



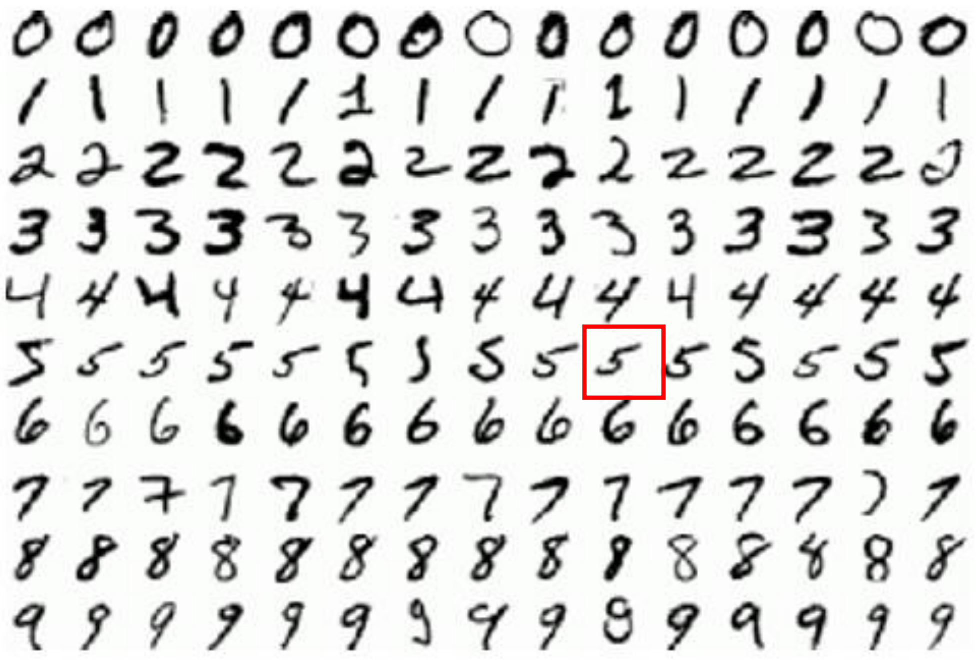
Learning is based on precedents

X is the set of objects

Y is the set of answers

$\boldsymbol{g}: X \to Y$ is unknow function (target function)

(X, Y) is dataset



Data

$\{x_1, x_2, \dots, x_n\} \subset$ X are samples

$y_i = y(x_i), i = 1 \dots n$, are known answers

Goal

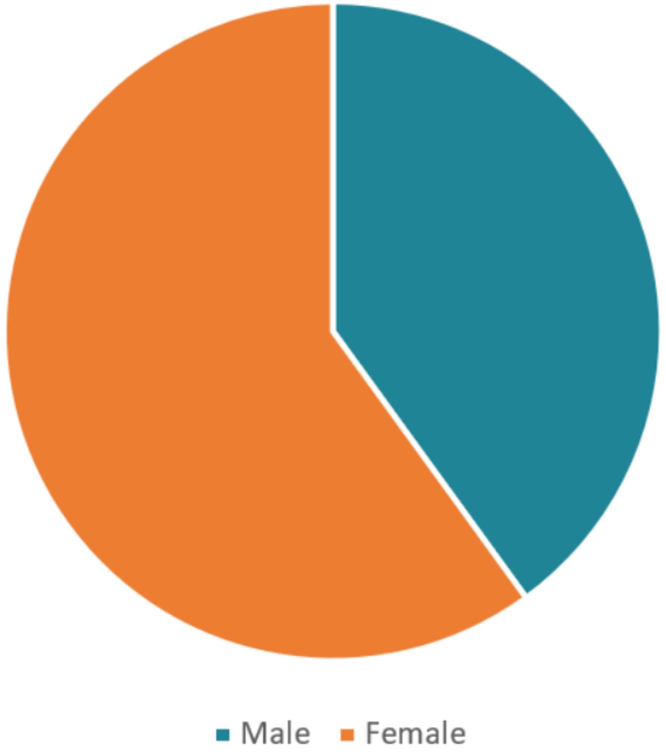
$\boldsymbol{a}$: $X \to Y$ is algorithm that approximates function g on dataset X, decision function

Learning is based on precedents

$f_j : X \to D_j,\ j = 1 \dots n$, is features of objects (numbers, result of measurement)

Types of features:

- $D_j = \{0,1\}$ is binary feature (true/false)
- D_j is categorical feature (race, pet breed, name)
- $D_j < \infty$ is ordinal feature. There is a clear ordering of the variables.
- $D_j = \mathbb{R}$ is Interval feature

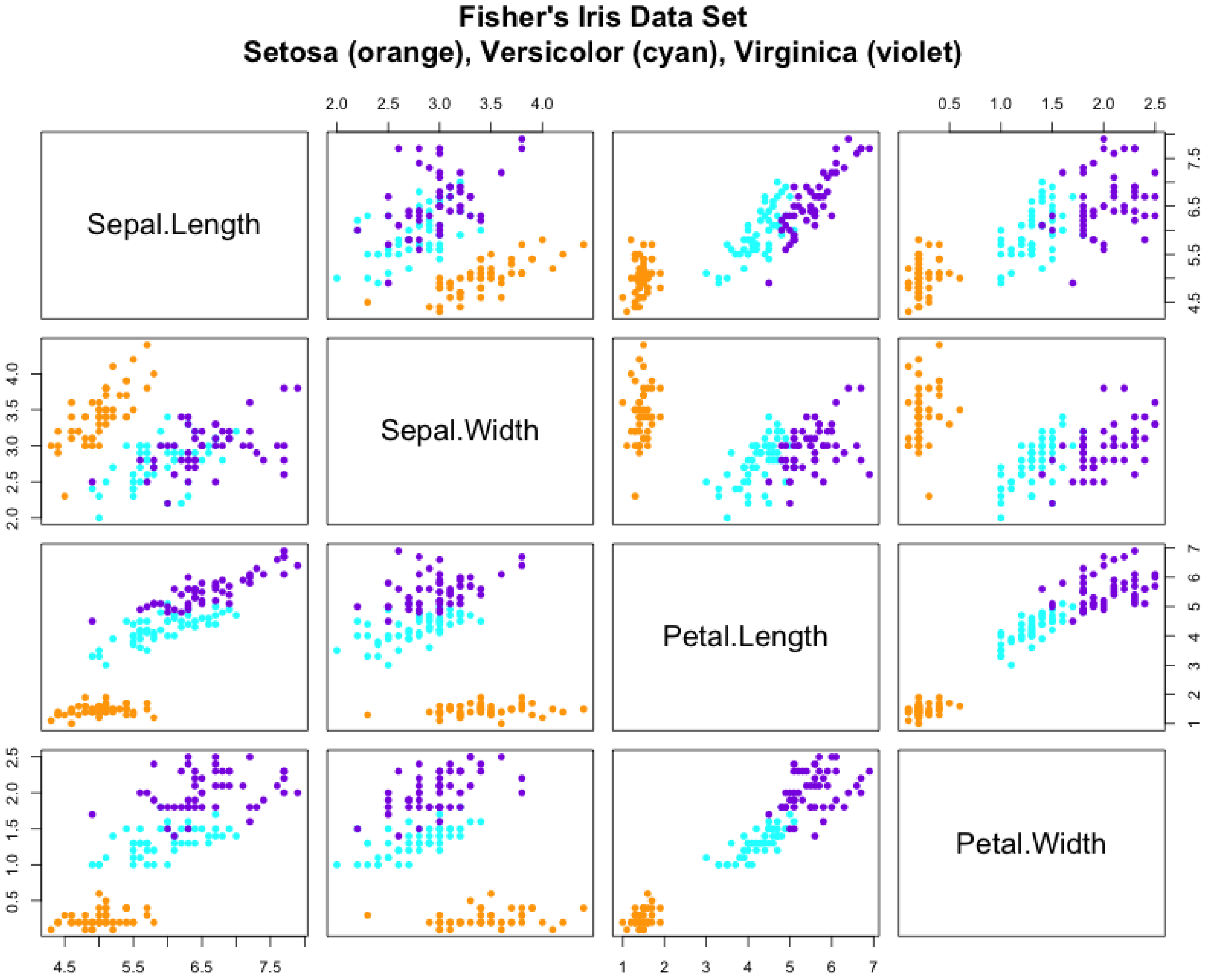


$\left(f_1(x), \dots, f_m(x)\right)$ – feature vector

$$F = \begin{pmatrix} f_1(x_1) & \dots & f_m(x_1) \\ \dots & \dots & \dots \\ f_1(x_n) & \dots & f_m(x_n) \end{pmatrix}$$ is feature matrix or Data frame

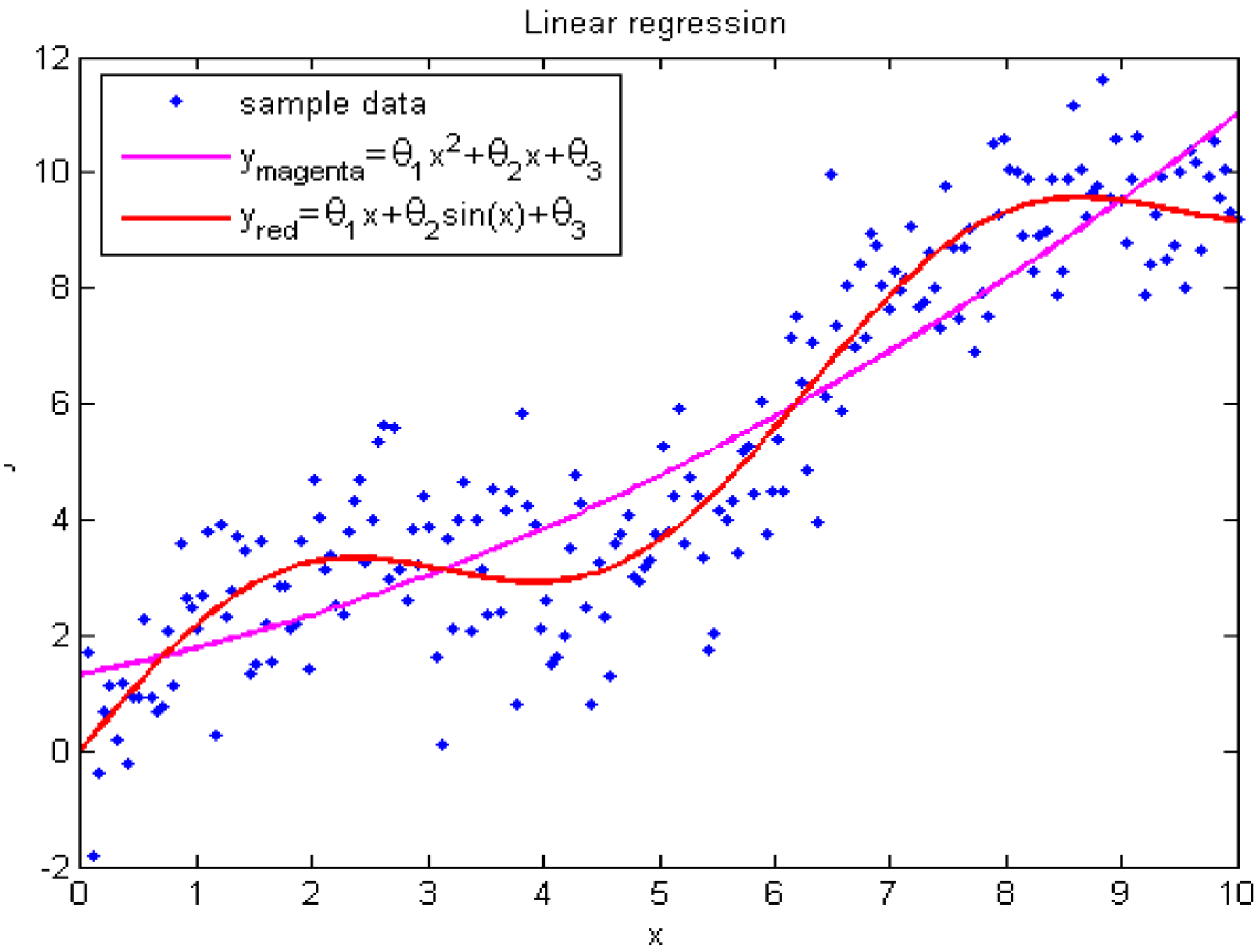
POLYTECH ENERGY SCHOOL
OIL&GAS IN ENERGY INDUSTRY

Iris classification



Regression

Different models produce different results



Learning steps

Train

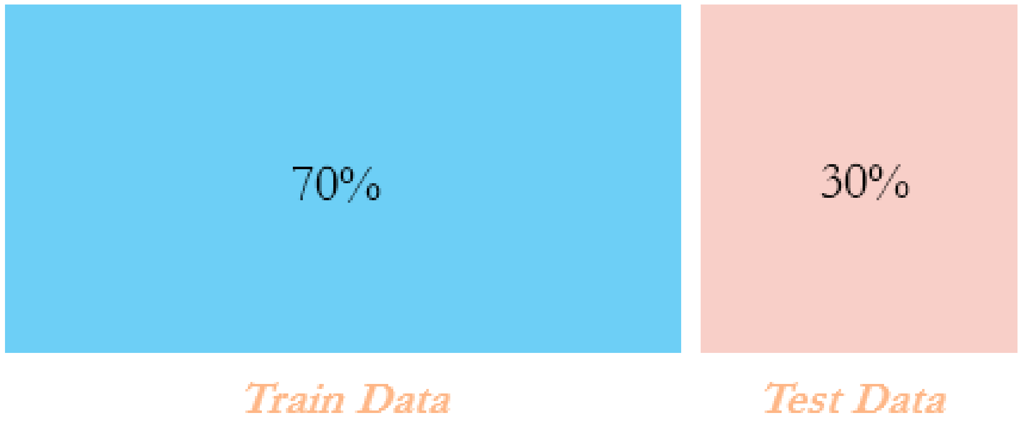
The learning algorithm builds the decision function based on the train data

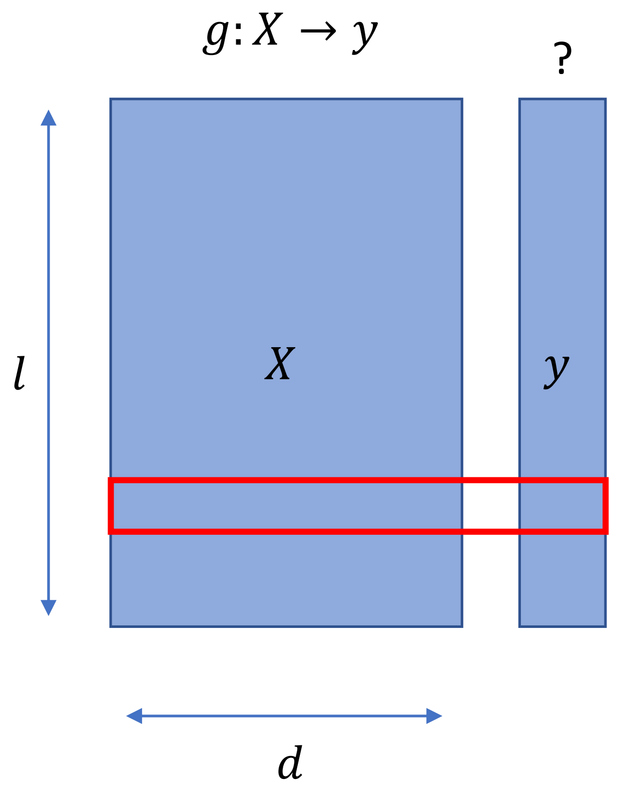
$$\boxed{\begin{pmatrix} f_1(x_1) & \dots & f_m(x_1) \\ \dots & \dots & \dots \\ f_1(x_n) & \dots & f_m(x_n) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}} \xrightarrow{\mu} a$$

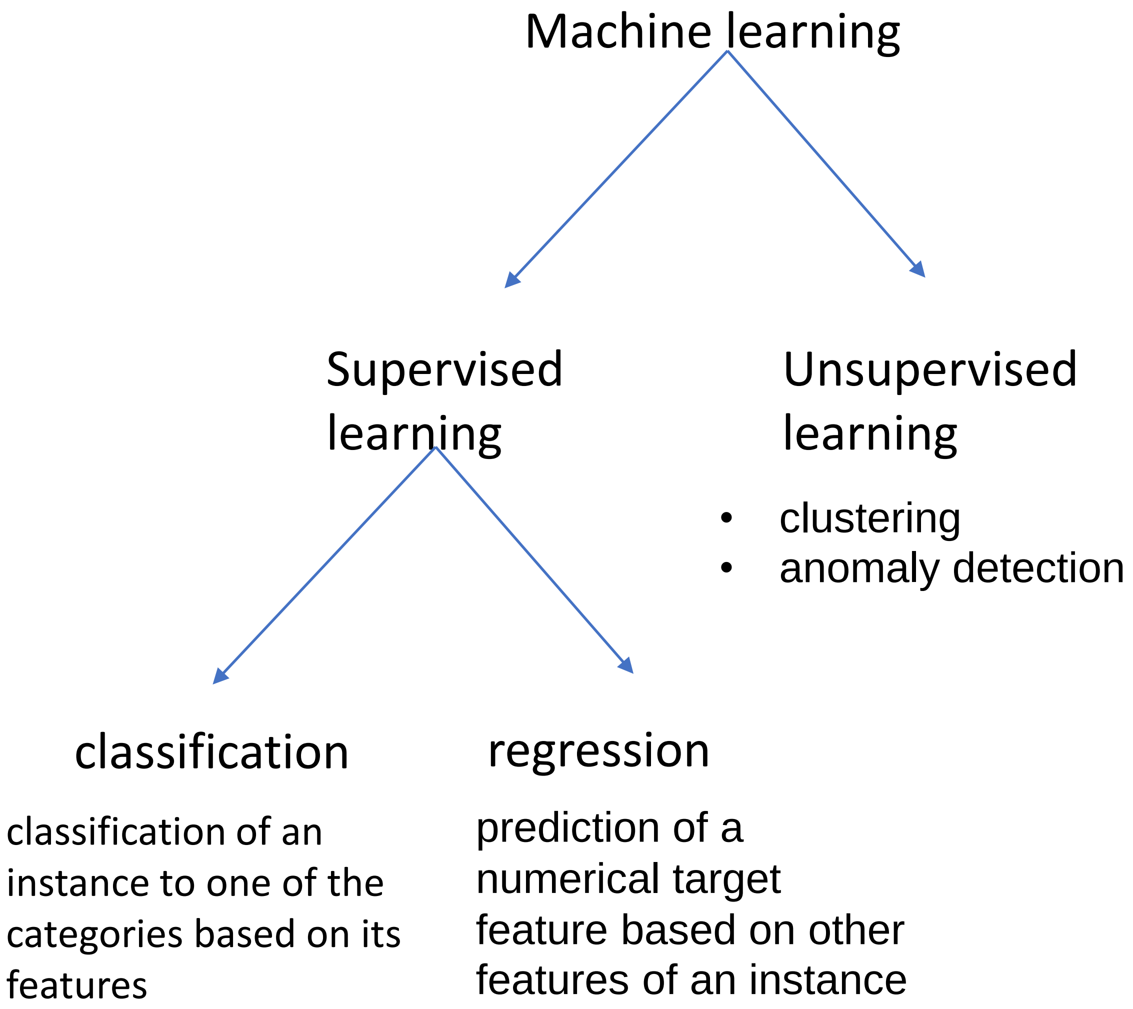
Test

The decision function is used on new data x' (test data)

$$\begin{pmatrix} f_1(x'_1) & \dots & f_m(x'_1) \\ \dots & \dots & \dots \\ f_1(x'_k) & \dots & f_m(x'_k) \end{pmatrix} \xrightarrow{a} \begin{pmatrix} a(x'_1) \\ \dots \\ a(x'_k) \end{pmatrix}$$





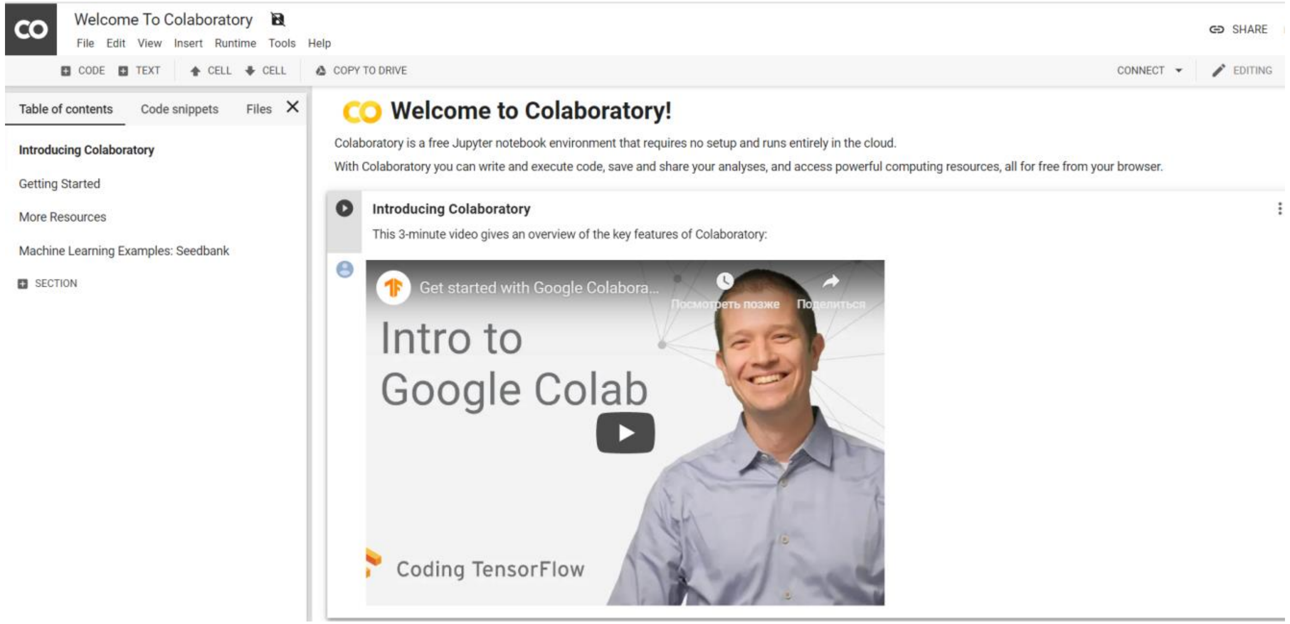


•**clustering** is identifying partitions of instances based on the features of these instances so that the members within the groups are more similar to each other than those in the other groups;
•**anomaly detection** is search for instances that are "greatly dissimilar" to the rest of the sample or to some group of instances

Programming

Google colaboratory

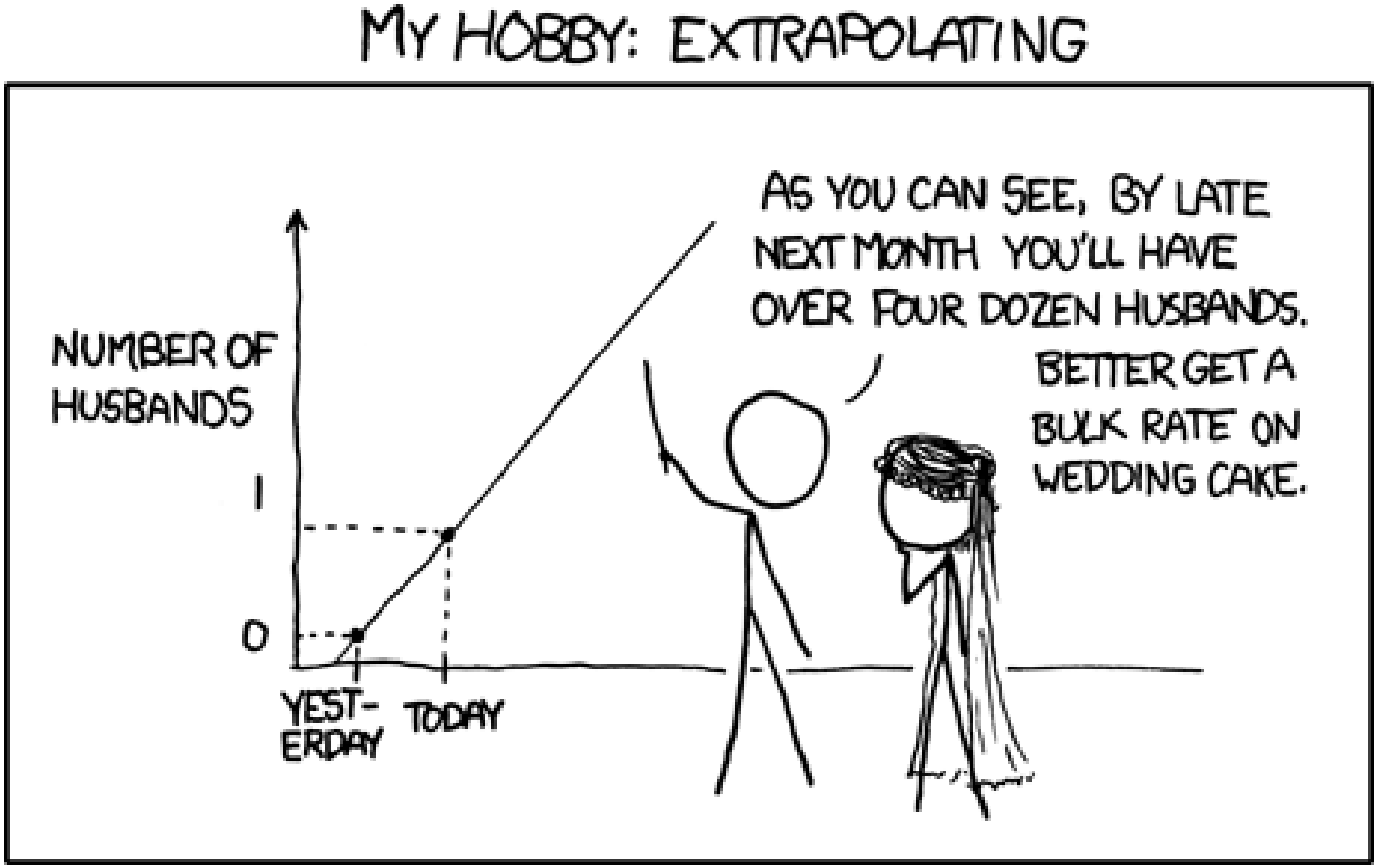
https://colab.research.google.com/



JetBrains PyCharm + Anaconda

ANACONDA

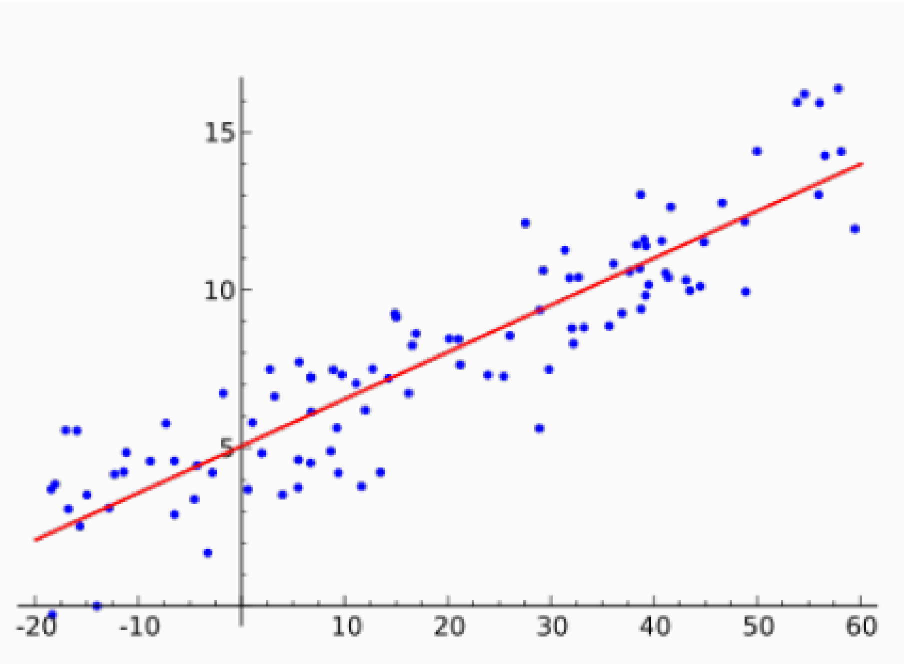
Linear regression



Model

- The liner regression model

$$y(x, w) = w_0 + \sum_{j=1}^{n} x^{(j)} w_j = x^T w, \qquad x = (1, x^{(1)}, \dots, x^{(n)})$$



- We want to predict y as function of vector x

$$h(x) = w_0 + \sum_{j=1}^{n} x^{(j)} w_j = x^T w$$

- $y \in R^n$ is the target variable;

$y^{(i)} = y\left(x^{(i)}\right),\ i = 1, \dots, m$

- $\mathbf{w}$ is the vector of the model parameters (in machine learning, these parameters are often referred to as weights);

- $\mathbf{x}$ is a matrix of features, x.size()=[m,n]

Liner regression

x_1	x_2	x_3	x_4	y
Area	**Number of rooms**	**Number of floors**	**Age**	**Price (в $1000)**
2104	5	1	45	460
1416	3	2	40	232
1534	3	3	30	315
852	2	1	36	178
…	…	…	…	…
…	…	…	…	…

$x^{(2)}$

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \qquad x_3^{(2)} = 2$$

Liner regression

$$\boxed{h_\theta(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4}$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \in \mathbb{R}^{n+1} \qquad W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \in \mathbb{R}^{n+1}$$

```
h=np.dot(X,theta.T)
print('X size=',X.shape)
print('w size=',theta.shape)
------------------------
# X size= (100, 5)
# w size= (5, 1)
```

$$XW^T = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [w_0 \;\; w_1 \;\; w_2 \;\; w_3 \;\; w_4] = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 = h_w(x)$$

$x_0 = 1$

$$X = \begin{bmatrix} 2104 & 5 & 1 & 45 \\ 1416 & 3 & 2 & 40 \\ 1534 & 3 & 3 & 30 \\ 852 & 2 & 1 & 36 \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 3 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

x_0

```
E = np.ones(len(X))
X = np.c_[E, X]
```

Ordinary Least Squares

- How to find the best parameters $\mathbf{w}$ for training data $(x_i, y_i)_{i=1}^{m}$?
- Ordinary Least Squares. Let's minimize

$$RSS(w) = \sum_{i=1}^{m} \left(y^{(i)} - x^{(i)T}\mathrm{w}\right)^2$$

- How can we do it?

Accurate solution

- To order obtain an accurate solution let's represent $RSS(\mathrm{w})$ in the following form

$$\mathrm{RSS}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}),$$

To solve this optimization problem, we need to calculate derivatives with respect to the model parameters. We set them to zero and solve the resulting equation for **w**

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

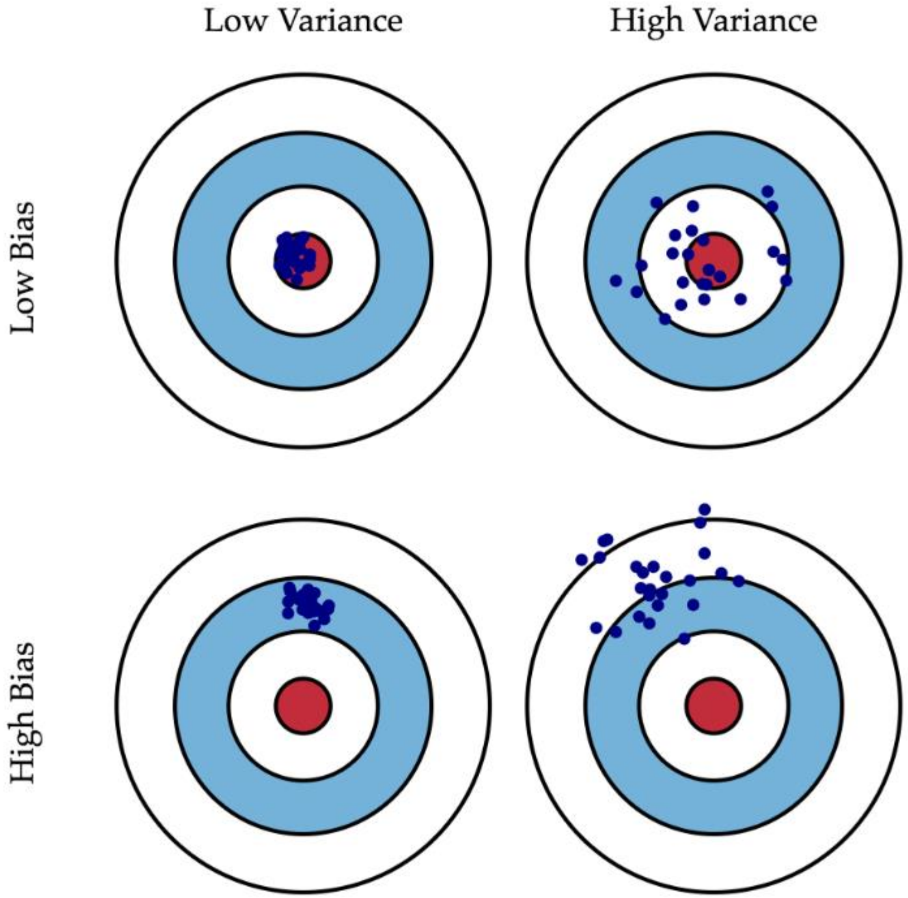
Notes:

- We do not need to scale data
- The solution is difficult to calculate when the number of features is very large $n \approx 10^6$
- The matrix $X^T X$ is irreversible when $m \leq n$
- The matrix $X^T X$ is irreversible when linearly dependent features exist

Bias-Variance Decomposition

$$Err(x) = Bias(h)^2 + Var(h) + \sigma^2$$

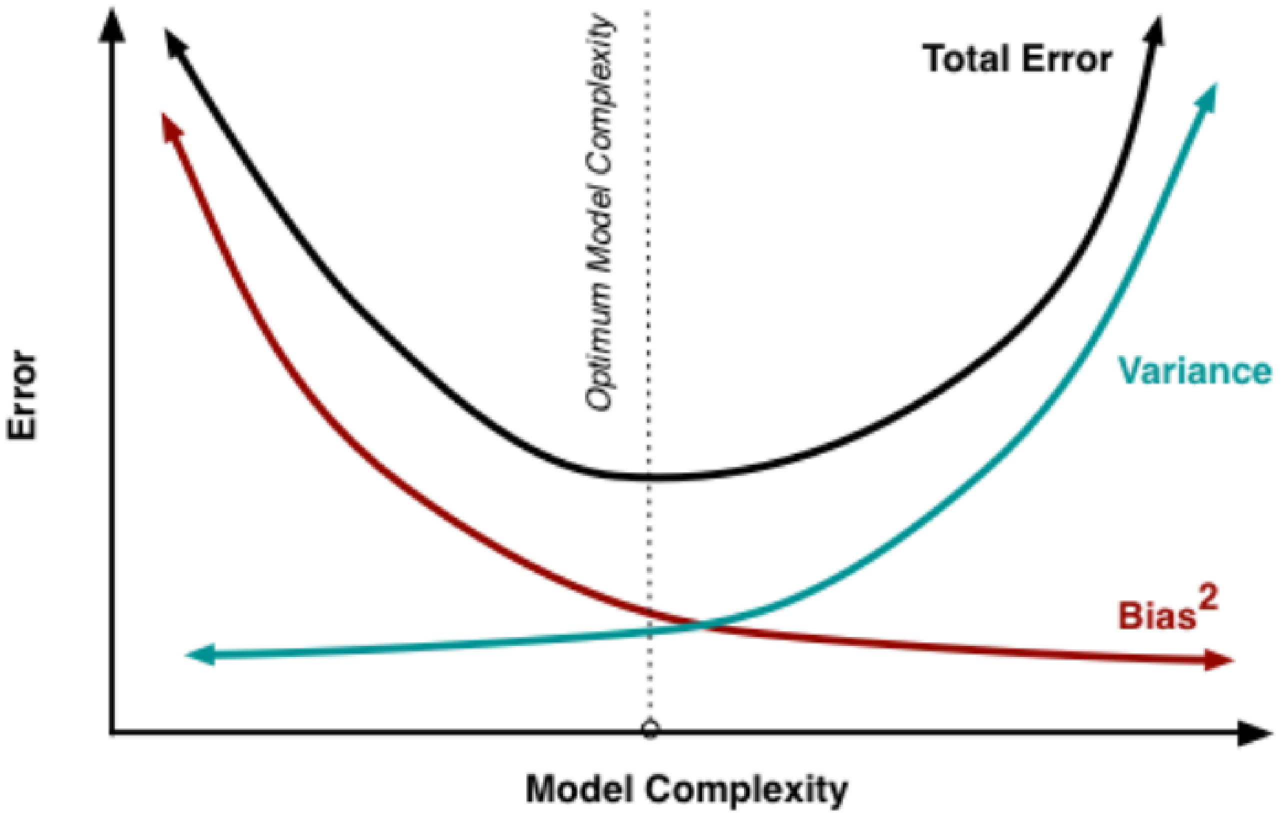
- $Bias(h)$ is the average error for all sets of data
- $Var(h)$ is error variability, or by how much error will vary if we train the model on different sets of data;
- irremovable error: σ^2



Bias-Variance Decomposition

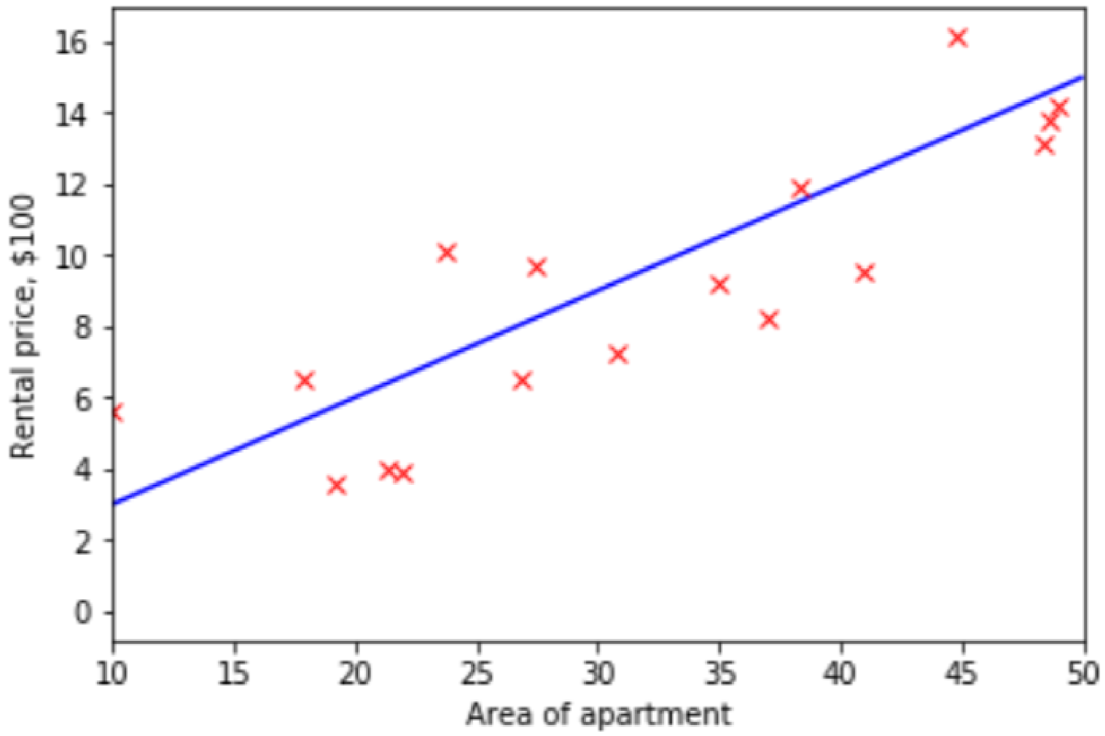
- Generally, **as the model** becomes **more computational** (e.g. when the number of free parameters grows), **the variance** of the estimate also **increases**, **but bias decreases**. The model learns well the training data, but it is incapable of generalization. Therefore, there are surprises in the test set.
- On the other side, **if the model is** too **simple**, it will not be able to learn the pattern.

- the OLS estimator of parameters of the linear model is the best for the class of linear unbiased estimator (The Gauss-Markov theorem).



Example. Cost function (MSE)

Area of apartment	Rental price
26.80	6.50
35.00	9.25
44.80	16.12
38.30	11.92
40.90	9.53
23.70	10.08
…	…



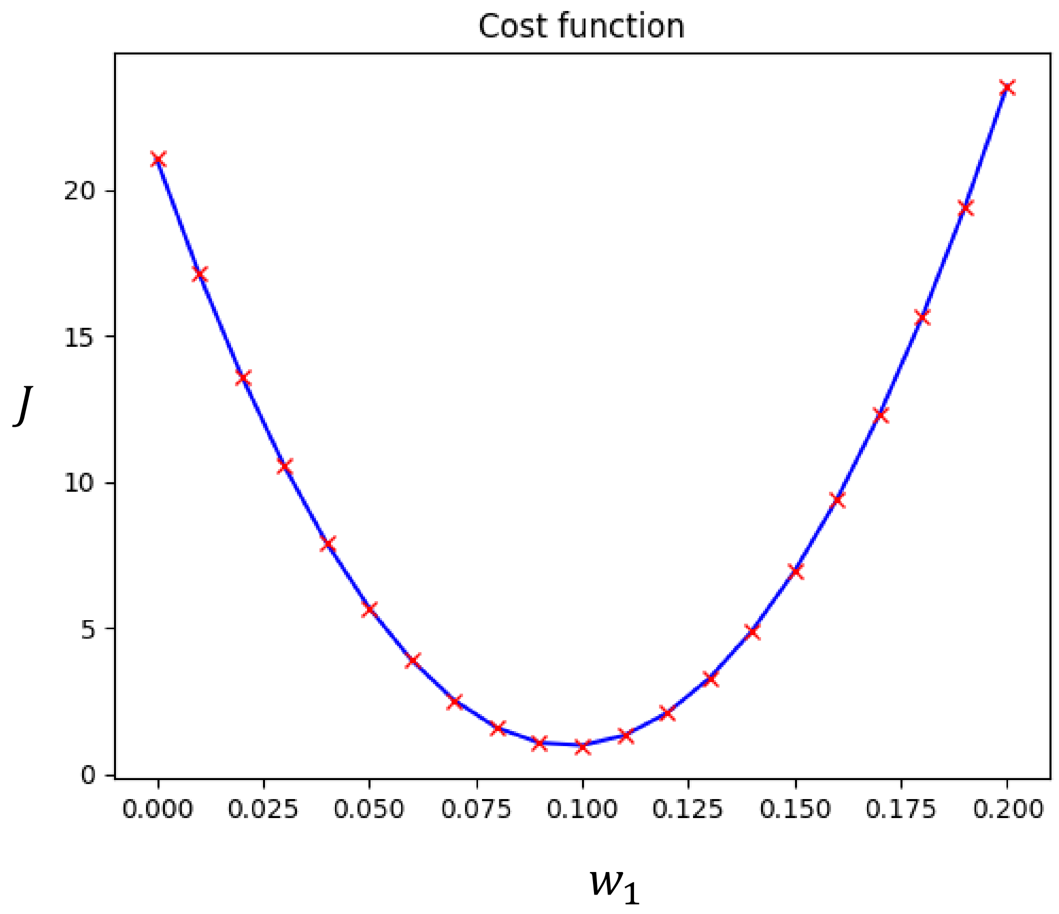
Hypothesis: $h_{\mathrm{w}} = w_0 + w_1 x$– one-dimensional linear regression

Git:

Cost function

$$J(w_0, w_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

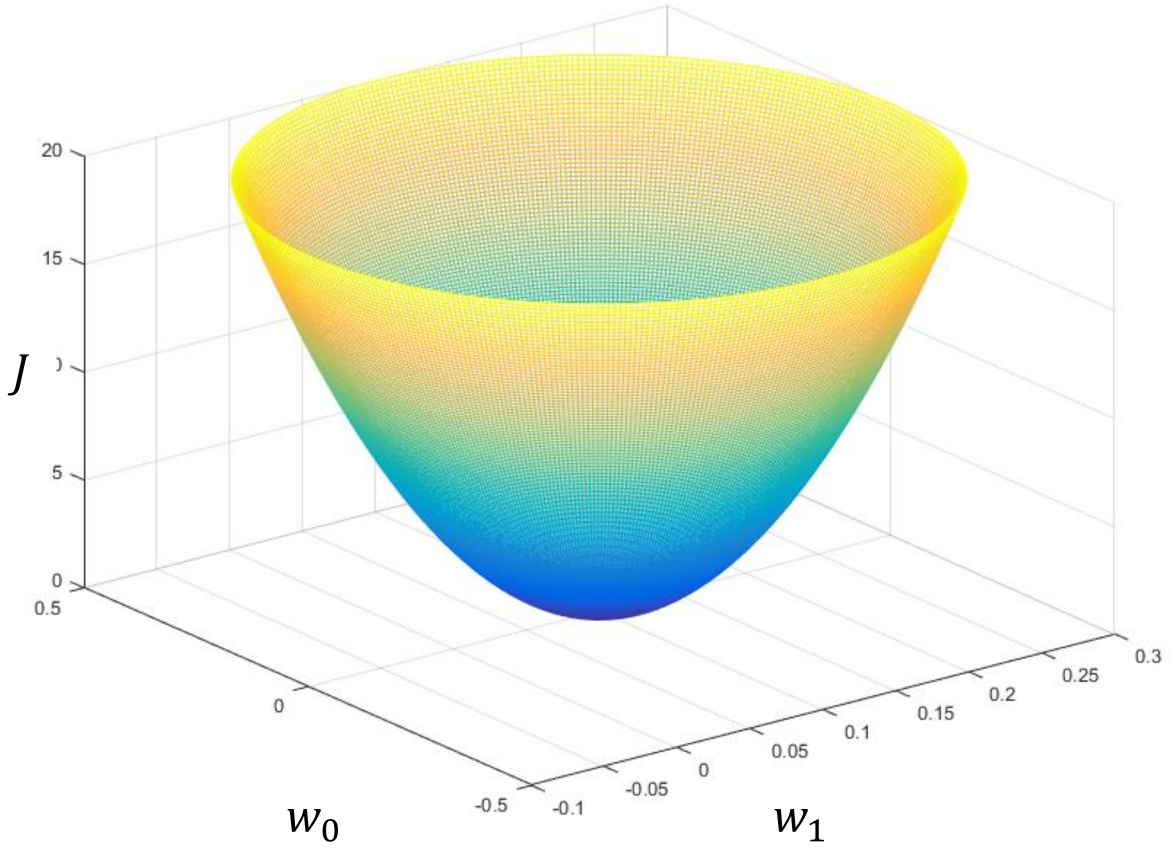
The learning task is the minimization $J(w_0, w_1)$.



Cost function

$$J(w_0, w_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

The learning task is the minimization $J(w_0, w_1)$.



Gradient descent

Let the function $J\ (w_0, w_1)$ is arbitrary function.

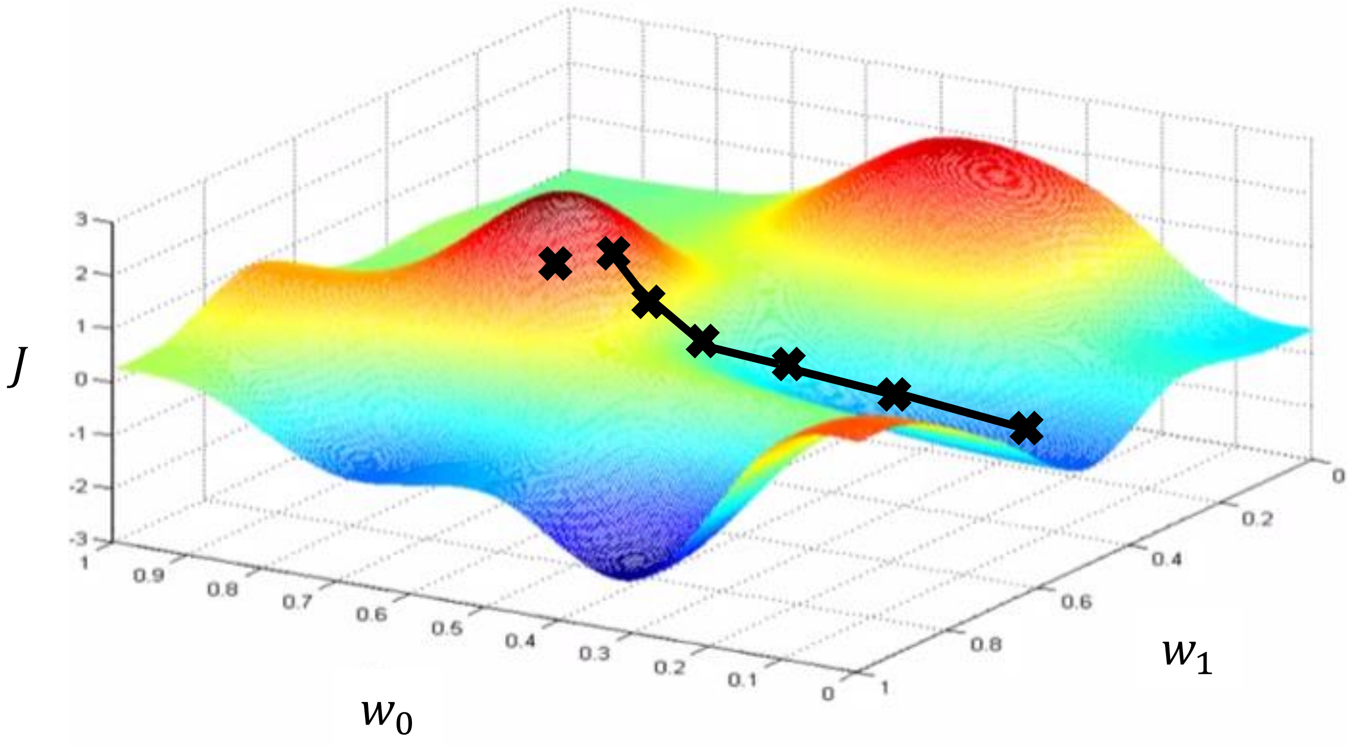
Our goal is to find a minimum $J\ (w_0, w_1)$.

Main idea:

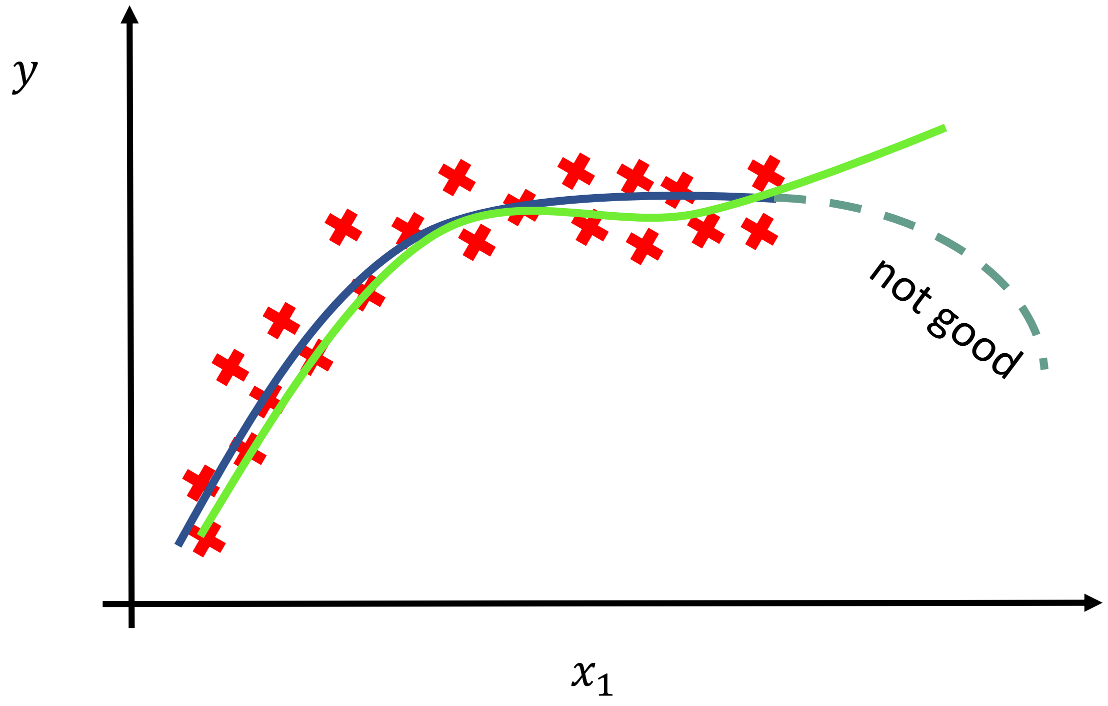
Initial values w_0 and w_1 is chosen randomly.

Usually $\mathrm{w}_0\ = w_1\ =\ 0.$

We change the values of w_0 and w_1 in order to reduce $J(w_0, w_1)$.



Gradient descent



$$h_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

$$h_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$$

$$x_1 = x_1$$
$$x_2 = x_1^2$$
$$x_3 = x_1^3$$

$$h_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

x_1, x_2

$$x_k = x_1^{i-j} x_2^j$$
$$i = 0 \dots s, j = 0 \dots s$$

Gradient descent

repeat until convergence {

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) \qquad \text{(for } j = 0,\ j = 1\text{)}.$$

}

α – learning rate.

w_0, w_1 are updated simultaneously!

Correct!

$$temp0 := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$
$$temp1 := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$
$$w_0 := temp0$$
$$w_1 := temp1$$

Incorrect!

$$temp0 := w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$
$$w_0 := temp0$$
$$temp1 := w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$
$$w_1 := temp1$$

Gradient descent

$$h_w(x) = w_0 + w_1 x$$

$$\frac{\partial}{\partial w_j} J(w_0, w_1) = \frac{\partial}{\partial w_j} \frac{1}{2m} \sum_{i=1}^{m} \left(w_0 + w_1 x^{(i)} - y^{(i)} \right)^2$$

$$w_0: \quad \frac{\partial}{\partial w_0} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)} \right)$$

$$w_1: \quad \frac{\partial}{\partial w_1} J(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

repeat until convergence {

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\theta_0, \theta_1)$$

}

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Gradient descent

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

Repeat until convergence {

$$w_j := w_j - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}}_{\frac{\partial}{\partial w_j} J(\mathrm{W})}$$

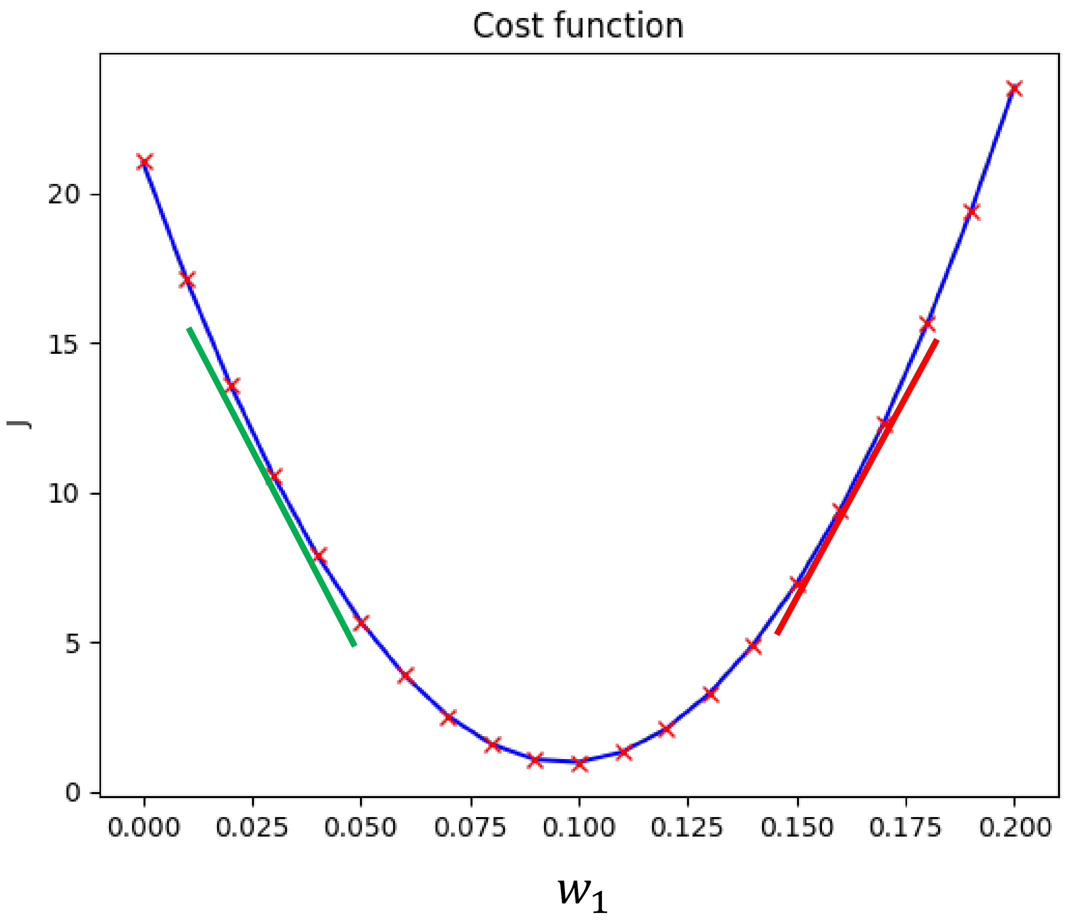
}

$$w_0 := w_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right) x_0^{(i)}$$

$$w_1 := w_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right) x_1^{(i)}$$

$$w_2 := w_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right) x_2^{(i)}$$

Gradient descent



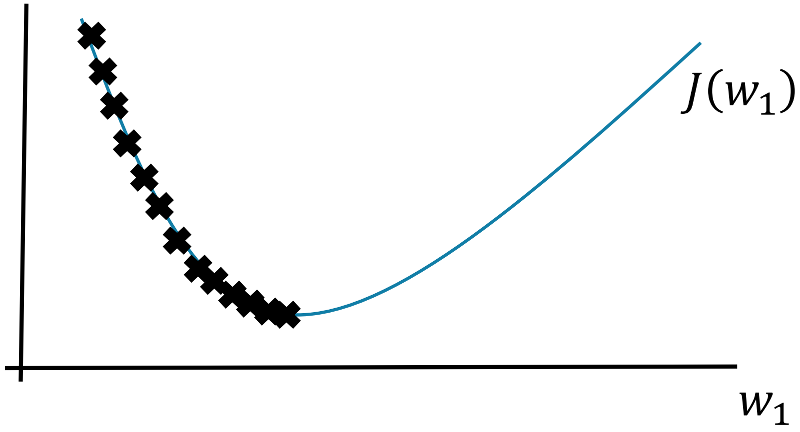
$w_1 - decrease$

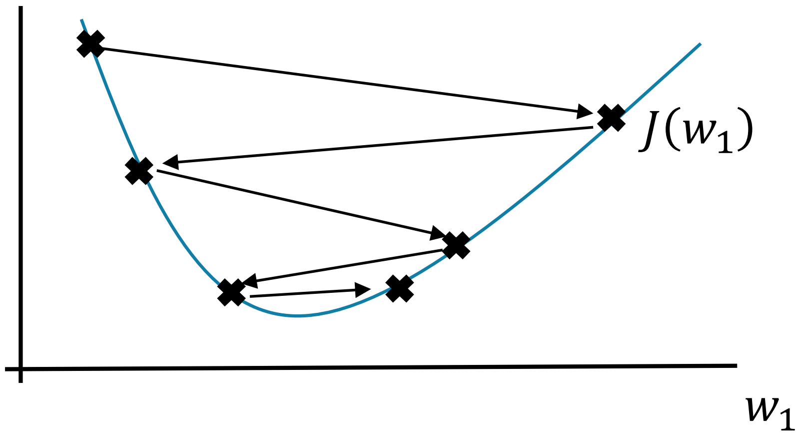
$w_1 - increase$

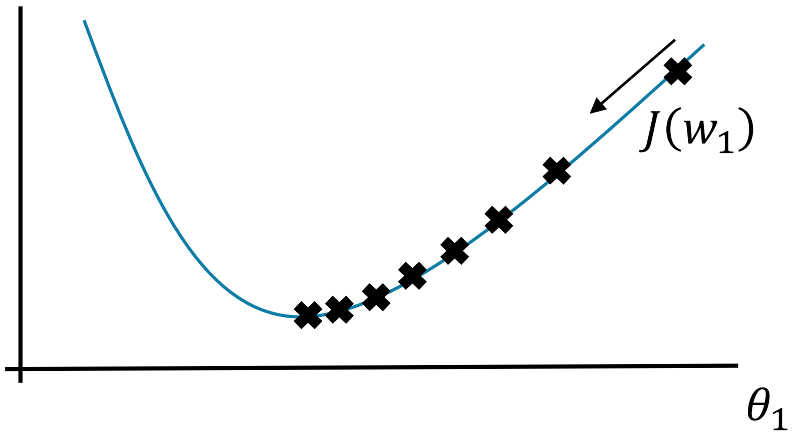
$$w_1 \coloneqq w_1 - \alpha \boxed{\frac{\mathrm{d}}{\mathrm{d}w_1} J(w_1)} \begin{array}{l} \geq 0 \\ \leq 0 \end{array}$$

Gradient descent

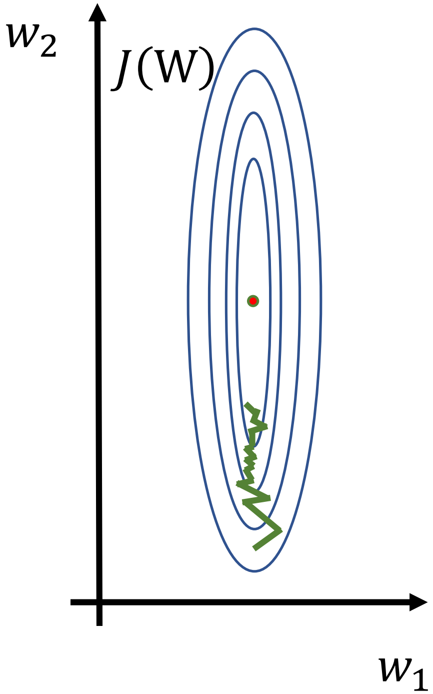
$$w_1 \coloneqq w_1 - \alpha \frac{\mathrm{d}}{\mathrm{d}w_1} J(w_1)$$



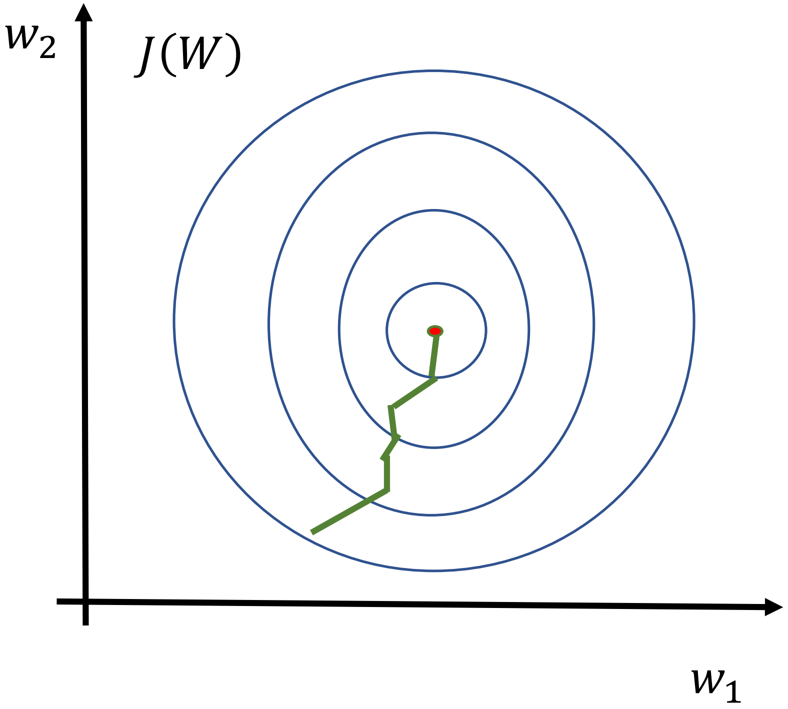




Gradient descent



$x_1 = 500 \ldots 2000$
$x_2 = 1 \ldots 5$

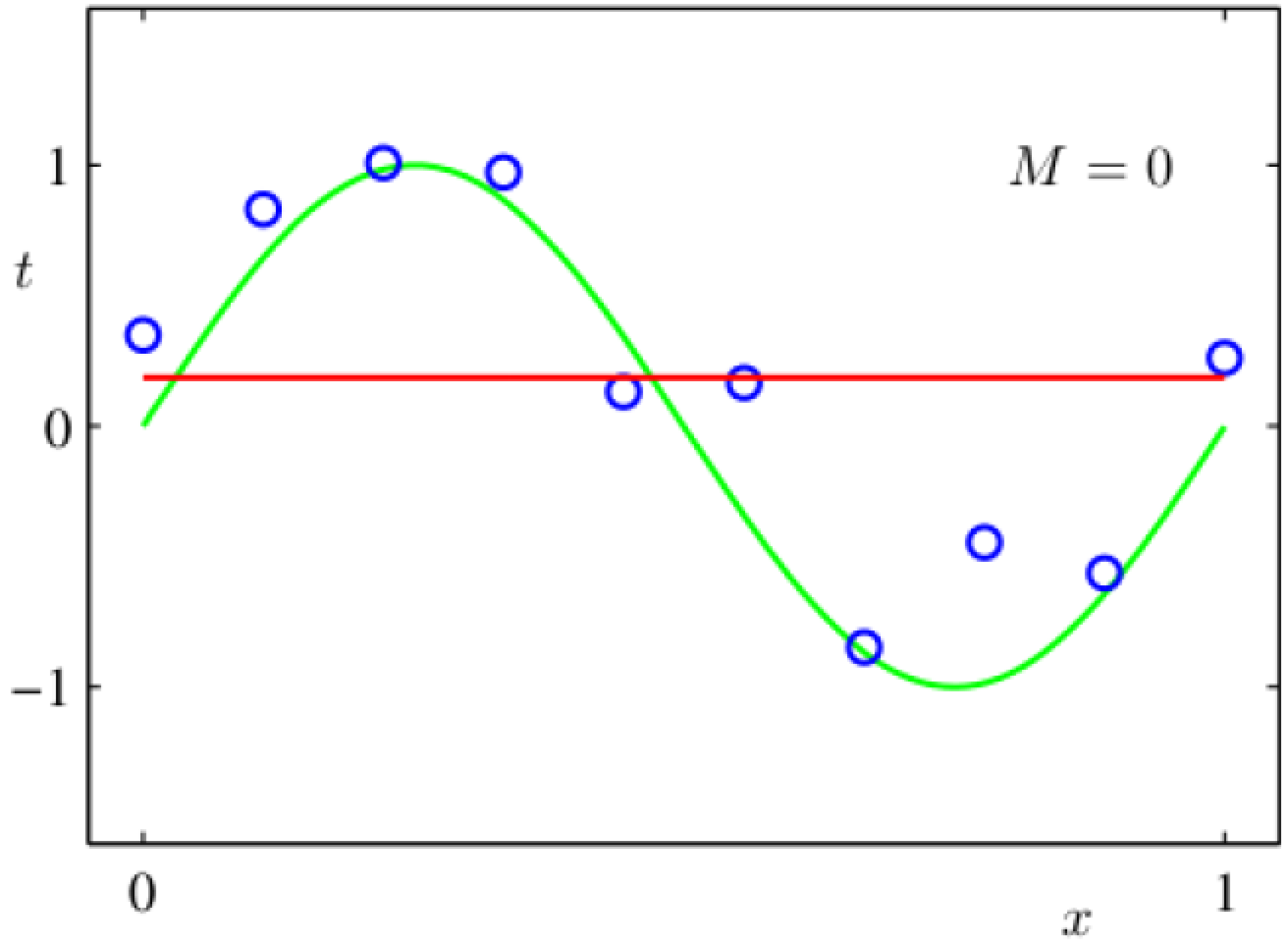


for $j \geq 1$

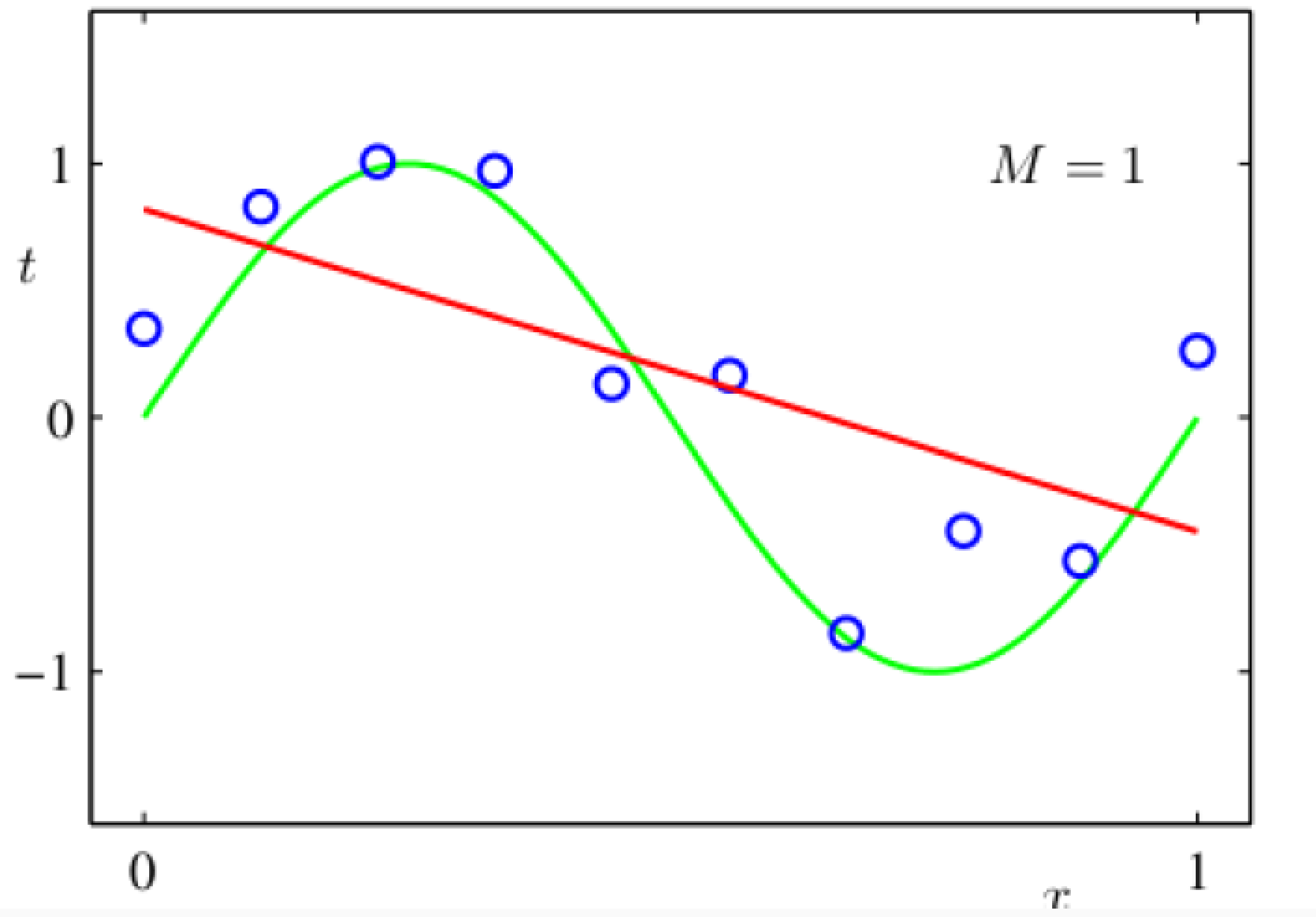
$$x_j = \frac{x_j - \mu}{\sigma}$$

μ − mean value x_j

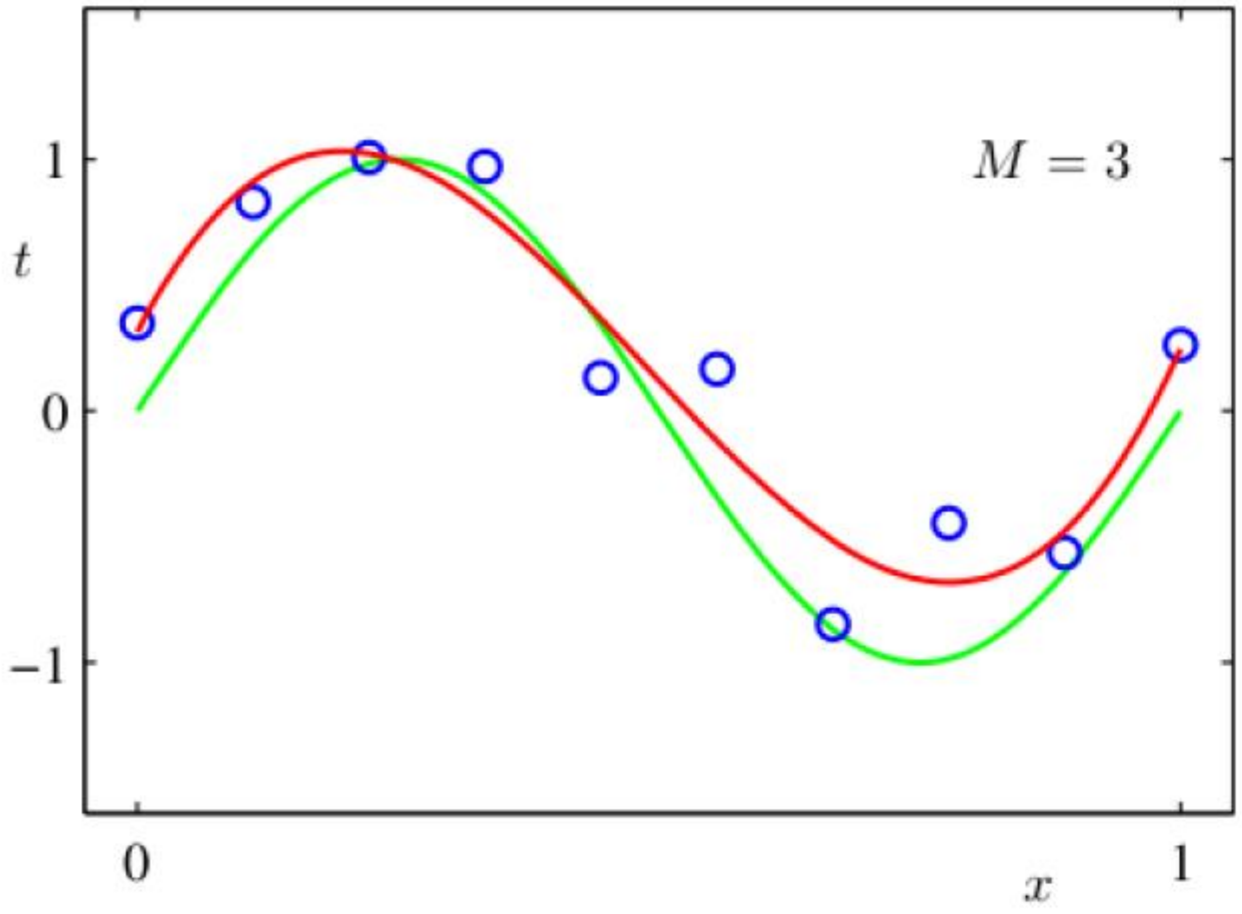
Polynomial regression



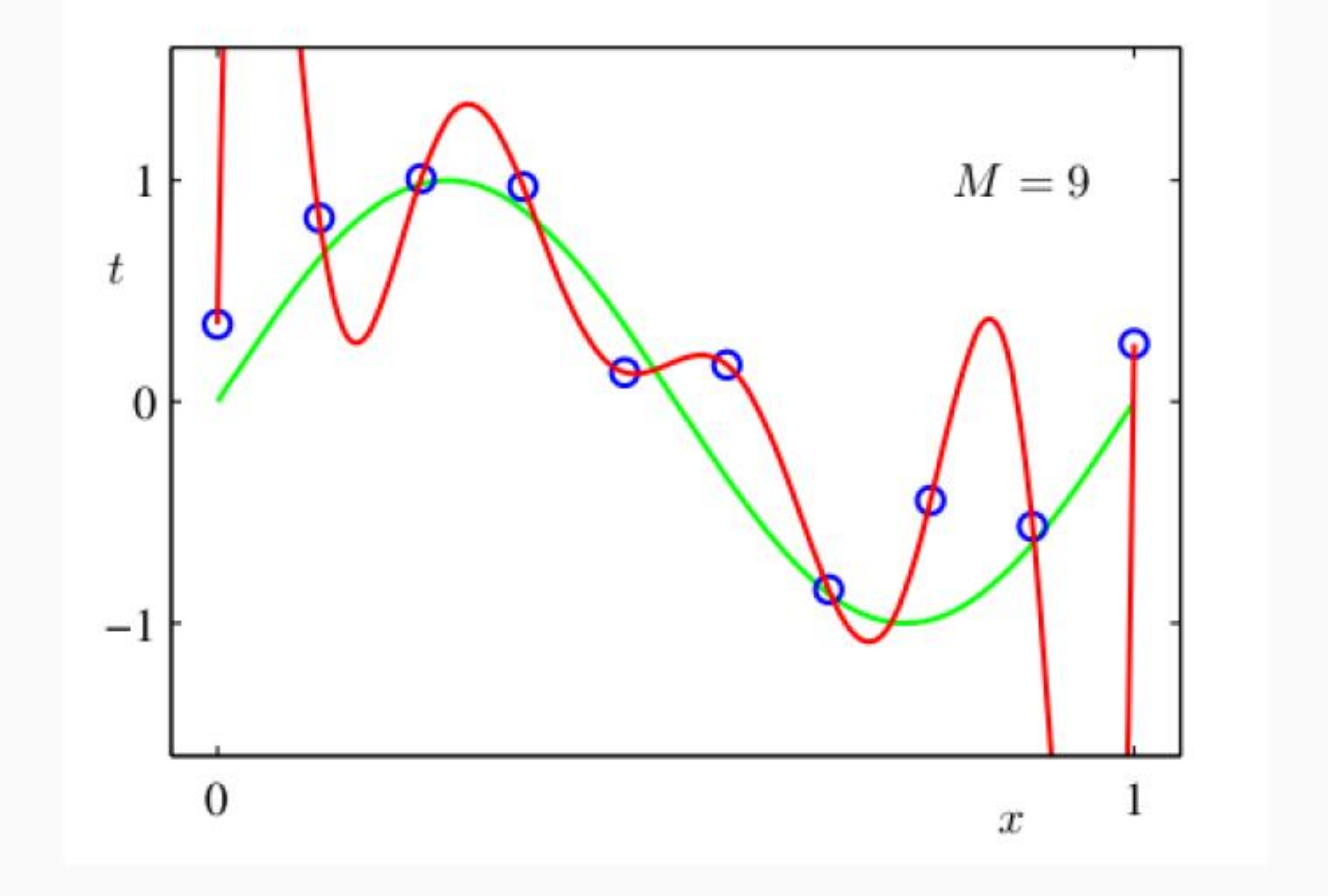
Polynomial regression



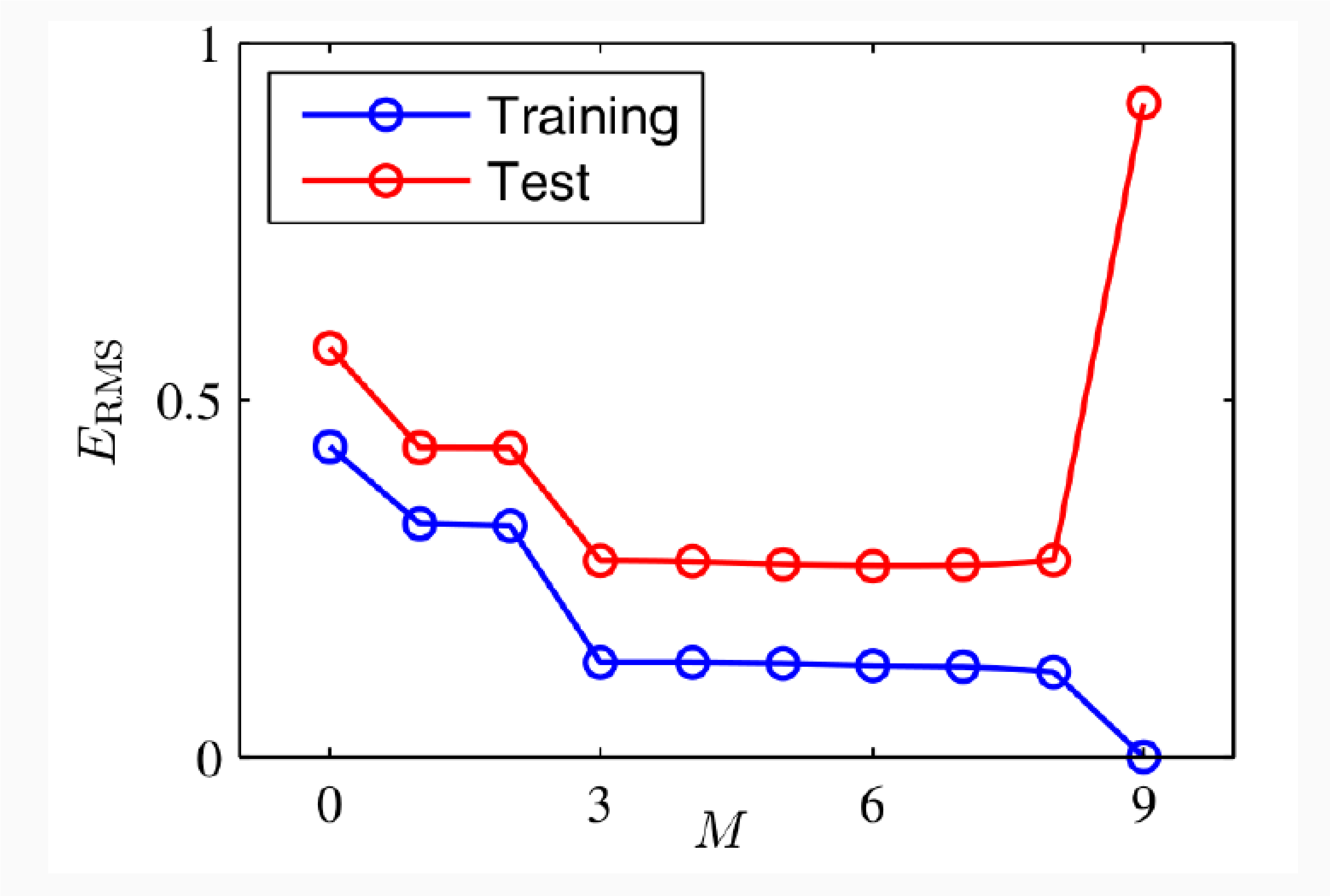
Polynomial regression



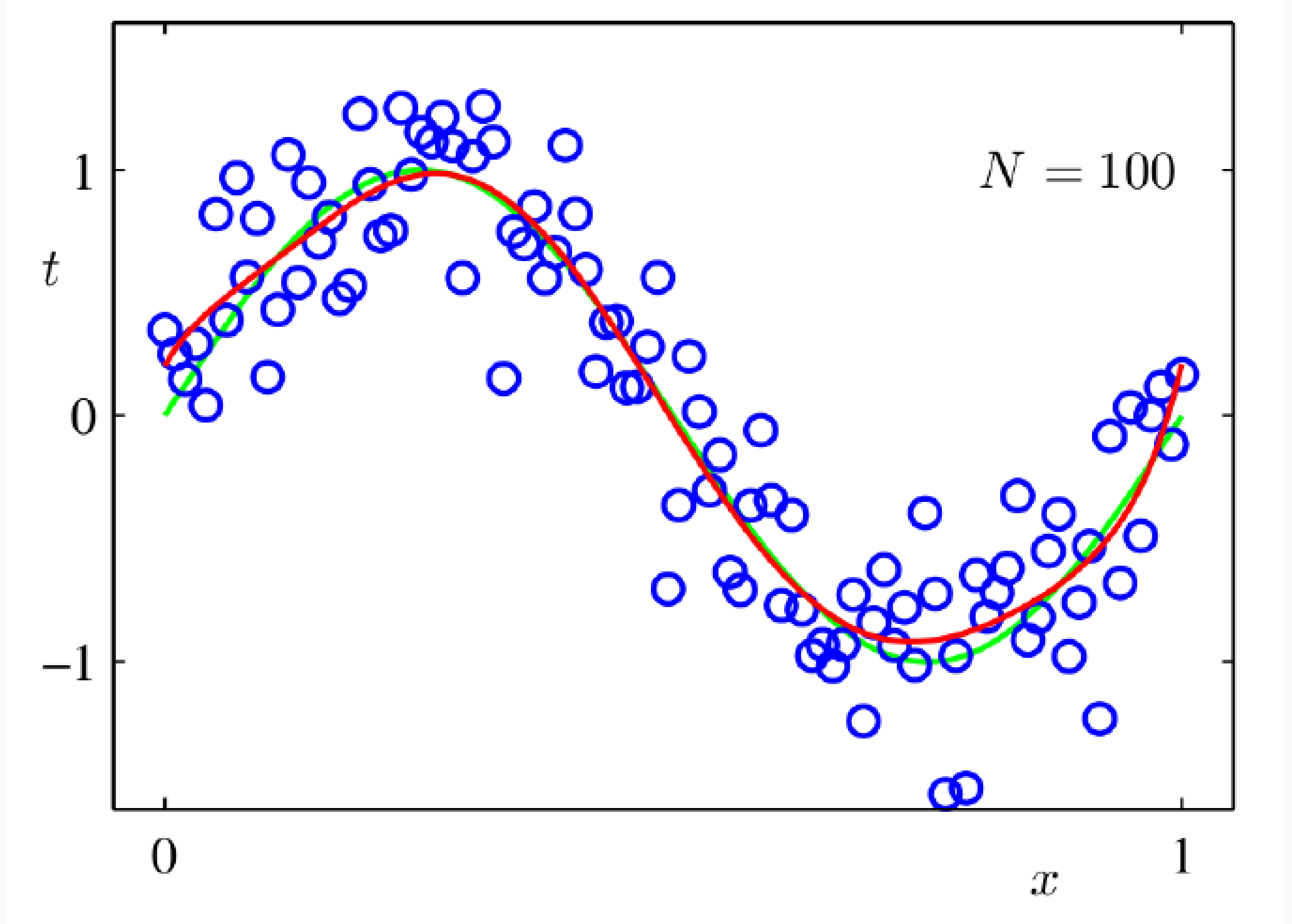
Polynomial regression



Root mean square Error



How to deal with it? Add data

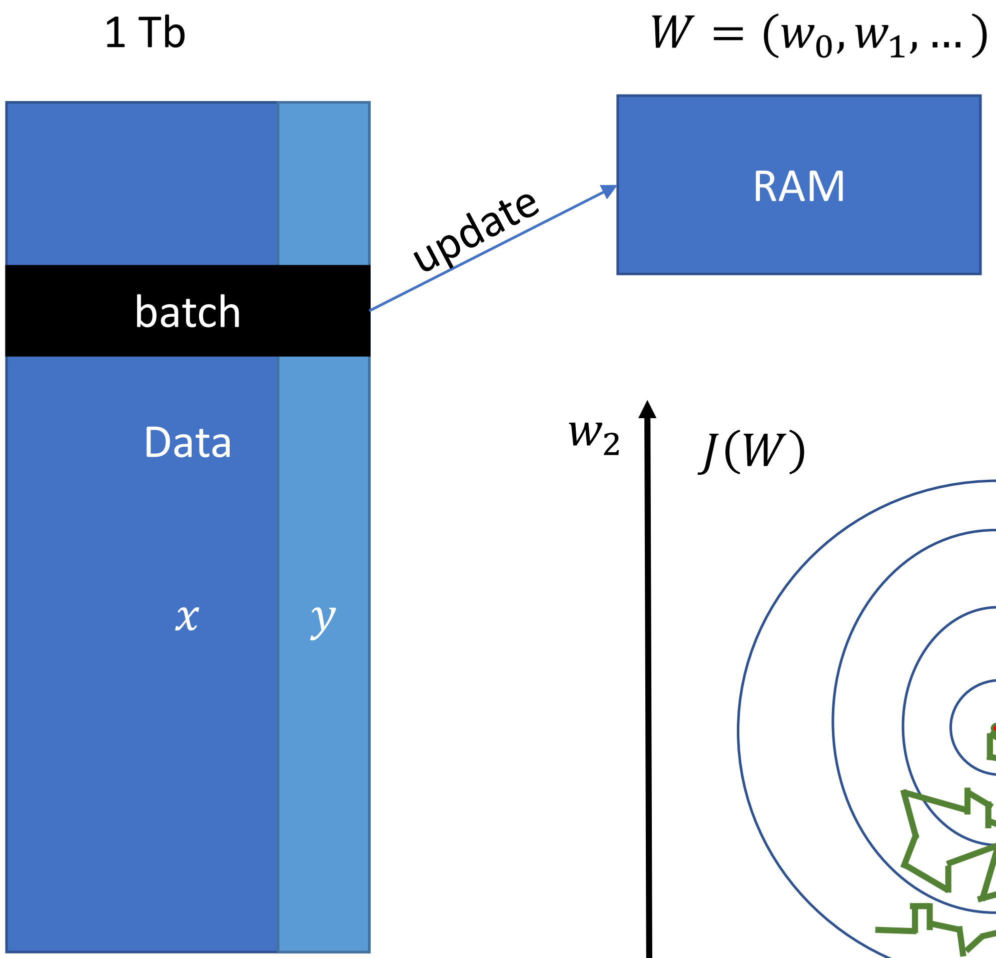


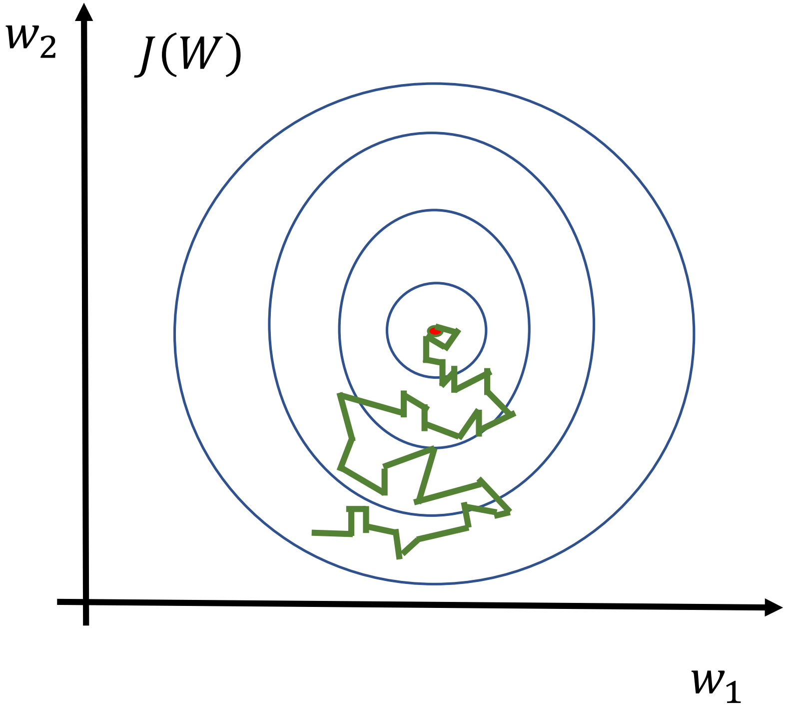
Regulirization

OLD: $$J(W) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

NEW: $$J(W) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}||w^2||$$

Stochastic gradient descent (SGD)





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

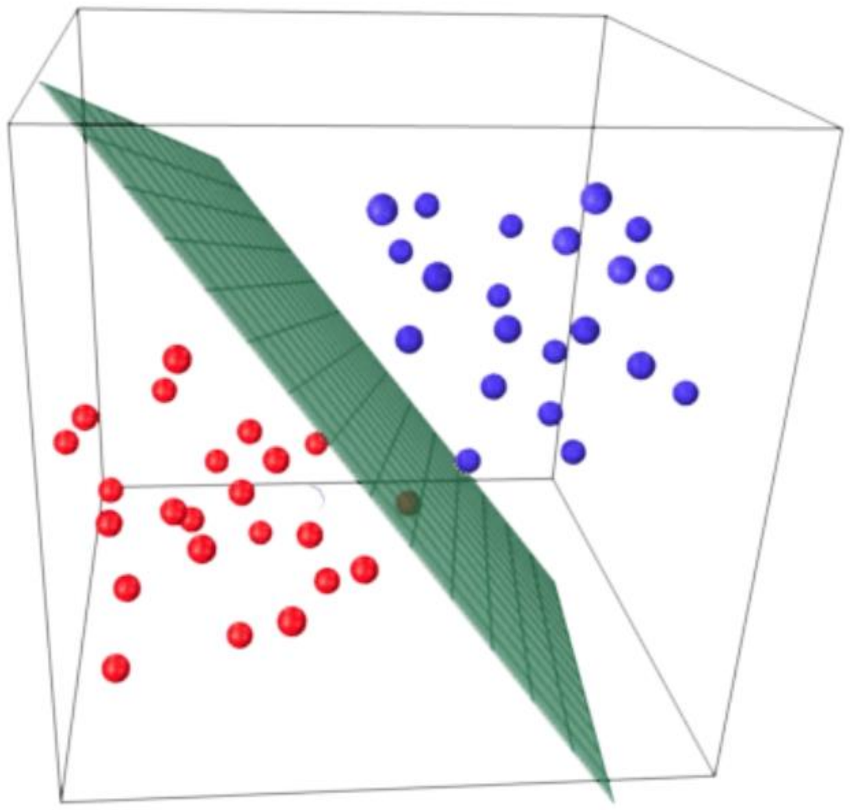
Repeat until convergence {

$$\theta_j := \theta_j - \alpha(h_w(x^{(i)}) - y^{(i)})x_j^{(i)}$$

}

Classification

Classification



Email: Spam / Not Spam
Online transactions: Fraud (Yes / No)
Tumor: Benign / Malignant

Classes can be many:

$$y \in \{0,1,2,3,\ldots\}$$

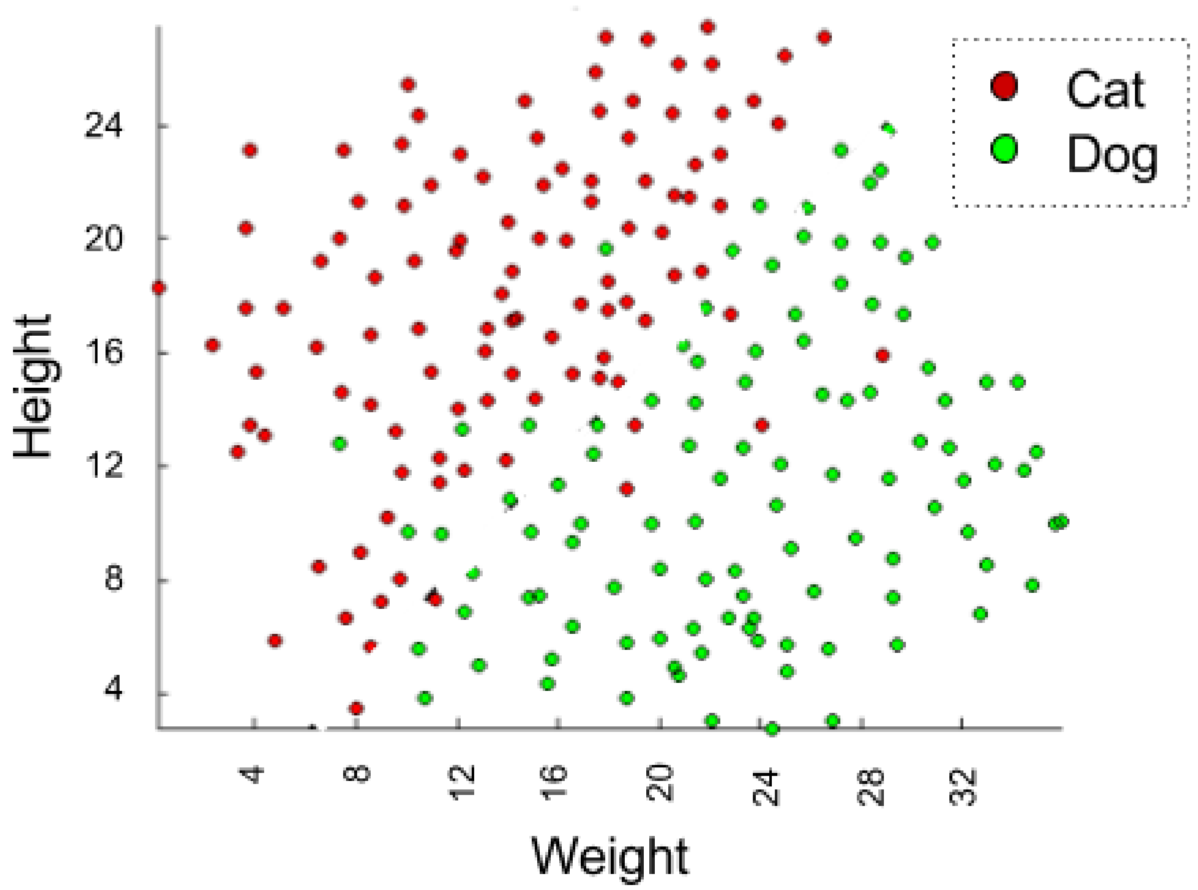
The basic idea behind the linear classifier is that the two values of the target class can be separated by a hyperplane in the feature space.

Classification

Data Schema

Continuous Variables		Discrete Variable
Height (in)	Weight (lb)	Species
20.4,	28.4,	Dog,
10.2,	8.9,	Cat,
...197 more entries...		
18.0,	0.0,	Cat,

Data Visualization



Classification

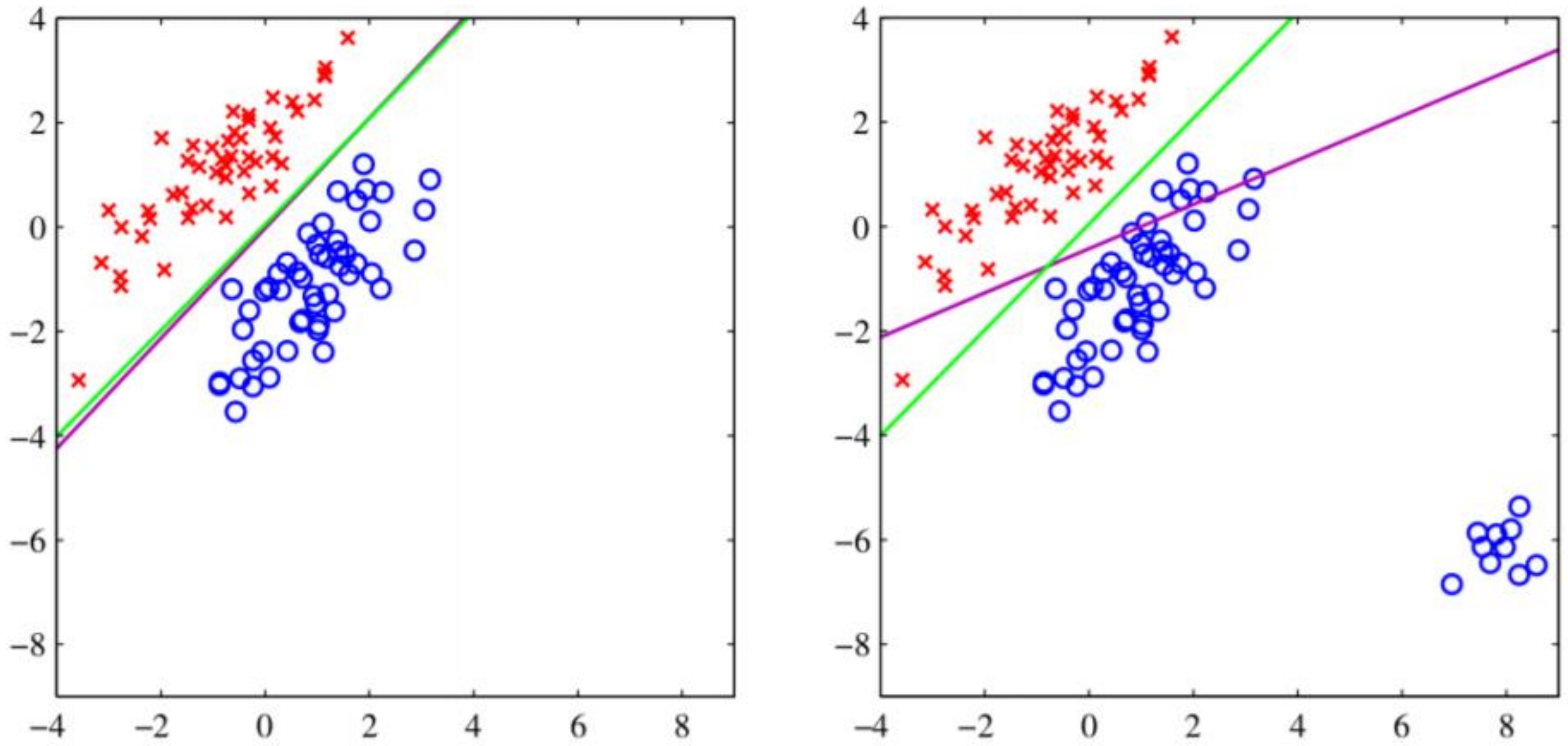
- Our task is to compare the vector of features **x** with one of the known classes
- Feature space is somehow divided into classes
- We are actually want to find a **decision surface** (or decision boundary).
- How to code classes? If we have two classes, then it is done in a very natural way. We define the variable y.

$$\mathrm{y} = 0 \text{ corresponds to } C_1; \quad \mathrm{y} = 1 \text{ corresponds to } C_2$$

- The value y can be interpreted as the **probability**
- If there are several classes, it is convenient to enter a vector

$$\boldsymbol{y} = (0,0, \ldots 1, \ldots, 0, 0)$$

Binary classification. OLS



Why does least squares work so badly?

The ordinary least squares implies Gaussian error distribution.
But binary vectors are not distributed in Gauss

Logistic regression

Let's $P(x)$ - the probability of an event x

The odds ratio

$$OR(x) = \frac{P(x)}{1 - P(x)}$$

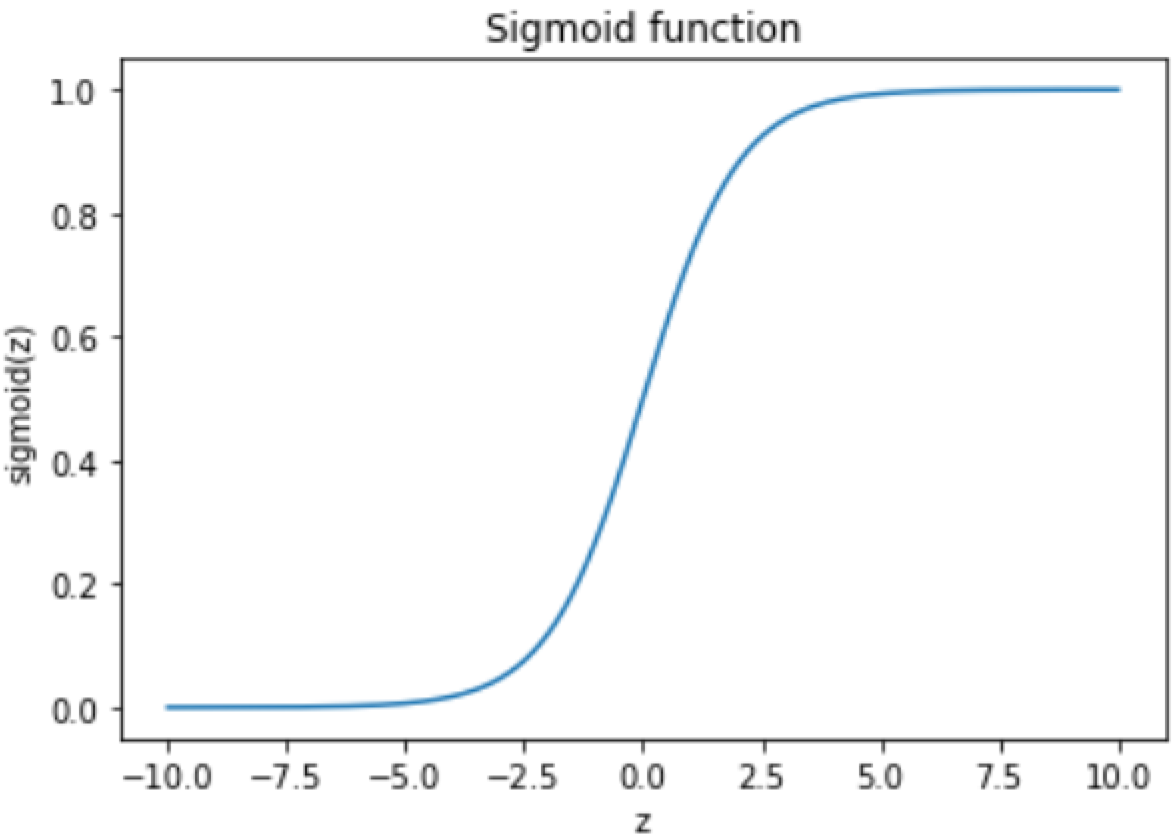
is the ratio of the probabilities of whether or not an event will happen.

why?

$$P(x) = (0,1)$$
$$OR(x) = (0, \infty)$$

profit!

we predict $\log\big(OR(x)\big) \in \mathbb{R}$



How logistic regression will make a prediction?

$p^{(i)} = P(y_i = 1 \mid x_i, w)$, $\text{let's assume that we have somehow obtained weights } W$

$$\log(OR) = \log\left(\frac{p^{(i)}}{1 - p^{(i)}}\right) = w_0 + w_1 x_1 + w_2 x_2 = \mathrm{W}^T X^{(i)}$$

some algebra…

$$h_{\mathrm{w}}(x) = p^{(i)} = \frac{e^{\mathrm{w}^T X^{(i)}}}{1 + e^{\mathrm{w}^T X^{(i)}}} = \frac{1}{1 + e^{-\mathrm{w}^T X^{(i)}}} \stackrel{\text{def}}{=} \sigma(\mathrm{w}^T X^{(i)})$$

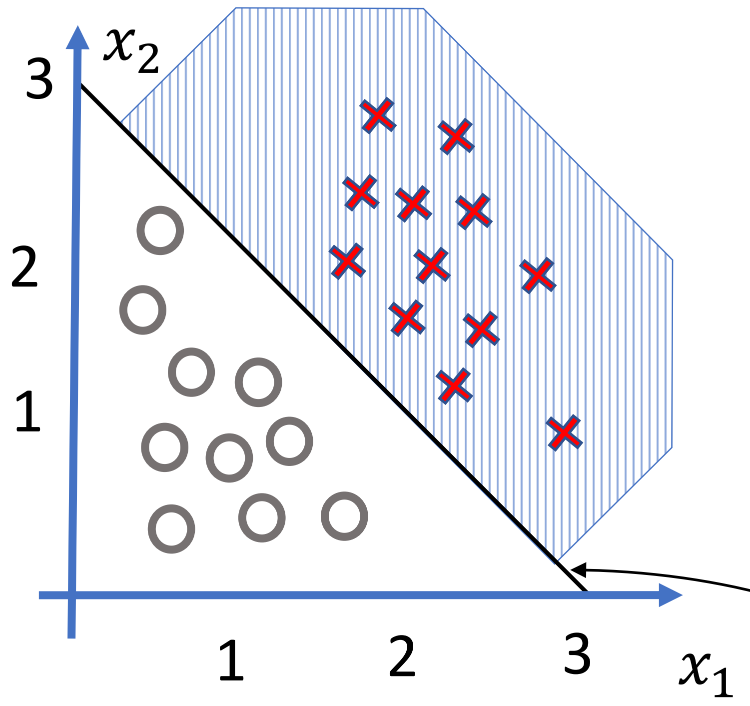
the sigmoid function.

Logistic regression

So, logistic regression predicts the probability of assigning an example to the $"+"$ class (assuming that we know the features and weights of the model) as a sigmoid transformation of a linear combination of the weight vector and the feature vector:

$$p^{+}(x) = P(y_i = 1 \mid x_i, w) = \sigma(w^T x_i).$$

Decision boundary



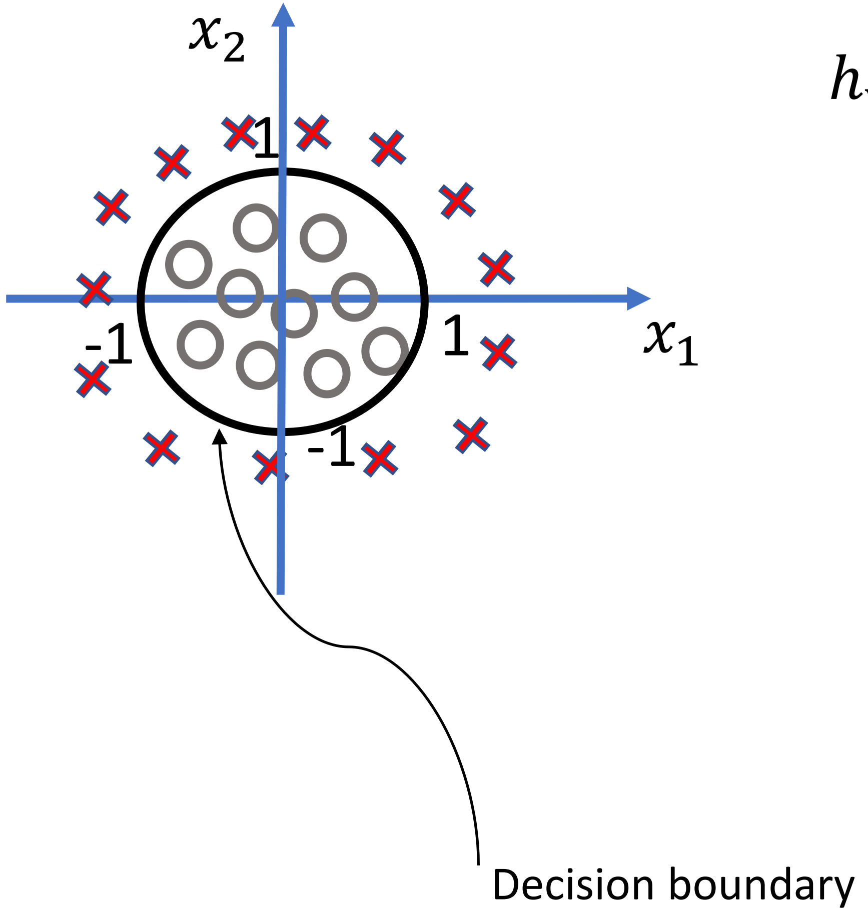
- $h_{\mathrm{W}}(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$

$$\mathrm{w} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$y = 1\colon\ \mathrm{W}^T x = -3 + x_1 + x_2 \geq 0$$
$$y = 0\colon\ \mathrm{W}^T x = -3 + x_1 + x_2 \leq 0$$

Decision boundary ($h_{\mathrm{w}}(x) = 0$)

Decision boundary



$$h_{\mathrm{w}}(x) = g(\mathrm{w}_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2)$$

$$\mathrm{W} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$y = 1\colon\ \mathrm{w}^T x = -1 + w_1^2 + w_2^2 \geq 0$$

$$y = 0\colon\ \mathrm{w}^T x = -1 + w_1^2 + w_2^2 \leq 0$$

Likelihood maximization

Lets there is the dataset $D = \{\bar{x}_n, y_n\}, y_n \in \{0,1\}$

How to find the likelihood of Data $p(D|w)$?

$$p(D|w) = \Pi\, p(d|w) = \Pi_{d \in C_1}\, p(C_1|w) \cdot \Pi_{d \in C_2}\, p(C_2|w)$$

Use the trick :

$$p(d|w) = \left.\begin{matrix} p(C_1|w) \\ 1 - p(C_1|w) \end{matrix}\right\} = p(C_1|w)^y \big(1 - p(C_1|w)\big)^{1-y}$$

When:

$$p(D|w) = \Pi_\mathrm{n}\, p(C_1|w)^y \big(1 - p(C_1|w)\big)^{1-y} = \Pi_\mathrm{n}\, \sigma(w^T x_n)^{y_n} \big(1 - \sigma(w^T x_n)\big)^{1-y_n}$$

Maximize by w

Cost function

$$J(\mathrm{w}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}\left(h_{\mathrm{w}}\left(x^{(i)}\right), y^{(i)}\right)$$

$$\mathrm{Cost}(h_{\mathrm{w}}(x), y) = \begin{cases} -\log\left(h_{\mathrm{w}}(x)\right), \text{если } y = 1 \\ \log\left(1 - h_{\mathrm{w}}(x)\right), \quad \text{если } y = 0 \end{cases}$$

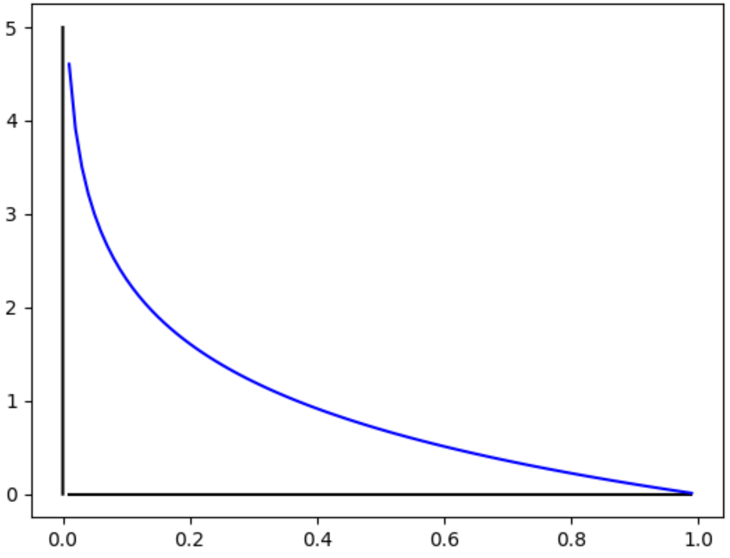
$$h_{\mathrm{w}}(x) = \sigma(\mathrm{w}^T x) = \frac{1}{1 + e^{-\mathrm{w}^T x}}$$

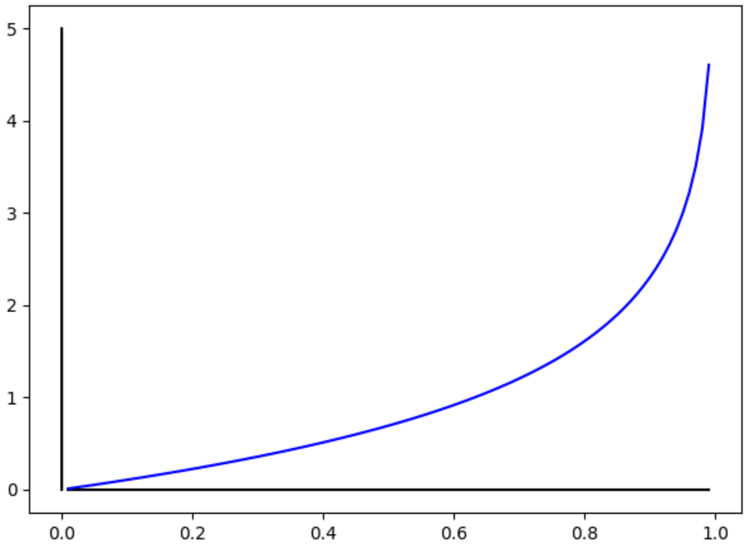
$=$

$$J(\mathrm{w}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}\,(h(x), y) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_{\mathrm{w}}\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_{\mathrm{w}}\left(x^{(i)}\right)\right)\right]$$

Cost function

$$\text{Cost}=\begin{cases} -\log\left(h_{\mathrm{w}}(x)\right) \; if \; y=1 \\ \log\left(1-h_{\mathrm{w}}(x)\right) \; if \; y=0 \end{cases}$$





If $y=1$

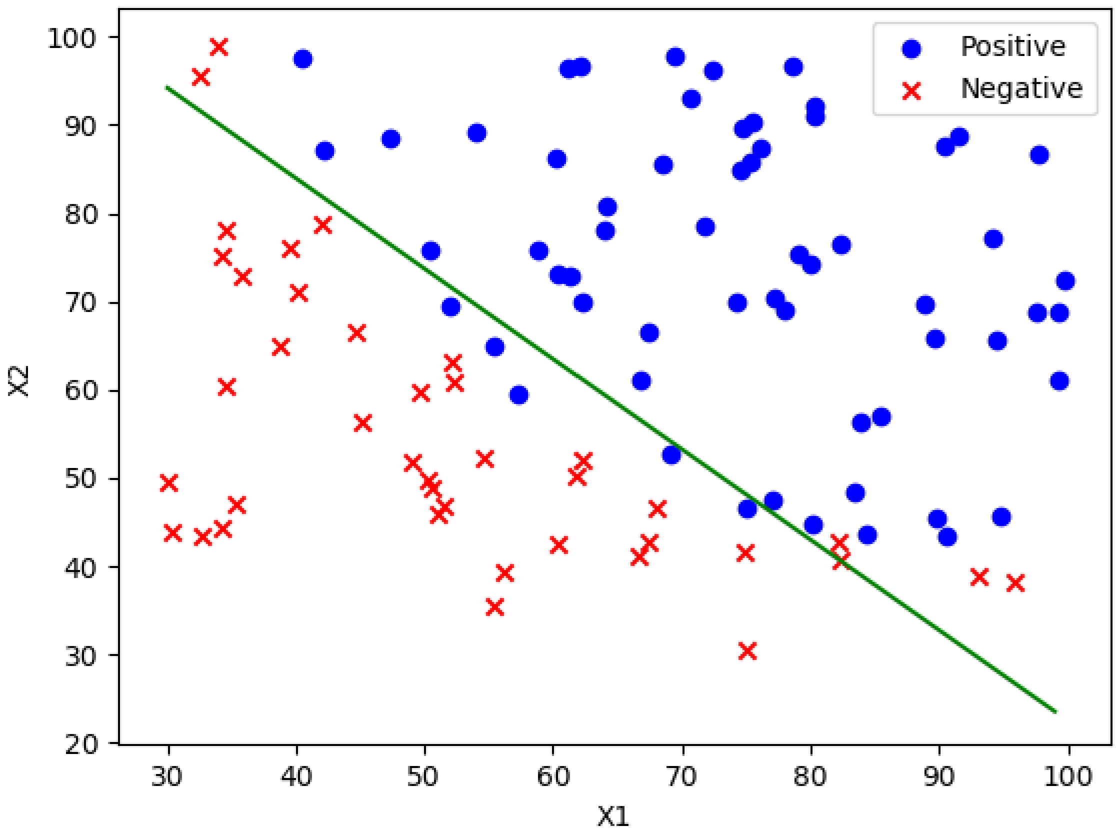
$$\text{Cost} = 0 \; if \; h_{\mathrm{w}}(x) = 1$$

Cost $\rightarrow \infty$ if $h_{\mathrm{w}}(x) \rightarrow 0$

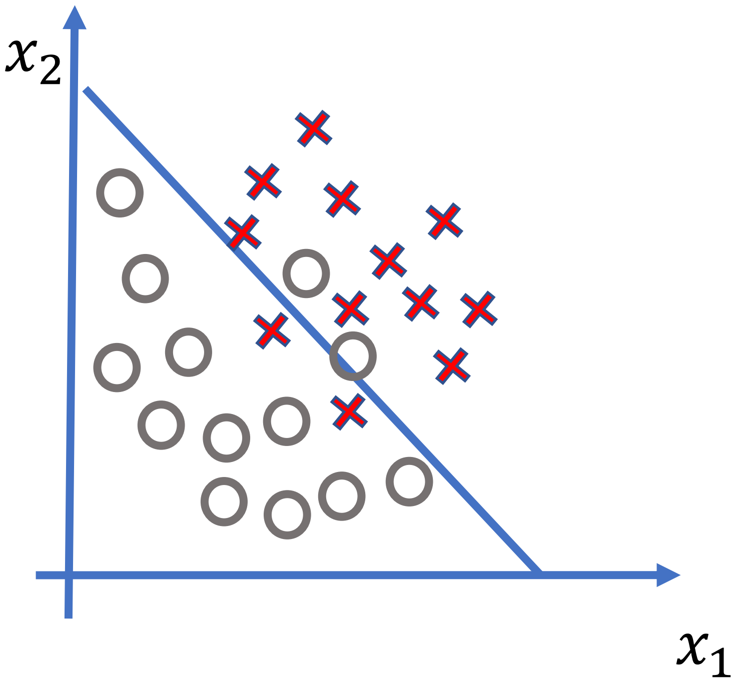
Classification

We use gradient dissent to minimize cost function

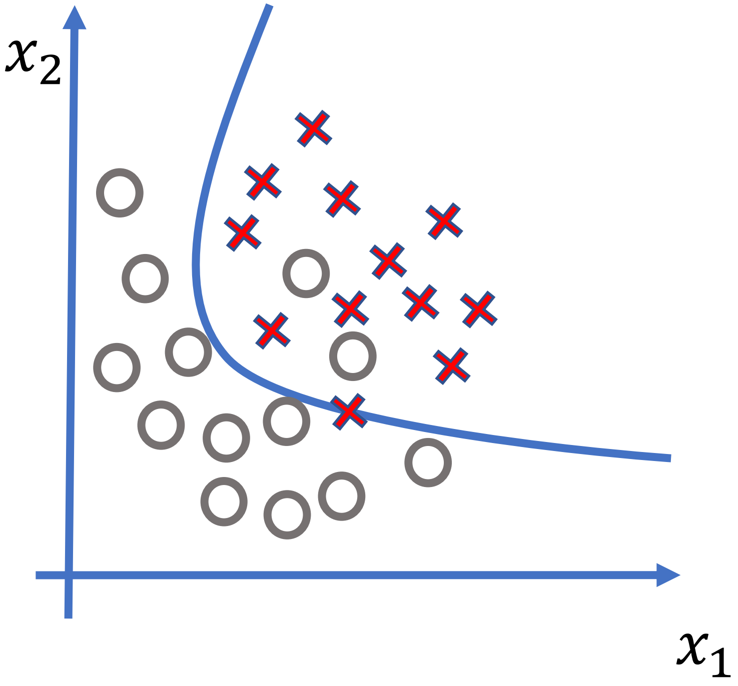
$$J'(\mathrm{w}) = \sum \big(y_n - \sigma(w^T \bar{x})\big)\bar{x}$$



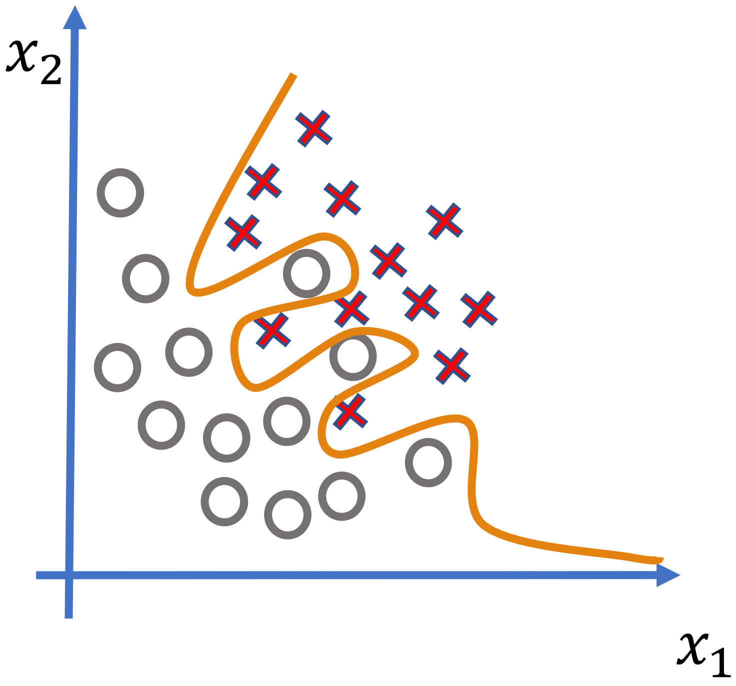
Overfitting



$h_w(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$



$h_w(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

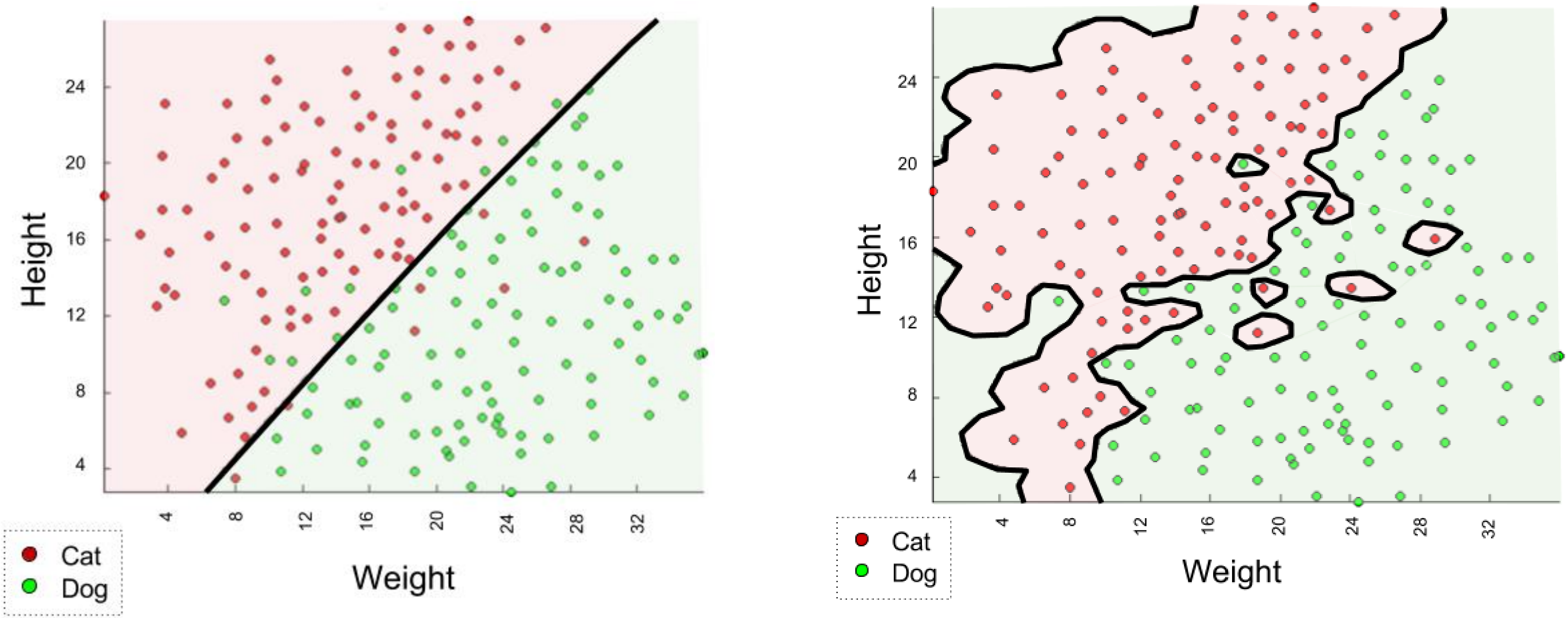


$h_w(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 x_2 + \theta_3 x_1^2 x_2^2 + \theta_4 x_2^2 x_3^2 + \theta_5 x_1^3 x_2 + \cdots)$

Underfit

Overfit

Overfitting



Regularization

The goal is to reduce the value of parameters $w_0, w_1 \dots w_n$

$$J(\mathrm{w}) = \frac{1}{2m}\left[\sum_{i=0}^{m}\left(h_{\mathrm{w}}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

- λ is a regularization parameter

if $\lambda \gg 1$ $(\lambda \approx 10^{10})$, то $w_1 \approx 0, w_2 \approx 0, w_3 \approx 0, \dots$

$h_{\mathrm{w}}(x) \approx w_0$

Gradient descent

repeat until convergence {

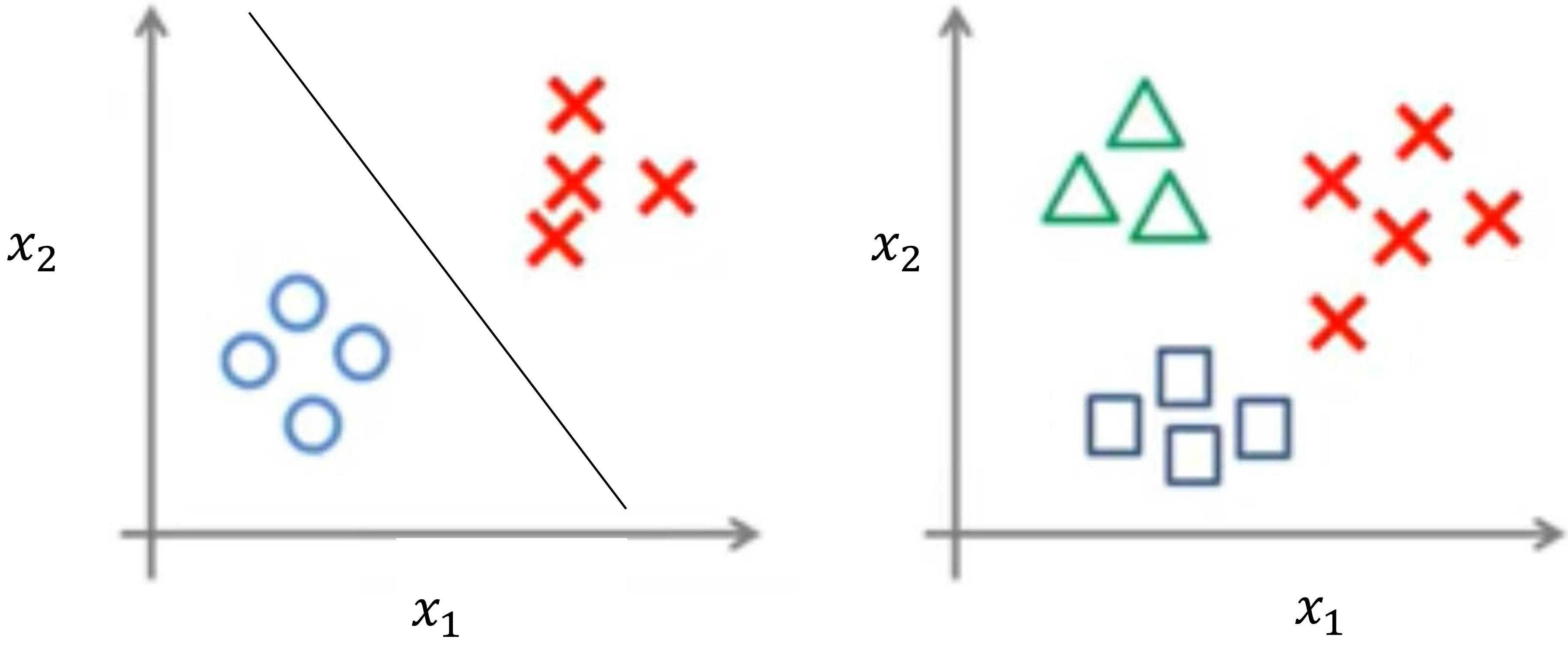
$$w_0 := w_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

$$w_j := w_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left[(h_w\left(x^{(i)}\right) - y^{(i)})x^{(i)} + \frac{\lambda}{m}w_j\right]$$

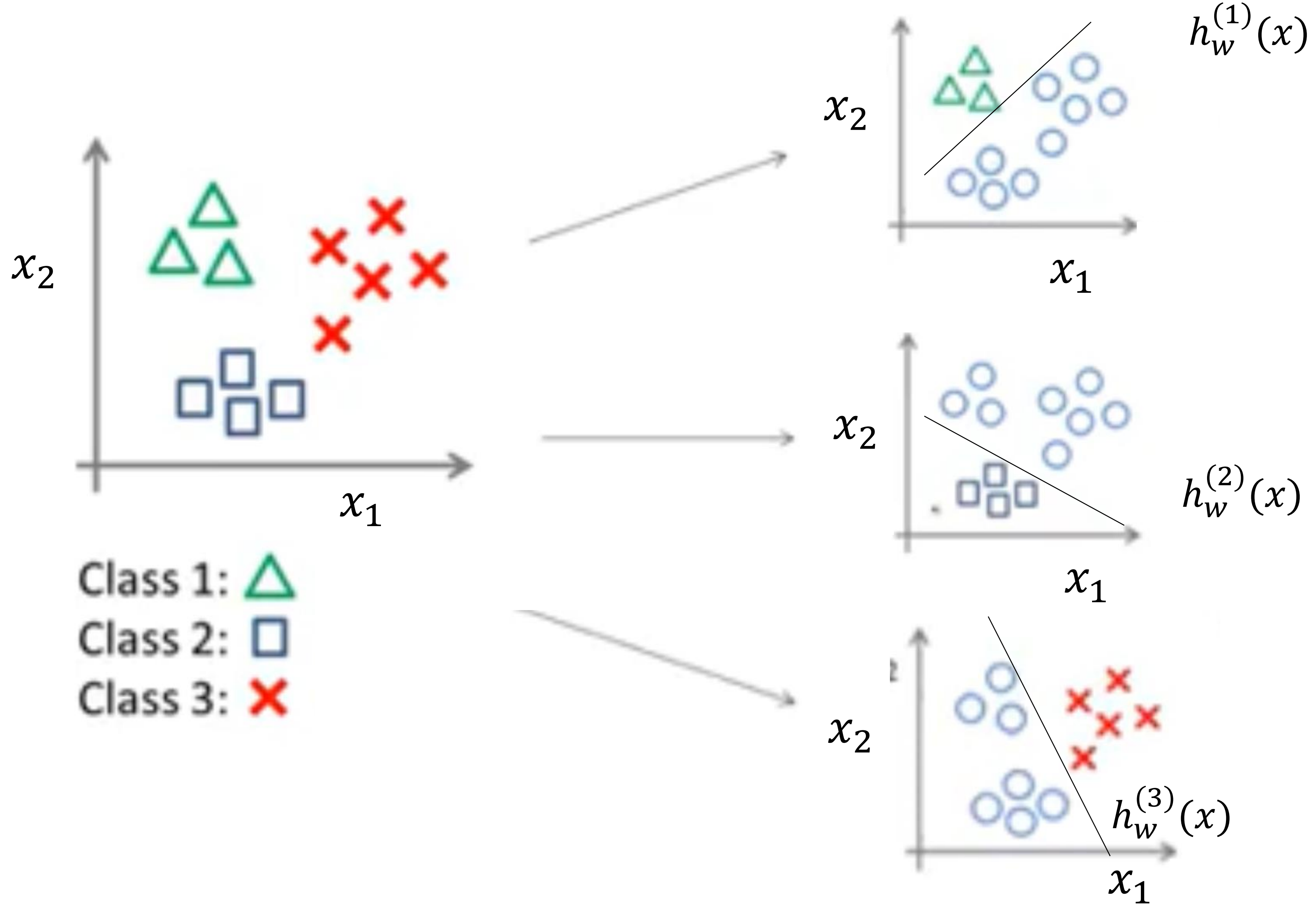
}

$$w_j := w_j\left(1 - \frac{\lambda}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

Multi-class classification



Multi-class classification



Multi-class classification

Train the logistic regression classifier $h_w(x)$ for each class i
to predict the probability that $y = i$

On a new input x to make a prediction, pick the class i that maximizes

$$\max_i h_w^{(i)}(x)$$

Multi-class classification

We want:

$$h_{\mathrm{w}}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

man

$$h_{\mathrm{w}}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

car

$$h_{\mathrm{w}}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

bike

$$h_{\mathrm{w}}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

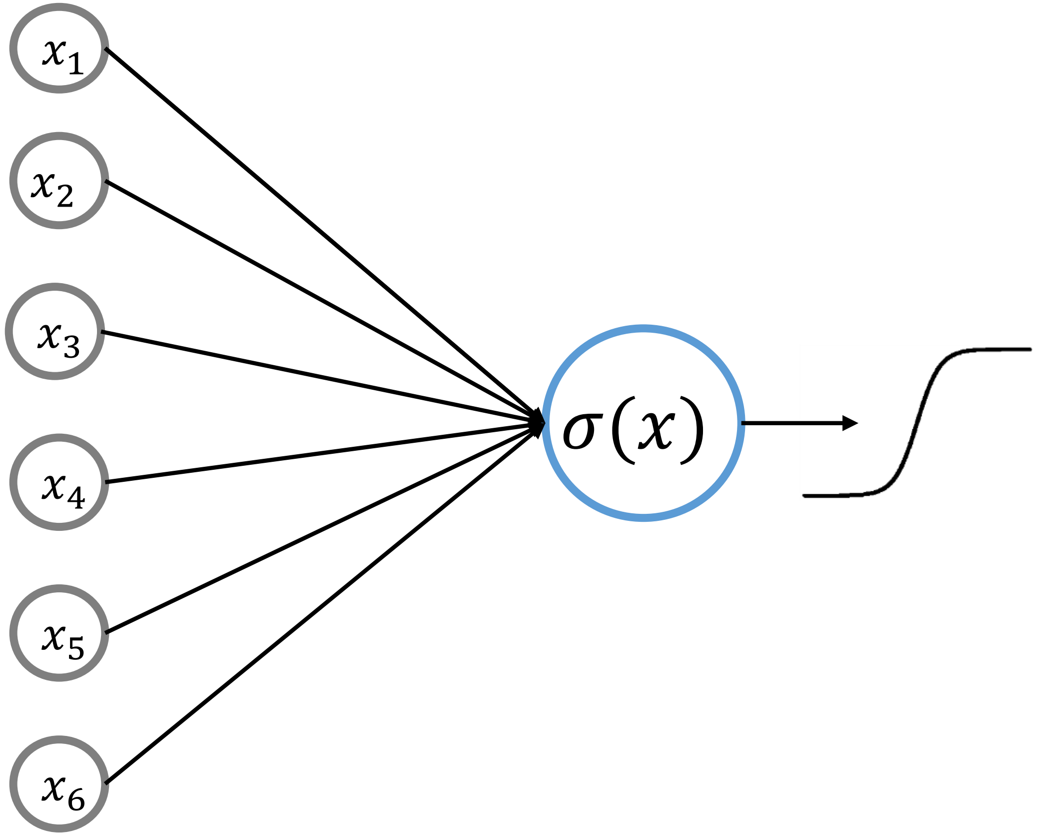
airplan

One-vs-all

Data train:

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$$

$$y^{(j)} = [0,1]$$



One-vs-all

Data train :

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$$y^{(j)} \in \left\{ \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \right\}$$